



Data Analytics for IT Networks

Developing Innovative Use Cases

Data Analytics for IT Networks

Developing Innovative Use Cases

John Garrett CCIE Emeritus No. 6204, MSPA

Cisco Press

<https://t.me/learningnets>

Data Analytics for IT Networks

Developing Innovative Use Cases

Copyright © 2019 Cisco Systems, Inc.

Published by:

Cisco Press

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

First Printing 1 18

Library of Congress Control Number: 2018949183

ISBN-13: 978-1-58714-513-1

ISBN-10: 1-58714-513-8

Warning and Disclaimer

This book is designed to provide information about Developing Analytics use cases. It is intended to be a guideline for the networking professional, written by a networking professional, toward understanding Data Science and Analytics as it applies to the networking domain. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED AS PART OF THE SERVICES FOR ANY PURPOSE. ALL SUCH DOCUMENTS AND RELATED GRAPHICS ARE PROVIDED “AS IS”

WITHOUT WARRANTY OF ANY KIND. MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL MICROSOFT AND/OR ITS RESPECTIVE

SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION AVAILABLE FROM THE SERVICES.

THE DOCUMENTS AND RELATED GRAPHICS CONTAINED HEREIN COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED HEREIN AT ANY TIME. PARTIAL SCREEN SHOTS MAY BE VIEWED IN FULL WITHIN THE SOFTWARE VERSION SPECIFIED.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

MICROSOFT® WINDOWS®, AND MICROSOFT OFFICE® ARE REGISTERED TRADEMARKS OF THE MICROSOFT CORPORATION IN THE U.S.A. AND OTHER COUNTRIES. THIS BOOK IS NOT SPONSORED OR ENDORSED BY OR AFFILIATED WITH THE MICROSOFT CORPORATION.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Editor-in-Chief: Mark Taub

Alliances Manager, Cisco Press: Arezou Gol

Product Line Manager: Brett Bartow

Managing Editor: Sandra Schroeder

Development Editor: Marianne Bartow

Project Editor: Mandie Frank

Copy Editor: Kitty Wilson

Technical Editors: Dr. Ammar Rayes, Nidhi Kao

Editorial Assistant: Vanessa Evans

Designer: Chuti Prasertsith

Composition: codemantia

Indexer: Erika Millen

Proofreader: Abigail Manheim



Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

 Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

About the Author

John Garrett is CCIE Emeritus (6204) and Splunk Certified. He earned an M.S. in predictive analytics from Northwestern University, and has a patent pending related to analysis of network devices with data science techniques. John has architected, designed, and implemented LAN, WAN, wireless, and data center solutions for some of the largest Cisco customers. As a secondary role, John has worked with teams in the Cisco Services organization to innovate on some of the most widely used tools and methodologies at Customer Experience over the past 12 years.

For the past 7 years, John's journey has moved through server virtualization, network virtualization, OpenStack and cloud, network functions virtualization (NFV), service assurance, and data science. The realization that analytics and data science play roles in all these brought John full circle back to developing innovative tools and techniques for Cisco Services. John's most recent role is as an Analytics Technical Lead, developing use cases to benefit Cisco Services customers as part of Business Critical Services for Cisco. John lives with his wife and children in Raleigh, North Carolina.

About the Technical Reviewers

Dr. Ammar Rayes is a Distinguished Engineer at Advance Services Technology Office for Cisco, focusing on network analytics, IoT, and machine learning. He has authored 3 books and more than 100 publications in refereed journals and conferences on advances in software- and networking-related technologies, and he holds more than 25 patents. He is the founding president and board member of the International Society of Service Innovation Professionals (www.issip.org), editor-in-chief of the journal *Advancements in Internet of Things* and an editorial board member of the European Alliance for Innovation—Industrial Networks and Intelligent Systems. He has served as associate editor on the journals *ACM Transactions on Internet Technology* and *Wireless Communications and Mobile Computing* and as guest editor on multiple journals and several *IEEE Communications Magazine* issues. He has co-chaired the Frontiers in Service conference and appeared as keynote speaker at several IEEE and industry conferences.

At Cisco, Ammar is the founding chair of Cisco Services Research and the Cisco Services Patent Council. He received the Cisco Chairman's Choice Award for IoT Excellent Innovation and Execution.

He received B.S. and M.S. degrees in electrical engineering from the University of Illinois at Urbana and a Ph.D. in electrical engineering from Washington University in St. Louis, Missouri, where he received the Outstanding Graduate Student Award in Telecommunications.

Nidhi Kao is a Data Scientist at Cisco Systems who develops advanced analytic solutions for Cisco Advanced Services. She received a B.S. in biochemistry from North Carolina State University and an M.B.A. from the University of North Carolina Kenan Flagler Business School. Prior to working at Cisco Systems, she held analytic chemist and research positions in industry and nonprofit laboratories.

Dedications

This book is dedicated to my wife, Veronica, and my children, Lexy, Trevor, and Mason. Thank you for making it possible for me to follow my passions through your unending support.

Acknowledgments

I would like to thank my manager, Ulf Vinneras, for supporting my efforts toward writing this book and creating an innovative culture where Cisco Services incubation teams can thrive and grow.

To that end, thanks go out to all the people in these incubation teams in Cisco Services for their constant sharing of ideas and perspectives. Your insightful questions, challenges, and solutions have led me to work in interesting roles that make me look forward to coming to work every day. This includes the people who are tasked with incubation, as well as the people from the field who do it because they want to make Cisco better for both employees and customers.

Thank you, Nidhi Kao and Ammar Rayes, for your technical expertise and your time spent reviewing this book. I value your expertise and appreciate your time. Your recommendations and guidance were spot-on for improving the book.

Finally, thanks to the Pearson team for helping me make this career goal a reality. There are many areas of publishing that were new to me, and you made the process and the experience very easy and enjoyable.

Contents at a Glance

Chapter 1	Getting Started with Analytics	1
Chapter 2	Approaches for Analytics and Data Science	13
Chapter 3	Understanding Networking Data Sources	35
Chapter 4	Accessing Data from Network Components	55
Chapter 5	Mental Models and Cognitive Bias	97
Chapter 6	Innovative Thinking Techniques	127
Chapter 7	Analytics Use Cases and the Intuition Behind Them	147
Chapter 8	Analytics Algorithms and the Intuition Behind Them	217
Chapter 9	Building Analytics Use Cases	273
Chapter 10	Developing Real Use Cases: The Power of Statistics	285
Chapter 11	Developing Real Use Cases: Network Infrastructure Analytics	323
Chapter 12	Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry	355
Chapter 13	Developing Real Use Cases: Data Plane Analytics	389
Chapter 14	Cisco Analytics	425
Chapter 15	Book Summary	435
Appendix A	Function for Parsing Packets from pcap Files	443
	Index	445

Contents

	Foreword	xvii
	Introduction: Your future is in your hands!	xviii
Chapter 1	Getting Started with Analytics	1
	What This Chapter Covers	1
	Data: You as the SME	2
	Use-Case Development with Bias and Mental Models	2
	Data Science: Algorithms and Their Purposes	3
	What This Book Does <i>Not</i> Cover	4
	Building a Big Data Architecture	4
	Microservices Architectures and Open Source Software	5
	R Versus Python Versus SAS Versus Stata	6
	Databases and Data Storage	6
	Cisco Products in Detail	6
	Analytics and Literary Perspectives	7
	Analytics Maturity	7
	Knowledge Management	8
	Gartner Analytics	8
	Strategic Thinking	9
	Striving for “Up and to the Right”	9
	Moving Your Perspective	10
	Hot Topics in the Literature	11
	Summary	12
Chapter 2	Approaches for Analytics and Data Science	13
	Model Building and Model Deployment	14
	Analytics Methodology and Approach	15
	Common Approach Walkthrough	16
	Distinction Between the Use Case and the Solution	18
	Logical Models for Data Science and Data	19
	Analytics as an Overlay	20
	Analytics Infrastructure Model	22
	Summary	33
Chapter 3	Understanding Networking Data Sources	35
	Planes of Operation on IT Networks	36
	Review of the Planes	40

	Data and the Planes of Operation	42
	Planes Data Examples	44
	A Wider Rabbit Hole	49
	A Deeper Rabbit Hole	51
	Summary	53
Chapter 4	Accessing Data from Network Components	55
	Methods of Networking Data Access	55
	Pull Data Availability	57
	Push Data Availability	61
	Control Plane Data	67
	Data Plane Traffic Capture	68
	Packet Data	70
	Other Data Access Methods	74
	Data Types and Measurement Considerations	76
	Numbers and Text	77
	Data Structure	82
	Data Manipulation	84
	Other Data Considerations	87
	External Data for Context	89
	Data Transport Methods	89
	Transport Considerations for Network Data Sources	90
	Summary	96
Chapter 5	Mental Models and Cognitive Bias	97
	Changing How You Think	98
	Domain Expertise, Mental Models, and Intuition	99
	Mental Models	99
	Daniel Kahneman's System 1 and System 2	102
	Intuition	103
	Opening Your Mind to Cognitive Bias	104
	Changing Perspective, Using Bias for Good	105
	Your Bias and Your Solutions	106
	How You Think: Anchoring, Focalism, Narrative Fallacy, Framing, and Priming	107
	How Others Think: Mirroring	110
	What Just Happened? Availability, Recency, Correlation, Clustering, and Illusion of Truth	111

	Enter the Boss: HIPPO and Authority Bias	113
	What You Know: Confirmation, Expectation, Ambiguity, Context, and Frequency Illusion	114
	What You Don't Know: Base Rates, Small Numbers, Group Attribution, and Survivorship	117
	Your Skills and Expertise: Curse of Knowledge, Group Bias, and Dunning-Kruger	119
	We Don't Need a New System: IKEA, Not Invented Here, Pro-Innovation, Endowment, Status Quo, Sunk Cost, Zero Price, and Empathy	121
	I Knew It Would Happen: Hindsight, Halo Effect, and Outcome Bias	123
	Summary	124
Chapter 6	Innovative Thinking Techniques	127
	Acting Like an Innovator and Mindfulness	128
	Innovation Tips and Techniques	129
	Developing Analytics for Your Company	140
	Defocusing, Breaking Anchors, and Unpriming	140
	Lean Thinking	142
	Cognitive Trickery	143
	Quick Innovation Wins	143
	Summary	144
Chapter 7	Analytics Use Cases and the Intuition Behind Them	147
	Analytics Definitions	150
	How to Use the Information from This Chapter	151
	Priming and Framing Effects	151
	Analytics Rube Goldberg Machines	151
	Popular Analytics Use Cases	152
	Machine Learning and Statistics Use Cases	153
	Common IT Analytics Use Cases	170
	Broadly Applicable Use Cases	199
	Some Final Notes on Use Cases	214
	Summary	214
Chapter 8	Analytics Algorithms and the Intuition Behind Them	217
	About the Algorithms	217
	Algorithms and Assumptions	218
	Additional Background	219

Data and Statistics	221
Statistics	221
Correlation	224
Longitudinal Data	225
ANOVA	227
Probability	228
Bayes' Theorem	228
Feature Selection	230
Data-Encoding Methods	232
Dimensionality Reduction	233
Unsupervised Learning	234
Clustering	234
Association Rules	240
Sequential Pattern Mining	243
Collaborative Filtering	244
Supervised Learning	246
Regression Analysis	246
Classification Algorithms	248
Decision Trees	249
Random Forest	250
Gradient Boosting Methods	251
Neural Networks	252
Support Vector Machines	258
Time Series Analysis	259
Text and Document Analysis	262
Natural Language Processing (NLP)	262
Information Retrieval	263
Topic Modeling	265
Sentiment Analysis	266
Other Analytics Concepts	267
Artificial Intelligence	267
Confusion Matrix and Contingency Tables	267
Cumulative Gains and Lift	269
Simulation	271
Summary	271

Chapter 9 Building Analytics Use Cases 273

- Designing Your Analytics Solutions 274
- Using the Analytics Infrastructure Model 275
- About the Upcoming Use Cases 276
 - The Data 276
 - The Data Science 278
 - The Code 280
- Operationalizing Solutions as Use Cases 281
 - Understanding and Designing Workflows 282
- Tips for Setting Up an Environment to Do Your Own Analysis 282
- Summary 284

Chapter 10 Developing Real Use Cases: The Power of Statistics 285

- Loading and Exploring Data 286
- Base Rate Statistics for Platform Crashes 288
- Base Rate Statistics for Software Crashes 299
- ANOVA 305
- Data Transformation 310
 - Tests for Normality 311
 - Examining Variance 313
- Statistical Anomaly Detection 318
- Summary 321

Chapter 11 Developing Real Use Cases: Network Infrastructure Analytics 323

- Human DNA and Fingerprinting 324
- Building Search Capability 325
 - Loading Data and Setting Up the Environment 325
 - Encoding Data for Algorithmic Use 328
 - Search Challenges and Solutions 331
- Other Uses of Encoded Data 336
- Dimensionality Reduction 337
- Data Visualization 340
- K-Means Clustering 344
- Machine Learning Guided Troubleshooting 350
- Summary 353

Chapter 12 Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry 355

- Data for This Chapter 356
- OSPF Routing Protocols 357
- Non-Machine Learning Log Analysis Using pandas 357
 - Noise Reduction 360
 - Finding the Hotspots 362
- Machine Learning–Based Log Evaluation 366
 - Data Visualization 367
 - Cleaning and Encoding Data 369
 - Clustering 373
 - More Data Visualization 375
 - Transaction Analysis 379
- Task List 386
- Summary 387

Chapter 13 Developing Real Use Cases: Data Plane Analytics 389

- The Data 390
- SME Analysis 394
- SME Port Clustering 407
- Machine Learning: Creating Full Port Profiles 413
- Machine Learning: Creating Source Port Profiles 419
- Asset Discovery 422
- Investigation Task List 423
- Summary 424

Chapter 14 Cisco Analytics 425

- Architecture and Advisory Services for Analytics 426
- Stealthwatch 427
- Digital Network Architecture (DNA) 428
- AppDynamics 428
- Tetration 430
- Crosswork Automation 431
- IoT Analytics 432
- Analytics Platforms and Partnerships 433
- Cisco Open Source Platform 433
- Summary 434

Chapter 15 Book Summary 435

Analytics Introduction and Methodology 436

All About Networking Data 438

Using Bias and Innovation to Discover Solutions 439

Analytics Use Cases and Algorithms 439

Building Real Analytics Use Cases 440

Cisco Services and Solutions 442

In Closing 442

Appendix A Function for Parsing Packets from pcap Files 443

Index 445

Reader Services

Register your copy at www.ciscopress.com/title/ISBN for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to www.ciscopress.com/register and log in or create an account.* Enter the product ISBN 9781587145131 and click Submit. When the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us to receive exclusive discounts on future editions of this product.

Icons Used in This Book



Cloud-line



Laptop



Router



Relational
Database

Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

Foreword

What's the future of network engineers? This is a question haunting many of us. In the past, it was somewhat easy; study for your networking certification, have the CCIE or CCDE as the ultimate goal, and your future was secured.

In my job as a General Manager within the Cisco Professional Services organization, working with Fortune 1000 clients from around the world, I meet a lot of people with opinions in this matter, with views ranging from “we just need software programmers in the future” to “data scientist is the way to go as we will automate everything.” Is either of these views correct?

My simple answer to this is, “no,” the long answer is a little more complicated.

The changes in the networking industry are to a large extent the same as the automotive industry; today most cars are computerized. Imagine though, if a car was built by people that only knew software programming, and didn't know anything about the car design, the engine, or security. The “architect” of a car needs to be an in-depth expert on car design, and at the same time know enough about software capabilities, and what can be achieved, in a way that still keeps the “soul” of the car and enhances the overall result.

When it comes to the future of networking, it is very much the same. If we replaced skilled network engineers with data science engineers, the result would be mediocre. At the same time, there is no doubt that the future of networking will be built on data science.

In my view, the ideal structure of any IT team is a core of very knowledgeable network engineers, working very closely together with skilled data scientists. The network engineers that take the time to learn the basics of data science, and start to expand into that area will automatically be the bridge to the data science, and these engineers will soon become the most critical asset in that IT department.

The author of this book, John Garrett, is a true example of someone that has made this journey. With many years of experience working with the largest Cisco clients around the world, as one of our more senior network and data center technical leads, John saw the movement of data science approaching, and decided to invest himself in learning this new discipline. I would say he did not only learn it but instead mastered the art.

In this book, John helps the reader along the journey of learning data analytics in a very practical and applied way, providing the tools to almost immediately provide value to your organization.

At the end of the day, career progress is very linked to providing unique value. If you have decided to invest in yourself, and build data science skills on top of your telecommunication, datacenter, security, or IT knowledge, this book is the perfect start.

I would argue that John is a proof point to this matter, moving from a tech lead consultant to now being part of a small core team focusing on innovation to create the future of professional services from Cisco. A confirmation of this is also the number of patent submissions that John has pending in the area, as networking skills combined with data science opened up entirely new avenues of capabilities and solutions.

By Ulf Vinneras, Cisco General Manager Customer Experience/Cross Architecture

Introduction: Your future is in your hands!

Analytics and data science are everywhere. Everything today is connected by networks. In the past networking and data science were distinct career paths, but this is no longer the case. Network and information technology (IT) specialists can benefit from understanding analytics, and data scientists can benefit from understanding how computer networks operate and produce data. People in both roles are responsible for building analytics solutions and use cases that improve the business.

This book provides the following:

- An introduction to data science methodologies and algorithms for network and IT professionals
- An understanding of computer network data that is available from these networks for data scientists
- Techniques for uncovering innovative use cases that combine the data science algorithms with network data
- Hands-on use-case development in Python and deep exploration of how to combine the networking data and data science techniques to find meaningful insights

After reading this book, data scientists will experience more success interacting with IT networking experts, and IT networking experts will be able to aid in developing complete analytics solutions. Experts from either area will learn how to develop networking use cases independently.

My Story

I am a network engineer by trade. Prior to learning anything about analytics, I was an engineer working in data networking. Thanks to my many years of experience, I could design most network architectures that used any electronics to move any kind of data—business critical or not—in support of world-class applications. I thought I knew everything I needed to know about networking.

Then digital transformation happened. The software revolution happened. Everything went software defined. Everything is “virtual” and “containerized” now. Analytics is everywhere. With all these changes, I found that I didn’t know as much as I once thought I did.

If this sounds like your story, then you have enough experience to realize that you need to understand the next big thing if you want to remain relevant in a networking-related role—and analytics applied in your networking domain of expertise is the next big thing for you. If yours is like many organizations today, you have tons of data, and you have analytics tools and software to dive into it, but you just do not really know what to do with it. How can your skills be relevant here? How do you make the connection from these buckets, pockets, and piles of data to solving problems for your company? How

can you develop use cases that solve both business and technical problems? Which use cases provide some real value, and which ones are a waste of your time?

Looking for that next big thing was exactly the situation I found myself in about 10 years ago. I was experienced when it came to network design. I was a 5 year CCIE, and I had transitioned my skill set from campus design to wireless to the data center. I was working in one of the forward-looking areas of Cisco Services, Cisco Advanced Services. One of our many charters was “proactive customer support,” with a goal of helping customers avoid costly outages and downtime by preventing problems from happening in the first place. While it was not called *analytics* back then, the work done by Cisco Advanced Services could fall into a bucket known today as *prescriptive analytics*.

If you are an engineer looking for that next step in your career, many of my experiences will resonate with you. Many years ago, I was a senior technical practitioner deciding what was next for developing my skill set. My son was taking Cisco networking classes in high school, and the writing was on the wall that being only a network engineer was not going to be a viable alternative in the long term. I needed to level up my skills in order to maintain a senior-level position in a networking-related field, or I was looking at a role change or a career change in the future.

Why analytics? I was learning through my many customer interactions that we needed to do more with the data and expertise that we had in Cisco Services. The domain of coverage in networking was small enough back then that you could identify where things were “just not right” based on experience and intuition. At Cisco, we know how to use our collected data, our knowledge about data on existing systems, and our intuition to develop “mental models” that we regularly apply to our customer network environments.

What are mental models? Captain Sully on US Airways flight 1549 used mental models when he made an emergency landing on the Hudson River in 2009. Given all of the airplane telemetry data, Captain Sully knew best what he needed to do in order to land the plane safely and protect the lives of hundreds of passengers. Like experienced airplane pilots, experienced network engineers like you know how to avoid catastrophic failures. Mental models are powerful, and in this book, I tell you how to use mental models and innovation techniques to develop insightful analytics use cases for the networking domain.

The Services teams at Cisco had excellent collection and reporting. Expert analysis in the middle was our secret sauce. In many cases, the anonymized data from these systems became feeds to our internal tools that we developed as “digital implementations” of our mental models. We built awesome collection mechanisms, data repositories, proprietary rule-matching systems, machine reasoning systems, and automated reporting that we could use to summarize all the data in our findings for Cisco Services customers. We were finding insights but not actively looking for them using analytics and machine learning.

My primary interest as a futurist thinker was seeking to understand what was coming next for Cisco Advanced Services and myself. What was the “next big thing” for which we needed to be prepared? In this pursuit, I explored a wide array of new technology areas over the course of 10 years. I spent some years learning and designing VMware,

OpenStack, network functions virtualization (NFV), and the associated virtual network functions (VNFs) solutions on top of OpenStack. I then pivoted to analytics and applied those concepts to my virtualization knowledge area.

After several years working on this cutting edge of virtualized software infrastructure design and analytics, I learned that whether the infrastructure is physical or virtual, whether the applications are local or in the cloud, the importance of being able to find insights within the data that we get from our networking environments is critical to the success of these environments. I also learned that the growth of data science and the availability of computer resources to munge through the data make analytics and data science very attainable for any networking professional who wishes to pivot in this direction.

Given this insight, I spent 3 years of time outside work, including many evenings, weekends, and all of my available vacation time in order to earn a master's degree in predictive analytics from Northwestern University. Around that same time I began reading (or listening to) hundreds of books, articles, and papers about analytics topics. I also consumed interesting writings about algorithms, data science, innovation, innovative techniques, brain chemistry, bias, and other topics related to turning data into value by using creative thinking techniques. You are an engineer, so you can associate this to learning that next new platform, software, or architecture. You go all in.

Another driver for me was that I am work centered, driven to succeed, and competitive by nature. Maybe you are, too. My customers who had purchased Cisco services were challenging us to do better. It was no longer good enough to say that everything is connected, traffic is moving just fine across your network, and if there is a problem, the network protocols will heal themselves. Our customers wanted more than that.

Cisco Advanced Services customers are highly skilled, and they wanted more than simple reporting. They wanted visibility and insights across many domains. My customers wanted data, and they wanted dashboards that shared data with them so they could determine what was wrong on their own. One customer (we will call him Dave because that was his name) wanted to be able to use his own algorithms, his own machines, and his own people to determine what was happening at the lower levels of his infrastructure. He wanted to correlate this network data with his applications and his business metrics. For me, as a very senior network and data center engineer, I felt like I was not getting the job done. I could not do the analytics. I did not have a solution that I could propose for his purpose. There was a new space in networking that I had not yet conquered. Dave wanted actionable intelligence derived from the data that he was providing to Cisco. Dave wanted real analytics insights. Challenge accepted.

That was the start of my journey into analytics and into making the transition from being a network engineer to being a data scientist with enough ability to bridge the gap between IT networking engineers and those mathematical wizards who do the hard-core data science. This book is a knowledge share of what I have learned over the past years as I have transitioned from being an enterprise-focused campus, WAN, and data center networking engineer to being a learning data scientist. I realized that it was not necessary to get to the Ph.D. level to use data science and predictive analytics. For my transition,

I wanted to be someone who can use enough data science principles to find use cases in the wild and apply them to common IT networking problems to find useful, relevant, and actionable insights for my customers.

I hope you enjoy reading about what I have learned on this journey as much as I have enjoyed learning it. I am still working at it, so you will get the very latest. I hope that my learning and experiences in data, data science, innovation, and analytics use cases can help you in your career.

How This Book Is Organized

Chapter 1, “Getting Started with Analytics,” defines some further details about what is explored in this book, as well as the current analytics landscape in the media. You cannot open your laptop or a social media application on your phone without seeing something related to analytics.

Chapter 2, “Approaches for Analytics and Data Science,” explores methodologies and approaches that will help you find success as a data scientist in your area of expertise. The simple models and diagrams that I have developed for internal Cisco trainings can help with your own solution framing activities.

Chapter 3, “Understanding Networking Data Sources,” begins by looking at network data and the planes of operation in networks that source this data. Virtualized solutions such as OpenStack and network functions virtualization (NFV) create additional complexities with sourcing data for analysis. Most network devices can perform multiple functions with the same hardware. This chapter will help you understand how they all fit together so you can get the right data for your solutions.

Chapter 4, “Accessing Data from Network Components,” introduces networking data details. Networking environments produce many different types of data, and there are multiple ways to get at it. This chapter provides overviews of the most common data access methods in networking. You cannot be a data scientist without data! If you are a seasoned networking engineer, you may only need to skim this chapter.

Chapter 5, “Mental Models and Cognitive Bias,” shifts gears toward innovation by spending time in the area of mental models, cognitive science, and bias. I am not a psychology expert or an authority in this space, but in this chapter I share common biases that you may experience in yourself, your users, and your stakeholders. This cognitive science is where things diverge from a standard networking book—but in a fascinating way. Understanding your audience is key to building successful use cases for them.

Chapter 6, “Innovative Thinking Techniques,” introduces innovative techniques and interesting tricks that I have used to uncover use cases in my role with Cisco. Understanding bias from Chapter 5 coupled with innovation techniques from this chapter will prepare you to maximize the benefit of the use cases and algorithms you learn in the upcoming chapters.

Chapter 7, “Analytics Use Cases and the Intuition Behind Them,” has you use your new knowledge of innovation to walk through analytics use cases across many industries. I have learned that combining the understanding of data with new and creative—and sometimes biased—thinking results in new understanding and new perspective.

Chapter 8, “Analytics Algorithms and the Intuition Behind Them,” walks through many common industry algorithms from the use cases in Chapter 7 and examines the intuition behind them. Whereas Chapter 7 looks at use cases from a top-down perspective, this chapter looks at algorithms to give you an inside-out view. If you know the problems you want to solve, this is your toolbox.

Chapter 9, “Building Analytics Use Cases,” brings back the models and methodologies from Chapter 2 and reviews how to turn your newfound ideas and algorithms into solutions. The use cases and data for the next four chapters are outlined here.

Chapter 10, “Developing Real Use Cases: The Power of Statistics,” moves from the abstract to the concrete and explores some real Cisco Services use cases built around statistics. There is still a very powerful role for statistics in our fancy data science world.

Chapter 11, “Developing Real Use Cases: Network Infrastructure Analytics,” looks at actual solutions that have been built using the feature information about your network infrastructure. A detailed look at Cisco Advanced Services fingerprinting, and other infrastructure-related capabilities is available here.

Chapter 12, “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry,” shows how to build solutions that use network event telemetry data. The popularity of pushing data from devices is growing, and you can build use cases by using such data. Familiar algorithms from previous chapters are combined with new data in this chapter to provide new insight.

Chapter 13, “Developing Real Use Cases: Data Plane Analytics,” introduces solutions built for making sense of data plane traffic. This involves analysis of the packets flowing across your network devices. Familiar algorithms are used again to show how you can use the same analytics algorithms in many ways on many different types of data to find different insights.

Chapter 14, “Cisco Analytics,” runs through major Cisco product highlights in the analytics space. Any of these products can function as data collectors, sources, or engines, and they can provide you with additional analytics and visualization capabilities to use for solutions that extend the capabilities and base offerings of these platforms. Think of them as “starter kits” that help you get a working product in place that you can build on in the future.

Chapter 15, “Book Summary,” closes the book by providing a complete wrap-up of what I hope you learned as you read this book.

Credits

Stephen R. Covey, *The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change*, 2004, Simon and Schuster.

ITU Annual Regional Human Capacity Building Workshop for Sub-Saharan Countries in Africa Mauritius, 28–30 June 2017

Empirical Model-Building and Response Surfaces, 1987, George box, John Wiley.

Predictably Irrational: The Hidden Forces that Shape Our Decisions, Dan Ariely, HarperCollins.

Thinking, Fast and Slow, Daniel Kahneman, Macmillan Publishers

Abraham Wald

Thinking, Fast and Slow, Daniel Kahneman, Macmillan Publishers

Thinking, Fast and Slow, Daniel Kahneman, Macmillan Publishers

Thinking, Fast and Slow, Daniel Kahneman, Macmillan Publishers

Charles Duhigg

De, B. E. (1985). Six thinking hats. Boston: Little, Browne and Company.

Henry Ford

Ries, E. (2011). *The lean startup: How constant innovation to creates radically successful businesses*. Penguin Books

The Post-Algorithmic Era Has Arrived By Bill Franks, Dec 14, 2017.

Figure Credits

Figure 8-13 Scikit-learn

Figure 8-32 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 8-33 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 8-34 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-07 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-08 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-18 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-22 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-23 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-24 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-26 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-27 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-30 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-31 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Figure 10-32 Screenshot of Jupyter Notebook © 2018 Project Jupyter

- Figure 10-34 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-37 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-38 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-39 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-40 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-47 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-49 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-51 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-53 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-54 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-61 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 10-62 Screenshot of Excel © Microsoft
- Figure 11-22 Screenshot of Business Critical Insights © 2018 Cisco Systems, Inc.
- Figure 11-32 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 11-34 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 11-38 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 11-41 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 11-51 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-10 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-12 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-13 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-14 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-15 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 13-35 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-03 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-04 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-05 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-07 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-08 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-09 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-10 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-11 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-12 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-15 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-18 Screenshot of Jupyter Notebook © 2018 Project Jupyter
- Figure 12-42 Screenshot of Jupyter Notebook © 2018 Project Jupyter

Getting Started with Analytics

Why should you care about analytics? Because networking—like every other industry—is undergoing transformation. Every industry needs to fill data scientist roles. Anyone who is already in an industry and learns data science is going to have a leg up because he or she already has industry subject matter expert (SME) skills, which will help in recognizing where analytics can provide the most benefit.

Data science is expected to be one of the hottest job areas in the near future. It is also one of the better-paying job areas. With a few online searches, you can spend hours reading about the skills gap, low candidate availability, and high pay for these jobs. If you have industry SME knowledge, you are instantly more valuable in the IT industry if you can help your company further the analytics journey. Your unique expertise combined with data science skills and your ability to find new solutions will set you apart.

This book is about uncovering use cases and providing you with baseline knowledge of networking data, algorithms, biases, and innovative thinking techniques. This will get you started on transforming yourself. You will not learn everything you need to know in one book, but this book will help you understand the analytics big picture, from the data to the use cases. Building models is one thing; building them into productive tools with good workflows is another thing; getting people to use them to support the business is yet another. You will learn ways to identify what is important to the stakeholders who use your analytics solutions to solve their problems. You will learn how to design and build these use cases.

What This Chapter Covers

Analytics discovery can be boiled down to three main themes, as shown in Figure 1-1. Understanding these themes is a critical success factor for developing effective use cases.

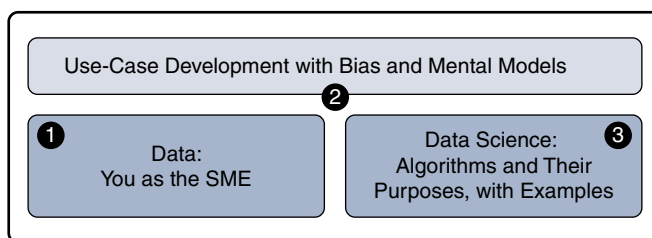


Figure 1-1 *Three Major Themes in This Book*

Data: You as the SME

You, as an SME, will spend the majority of your time working with data. Understanding and using networking data in detail is a critical success factor. Your claim to fame here is being an expert in the networking space, so you need to own that part. Internet surveys show that 80% or more of data scientists’ time is spent collecting, cleaning, and preparing data for analysis. I can confirm this from my own experience, and I have therefore devoted a few chapters of this book to helping you develop a deeper understanding of IT networking data and building data pipelines. This area of data prep is referred to as “feature engineering” because you need to use your knowledge and experience to translate the data from your world into something that can be used by machine learning algorithms.

I want to make a very important distinction about data sets and streaming data here, early in this book. *Building* analytics models and *deploying* analytics models can be two very different things. Many people build analytics models using batches of data that have been engineered to fit specific algorithms. When it comes time to deploy models that act on live data, however, you must deploy these models on actual streaming data feeds coming from your environment. Chapter 2, “Approaches for Analytics and Data Science,” provides a useful new model and methodology to make this deployment easier to understand and implement. Even online examples of data science mostly use captured data sets to show how to build models but lack actual deployment instructions. You will find the methodology provided in this book very valuable for building solutions that you can explain to your stakeholders and implement in production.

Use-Case Development with Bias and Mental Models

The second theme of this book is the ability to find analytics use cases that fit your data and are of interest to your company. Stakeholders often ask the questions “What problem are you going to solve?” and “If we give you this data and you get some cool insights, what can we do about them?” If your answers to these questions are “none” and “nothing,” then you are looking at the wrong use cases.

This second theme involves some creative thinking inside and outside your own mental models, thinking outside the box, and seeing many different perspectives by using bias

as a tool. This area, which can be thought of as “turning on the innovator,” is fascinating and ever growing. Once you master some skills in this space, you will be more effective at identifying potential use cases. Then your life becomes an exercise in prioritizing your time to focus on the most interesting use cases only. This book defines many techniques for fostering innovative thinking so you can create some innovative use cases in your own area of expertise.

Data Science: Algorithms and Their Purposes

The third theme of this book is the intuition behind some major analytics use cases and algorithms. As you get better at uncovering use cases, you will understand how the algorithms support key findings or insights. This understanding allows you to combine algorithms with your mental models and data understanding to create new and insightful use cases in your own space, as well as adjacent and sometimes opposing spaces.

You do not typically find these themes of networking expert, data expert, and data scientist in the same job roles. Take this as innovation tip number one: Force yourself to look at things from other perspectives and step out of your comfort zone. I still spend many hours a week of my own time learning and trying to gain new perspectives. Chapter 5, “Mental Models and Cognitive Bias,” examines these techniques. The purpose of this book is to help expand your thinking about where and how to apply analytics in your job role by taking a different perspective on these main themes. Chapter 7, “Analytics Use Cases and the Intuition Behind Them,” explores the details of common industry uses of analytics. You can mix and match them with your own knowledge and bias to broaden your thinking for innovation purposes.

I chose networking use cases for this book because networking has been my background for many years. My customer-facing experience makes me an SME in this space, and I can easily relate the areas of networking and data science for you. I repeat that the most valuable analytics use cases are found when you combine data science with your own domain expertise (which SMEs have) in order to find the insights that are most relevant in your domain. However, analytics use cases are everywhere. Throughout the book, a combination of popular innovation-fostering techniques are used to open your eyes, and your mind, to be able to recognize use cases when you see them.

After reading this book, you will have analytics skills related to different job roles, and you will be ready to engage in conversation on any of them. One book, however, is not going to make you an expert. As shown in Figure 1-2, this book prepares you with the baseline knowledge you need to take the next step in a number of areas, as your personal or professional interest dictates. The depth that you choose will vary depending on your interest. You will learn enough in this book to understand your options for next steps.

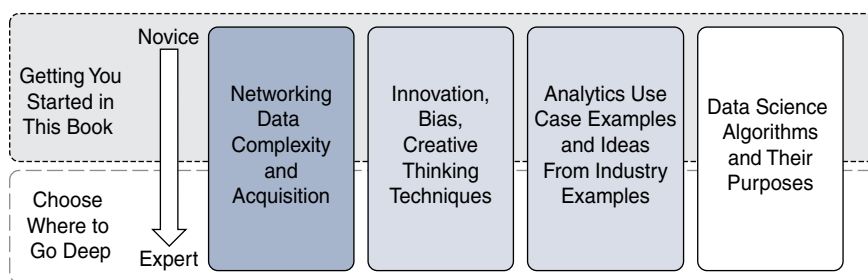


Figure 1-2 *Major Coverage Areas in This Book*

What This Book Does *Not* Cover

Data science and analytics is a very hot area right now. At the time of this writing, most “hot new jobs” predictions have data science and data engineering among the top five jobs for the next decade. The goal of this book is to get you started on your own analytics journey by filling some gaps in the Internet literature for you. However, a secondary goal of this book is to avoid getting so bogged down in analytics details and complex algorithms that you tune out.

This book covers a broad spectrum of useful material, going just deep enough to give you a starting point. Determining where to drill deep versus stay high-level can be difficult, but this book provides balanced material to help you make these choices. The first nine chapters of this book provide you with enough guidance to understand a solution architecture on a topic, and if any part of the solution is new to you, you will need to do some research to find the final design details of your solution.

Building a Big Data Architecture

An overwhelming number of big data, data platform, data warehouse, and data storage options are available today, but this book does not go into building those architectures. Components and functions provided in these areas, such as databases and message busses, may be referenced in the context of solutions. As shown in Figure 1-3, these components and functions provide a centralized engine for operationalizing analytics solutions.

These data processing resources are central to almost all analytics solutions. Suggestions for how to build and maintain them are widely documented, and these resources are available in the cloud for very reasonable cost. While it is interesting to know how to build these architectures, for a new analytics professional, it is more important to know how to use them. If you are new to analytics, learning data platform details will slow down your learning in the more important area of analytics algorithms and finding the use cases.

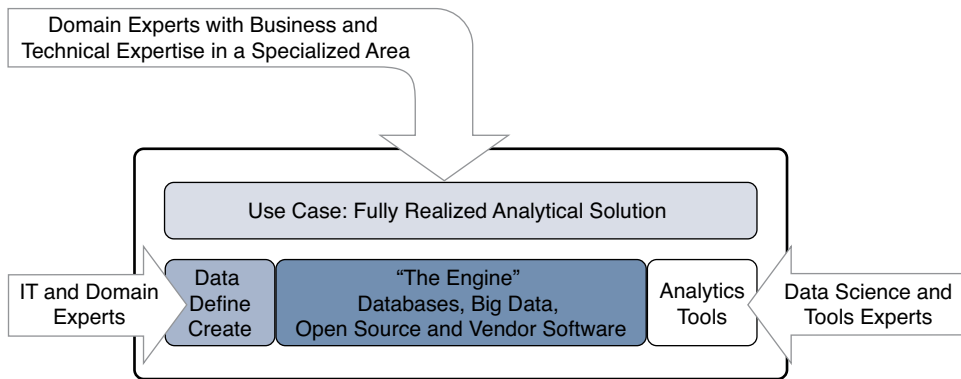


Figure 1-3 *Scope of Coverage for This Book*

Methods and use cases for the networking domain are lacking. In addition, it is not easy to find innovative ways to develop interesting and useful data science use cases across disparate domains of expertise. While big data platforms/systems are a necessary component of any deployed solution, they are somewhat commoditized and easy to acquire, and the trend in this direction continues.

Microservices Architectures and Open Source Software

Fully built and deployed analytics solutions often include components reflecting some mix of vendor software and open source software. You build these architectures using servers, virtual machines, containers, and application programming interface (API) reachable functions, all stitched together into a working pipeline for each data source, as illustrated in Figure 1-4. A container is like a very lightweight virtual machine, and microservices are even lighter: A microservice is usually a container with a single purpose. These architectures are built on demand, as needed.

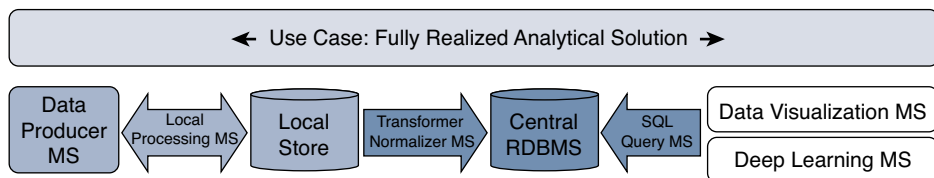


Figure 1-4 *Microservices Architecture Example*

Based on the trends in analytics, most analytics pipelines are expected to be deployed as such systems of microservices in the future (if they are not already). Further, automated systems deploy microservices at scale and on demand. This is a vast field of current

activity, research, and operational spending that is not covered in this book. Popular cloud software such as OpenStack and Kubernetes, along with network functions virtualization (NFV), has proven that this functionality, much like the building of big data platforms, is becoming commoditized as automation technology and industry expertise in this space advance.

R Versus Python Versus SAS Versus Stata

This book does not recommend any particular platform or software. Arguments about which analytics software provides the best advantages for specific kinds of analysis are all over the Internet. This book is more concept focused than code focused, and you can use the language of your choice to implement it. Code examples in this book are in Python. It might be a cool challenge for you to do the same things in your own language of choice. If you learn and understand an algorithm, then the implementation in another language is mainly just syntax (though there are exceptions, as some packages handle things like analytics vector math much better than others). As mentioned earlier, an important distinction is the difference between building a model and deploying a model. It is possible that you will build a model in one language, and your software development team will then deploy it in a different language.

Databases and Data Storage

This book does not cover databases and data storage environments. At the center of most analytics designs, there are usually requirements to store data at some level, either processed or raw, with or without associated schemas for database storage. This core component exists near or within the central engine. Just as with the overall big data architectures, there are many ways to implement database layer functionality, using a myriad of combinations of vendor and open source software. Loads of instruction and research are freely available on the Internet to help you. If you have not done it before, take an hour, find a good site or blog with instructions, and build a database. It is surprisingly simple to spin up a quick database implementation in a Linux environment these days, and storage is generally low cost. You can also use cloud-based resources and storage. The literature surrounding the big data architecture is also very detailed in terms of storage options.

Cisco Products in Detail

Cisco has made massive investments in both building and buying powerful analytics platforms such as Tetration, AppDynamics, and Stealthwatch. This book does not cover such products in detail, and most of them are already covered in depth in other books. However, because these solutions can play parts in an overall analytics strategy, this book covers how the current Cisco analytics solutions fit into the overall analytics picture and provides an overview of the major use cases that these platforms can provide for your environment. (This coverage is about the use cases, however, not instructions for using the products.)

Analytics and Literary Perspectives

No book about analytics would be complete without calling out popular industry terminology and discussion about analytics. Some of the terminology that you will encounter is summarized in Figure 1-5. The rows in this figure show different aspects of data and analytics, and the columns show stages of each aspect.

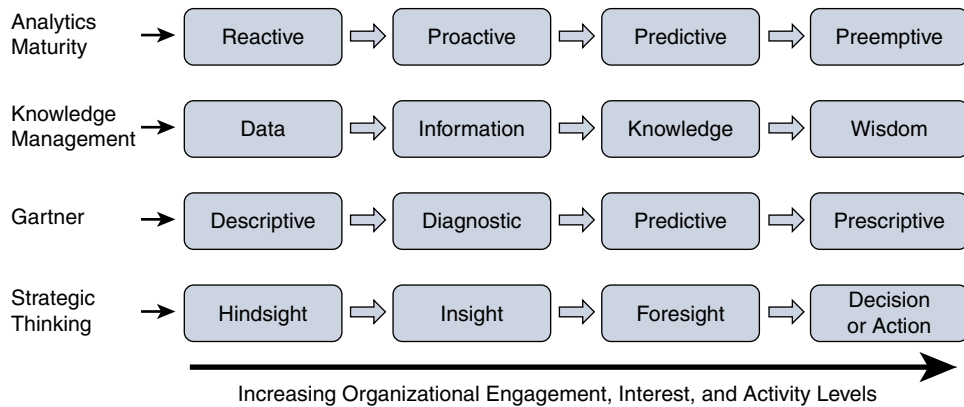


Figure 1-5 *Industry Terminology for Analytics*

Run an Internet search on each of the aspect row headings in Figure 1-5 to dig deeper into the initial purpose and interpretation. How you interpret them should reflect your own needs. These are continuums, and these continuums are valuable in determining the level of “skin in the game” when developing groundbreaking solutions for your environment.

If you see terminology that resonates with you, that is what you should lead with in your company. Start there and grow up or down, right or left. Each of the terms in Figure 1-5 may invoke some level of context bias in you or your audience, or you may experience all of them in different places. Every stage and row has value in itself. Each of these aspects has benefits in a very complete solutions architecture. Let’s quickly go through them.

Analytics Maturity

Analytics maturity in an organization is about how the organization uses its analytics findings. If you look at analytics maturity levels in various environments, you can describe organizational analytics maturity along a scale of reactive to proactive to predictive to preemptive—for each individual solution. As these words indicate, analytics maturity describes the level of maturity of a solution in the attempt to solve a problem with analytics.

For example, reactive maturity when combined with descriptive and diagnostic analytics simply means that you can identify a problem (descriptive) and see the root causes (diagnostic), but you probably go out and fix that problem through manual effort, change

controls, and feet on the street (reactive). If you are at the reactive maturity level, perhaps you see that a network device has consumed all of its memory, and you have identified a memory leak, and you have to schedule an “emergency change” to reboot/upgrade it. This is a common scenario in less mature networking environments. This need to schedule this emergency change and impact schedules of all involved is very much indicative of a reactive maturity level.

Continuing with the same example, if your organization is at the proactive maturity level, you are likely to use analytics (perhaps regression analysis) to proactively go look for the memory leak trend in all your other devices that are similar to this one. Then you can proactively schedule a change during a less expensive timeframe. You can identify places where this might happen using simple trending and heuristics.

At the predictive maturity level, you can use analytics models such as simple extrapolation or regression analysis to determine *when* this device will experience a memory leak. You can then better identify whether it needs to be in this week’s change or next month’s change, or whether you must fix it after-hours today. At this maturity level, models and visualizations show the predictions along with the confidence intervals assigned to memory leak impacts over time.

With preemptive maturity, your analytics models can predict *when* a device will have an issue, and your automated remediation system can automatically schedule the upgrade or reload to fix this known issue. You may or may not get a request to approve this automated work. Obviously, this “self-healing network” is the holy grail of these types of systems.

It is important to keep in mind that you do not need to get to a full preemptive state of maturity for all problems. There generally needs to be an evaluation of the cost of being preemptive versus the risk and impact of *not* being preemptive. Sometimes knowing is good enough. Nobody wants an analytics Rube Goldberg machine.

Knowledge Management

In the knowledge management context, analytics is all about managing the data assets. This involves extracting information from data such that it provides knowledge of what has happened or will happen in the future. When gathered over time, this information turns into knowledge about what is happening. After being seen enough times, this in-context knowledge provides wisdom about how things will behave in the future. Seeking wisdom from data is simply another way to describe insights.

Gartner Analytics

Moving further down the chart, popularized research from Gartner describes analytics in different categories as adjectives. This research first starts with *descriptive analytics*, which describes the state of the current environment, or the state of “what is.” Simple descriptive analytics often gets a bad name as not being “real analytics” because it simply provides data collection and a statement of the current state of the environment. This is

an incorrect assessment, however: Descriptive analytics is a foundational component in moving forward in analytics. If you can look at what is, then you can often determine, given the right expertise, what is wrong with the current state of “what is” and how descriptive analytics contributes to your getting into that state. In other words, descriptive analytics often involves simple charts, graphs, visualizations, or data tables of the current state of the environment that, when placed into the hands of subject matter experts (SME), are used to diagnose problems in the environment.

Where analytics begins to get interesting to many folks is when it moves toward predictive analytics. Say that you know that some particular state of descriptive analytics is a diagnostic indicator pointing toward some problem that you are interested in learning more about. You might then develop analytics systems that automatically identify the particular problem and predict with some level of accuracy that it will happen. This is the simple definition of predictive analytics. It is the “what will happen” part of analytics, which is also the “outcome” of predictive analytics from the earlier part of the maturity continuum. Using the previous example, perhaps you can see that memory in the device is trending upward, and you know the memory capacity of the device, so you can easily predict when there will be a problem. When you know the state and have diagnosed the problem with that state, and when you know how to fix that problem, you can prescribe the remedy for that condition. Gartner aptly describes this final category as *prescriptive analytics*. Let’s compare this to the preemptive maturity: *Preemptive* means that you have the capability to automatically do something based on your analytics findings, whereas *prescriptive* means you actually know what to do.

This continuum of descriptive analytics used for diagnostic analytics to support predictive analytics leads to prescriptive analytics. Prescriptive analytics is used to solve a problem because you know what to do about it. This flow is very intuitive and useful in understanding analytics from different perspectives.

Strategic Thinking

The final continuum on this diagram falls into the realm of strategic thinking, which is possibly the area of analytics most impacted by bias, as discussed in detail later in this book. The main states of hindsight, insight, and foresight map closely to the Gartner categories, and Gartner often uses these terms in the same diagrams. *Hindsight* is knowing what has already happened (sometimes using machine learning stats). *Insight* in this context is knowing what is happening now, based on current models and data trending up to this point in time. As in predictive analytics, *foresight* is knowing what will happen next. Making a decision or taking action based on foresight is simply another way to show that fully actionable items perceived to be coming in the future are actioned.

Striving for “Up and to the Right”

In today’s world, you can summarize any comparison topic into a 2x2 chart. Go out and find some 2x2 chart, and you immediately see that “up and to the right” is usually the best place to be. Look again at Figure 1-5 to uncover the “up and to the right” for

analytics. Cisco seeks to work in this upper-right quadrant, as shown in Figure 1-6. Here is the big secret in one simple sentence: From experience, seek the predictive knowledge that provides the wisdom for you to take preemptive action. Automate that, and you have an awesome service assurance system.

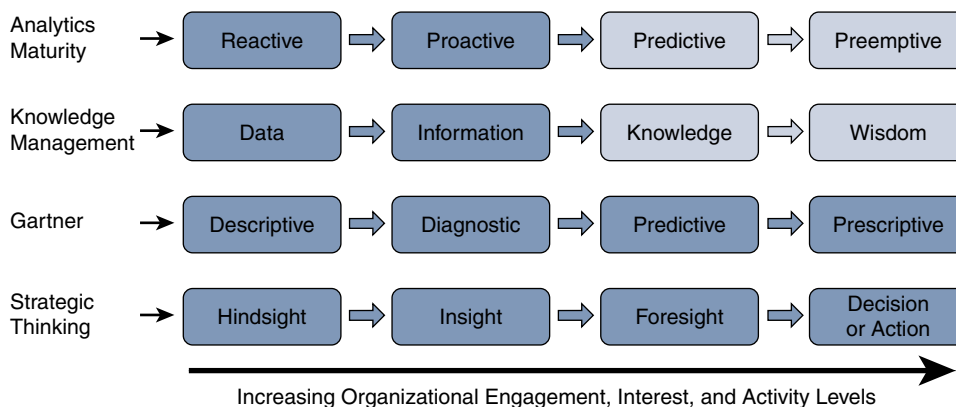


Figure 1-6 *Where You Want to Be with Analytics*

Moving Your Perspective

Depending on background, you will encounter people who prefer one or more of these analytics description areas. Details on each of them are widely available. Once again, the best way forward is to use the area that is familiar to your organization. Today, many companies have basic descriptive and diagnostic analytics systems in place, and they are proactive such that they can address problems in their IT environment before they have much user impact. However, there are still many addressable problems happening while IT staff are spending time implementing these reactive or proactive measures. Building a system that adds predictive capabilities on top of prescriptive analytics with preemptive capabilities that result from automated decision making is the best of all worlds. IT staff can then turn their focus to building smarter, better, and faster people, processes, tools, and infrastructures that bubble up the next case of predictive, prescriptive, and preemptive analytics for their environments. It really is a snowball effect of success.

Stephen Covey, in his book *The Seven Habits of Highly Successful People*, calls this exercise of improving your skills and capabilities “sharpening the saw.” “Sharpening the saw” is simply a metaphor for spending time planning, educating, and preparing yourself for what is coming so that you are more efficient at it when you need to do it. Covey uses an example of cutting down a tree, which takes eight hours with a dull saw. If you take a break from cutting and spend an hour sharpening the saw, the tree cutting takes only a few hours, and you complete the entire task in less than half of the original estimate of eight hours. How is this relevant to you? You can stare at the same networking data for years, or you can take some time to learn some analytics and data science and then go back to that same data and be much more productive with it.

Hot Topics in the Literature

In a book about analytics, it is prudent to share the current trends in the press related to analytics. The following are some general trends related to analytics right now:

- **Neural networks**—Neural networks, described in Chapter 8, “Analytics Algorithms and the Intuition Behind Them,” are very hot, with additions, new layers, and new activation functions. Neural networks are very heavily used in artificial intelligence, reinforcement learning, classification, prediction, anomaly detection, image recognition, and voice recognition.
- **Citizen data scientist**—Compute power is cheap and platforms are widely available to run a data set through black-box algorithms to see what comes out the other end. Sometimes even a blind squirrel finds a nut.
- **Artificial intelligence** and the singularity are hot topics. When will artificial intelligence be able to write itself? When will all jobs be lost to the machines? These are valid concerns as we transition to a knowledge worker society.
- **Automation and intent-based networking**—These areas are growing rapidly. The impact of automation is evident in this book, as not much time is spent on the “how to” of building analytics big data clusters. Automated building of big data solutions is available today and will be widely available and easily accessible in the near future.
- **Computer language translation**—Computer language translation is now more capable than most human translators.
- **Computer image comparison and analysis**—This type of analysis, used in industries such as medical imaging, has surpassed human capability.
- **Voice recognition**—Voice recognition technology is very mature, and many folks are talking to their phones, their vehicles, and assistants such as Siri and Alexa.
- **Open source software**—Open source software is still very popular, although the pendulum may be swinging toward people recognizing that open source software can increase operational costs tremendously and may provide nothing useful (unless you automate it!).

An increasingly hot topic in all of Cisco is full automation and orchestration of software and network repairs, guided by intent. *Orchestration* means applying automation in a defined order. What is *intent*? Given some state of policy that you “intend” your network to be, you can let the analytics determine when you deviate and let your automation go out and bring things back in line with the policy. That is intent-based networking (IBN) in one statement. While IBN is not covered in this book, the principles you learn will allow you to better understand and successfully deploy intent-based networks with full-service assurance layers that rely heavily on analytics.

Service assurance is another hot term in industry. Assuming that you have deployed a service—either physical or virtual, whether a single process or an entire pipeline of physical and virtual things—service assurance as applied to a solution implies that you

will keep that solution operating, abiding by documented service-level agreements (SLAs), by any means necessary, including heavy usage of analytics and automation. Service assurance systems are not covered in detail in this book because they require a fully automated layer to take action in order to be truly preemptive. Entire books are dedicated to building automated solutions. However, it is important to understand how to build the solutions that feed analytics findings into such a system; they are the systems that support the decisions made by the automated tools in the service assurance system.

Summary

This chapter defines the scope of coverage of this book, and the focus of analytics and generating use cases. It also introduces models of analytics maturity so you can see where things fit. You may now be wondering where you will be able to go next after reading this book. Most of the time, only the experts in a given industry take insights and recommended actions and turn them into fully automated self-healing mechanisms. It is up to you to apply the techniques that you learn in this book to your own environment. After reading this book, you can choose to next learn how to set up systems to “do something about it” (preemptive) when you know what do to (wisdom and prescriptive) and have decided that you can automate it (decision or action), as shown in Figure 1-7.

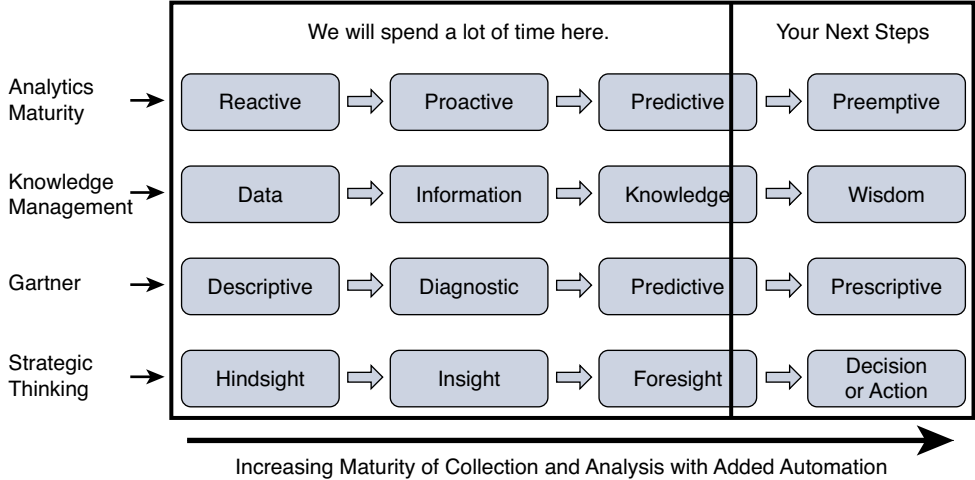


Figure 1-7 Next Steps for You with Analytics

The first step in teaching you to build your analytics skills is getting a usable analytics methodology as a foundation of knowledge for you to build upon as you progress through the chapters of this book. That occurs in the next chapter.

Approaches for Analytics and Data Science

This chapter examines a simple methodology and approach for developing analytics solutions. When I first started analyzing networking data, I used many spreadsheets, and I had a lot of data access, but I did not have a good methodology to approach the problems. You can only sort, filter, pivot, and script so much when working with a single data set in a spreadsheet. You can spend hours, days, or weeks diving into the data, slicing and dicing, pivoting this way and that...only to find that the best you can do is show the biggest and the smallest data points. You end up with no real insights. When you share your findings to glassy-eyed managers, the rows and columns of data are a lot more interesting to you than they are to them. I have learned through experience that you need more.

Analytics solutions look at data to uncover stories about what is happening now or what will be happening in the future. In order to be effective in a data science role, you must step up your storytelling game. You can show the same results in different ways—sometimes many different ways—and to be successful, you must get the audience to see what you are seeing. As you will learn in Chapter 5, “Mental Models and Cognitive Bias,” people have biases that impact how they receive your results, and you need to find a way to make your results relevant to each of them—or at least make your results relevant to the stakeholders who matter.

You have two tasks here. First, you need to find a way to make your findings interesting to nontechnical people. You can make data more interesting to nontechnical people with statistics, top-*n* reporting, visualization, and a good storyline. I always call this the “BI/BA of analytics,” or the simple descriptive analytics. Business intelligence (BI)/business analytics (BA) dashboards are a useful form of data presentation, but they typically rely on the viewer to find insight. This has value and is useful to some extent but generally tops out at cool visualizations that I call “*Sesame Street* analytics.”

If you are from my era, you grew up with the *Sesame Street* PBS show, which had a segment that taught children to recognize differences in images and had the musical tagline “One of these things is not like the others.” Visualizations with anomalies identified in

contrasting colors immediately help the audience see how “one of these things is not like the others,” and you do not need a story if you have shown this properly. People look at your visualization or infographic and just see it.

Your second task is to make the data interesting to the technical people, your new data science friends, your peers. You do this with models and analytics, and your visualizing and storytelling must be at a completely new level. If you present “*Sesame Street* analytics” to a technical audience, you are likely to hear “That’s just visualization; I want to know *why* is it an outlier.” You need to do more—with real algorithms and analytics—to impress this audience. This chapter starts your journey toward impressing both audiences.

Model Building and Model Deployment

As mentioned in Chapter 1, “Getting Started with Analytics,” when it comes to analytics models, people often overlook a very important distinction between *developing and building* and *implementing and deploying* models. The ability for your model to be usable outside your own computer is a critical success factor, and you need to know how to both build and deploy your analytics use cases. It is often the case that you build models centrally then deploy them at the edge of a network or at many edges of corporate or service provider networks. Where do you think the speech recognition models on your mobile phone were built? Where are they ultimately deployed? If your model is going to have impact in your organization, you need to develop workflows that use your model to benefit the business in some tangible way.

Many models are developed or built from batches of test data, perhaps with data from a lab or a big data cluster, built on users’ machines or inside an analytics package of data science algorithms. This data is readily available, cleaned, and standardized, and they have no missing values. Experienced data science people can easily run through a bunch of algorithms to visualize and analyze the data in different ways to glean new and interesting findings. With this captive data, you can sometimes run through hundreds of algorithms with different parameters, treating your model like a black box, and only viewing the results. Sometimes you get very cool-looking results that are relevant. In the eyes of management or people who do not understand the challenges in data science, such development activity looks like the simple layout in Figure 2-1, where data is simply combined with data science to develop a solution. Say hello to your nontechnical audience. This is not a disparaging remark; some people—maybe even most people—prefer to just get to the point, and nothing gets to the point better than results. These people do not care about the details that you needed to learn in order to provide solutions at this level of simplicity.

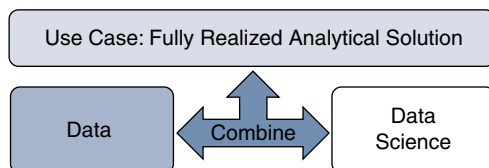


Figure 2-1 Simplified View of Data Science

Once you find a model, you bring in more data to further test and validate that the model's findings are useful. You need to prove beyond any reasonable doubt that the model you have on your laptop shows value. Fantastic. Then what? How can you bring all data across your company to your computer so that you can run it through the model you built?

At some point in the process, you will deploy your analytics to a production system, with real data, meaning that an automated system is set up to run new data, in batches or streaming, against your new model. This often involves working with a development team, whose members may or may not be experts in analytics. In some cases, you do not need to deploy into production at all because the insight is learned, and no further understanding is required. In either case, you then need to use your model against new batches of data to extend the value beyond the data you originally used to build and test it.

Because I am often the one with models on my computer, and I have learned how to deploy those models as part of useful applications, I share my experiences in turning models into useful tools in later chapters of this book, as we go through actual use cases.

Analytics Methodology and Approach

How you approach an analytics problem is one of the factors that determine how successful your solution will be in solving the problem. In the case of analytics problems, you can use two broad approaches, or methodologies, to get to insightful solutions. Depending on your background, you will have some predetermined bias in terms of how you want to approach problems. The ultimate goal is to convert data to value for your company. You get to that value by finding insights that solve technical or business problems. The two broad approaches, shown in Figure 2-2, are the “explore the data” approach, and the “solve the business problem” approach.

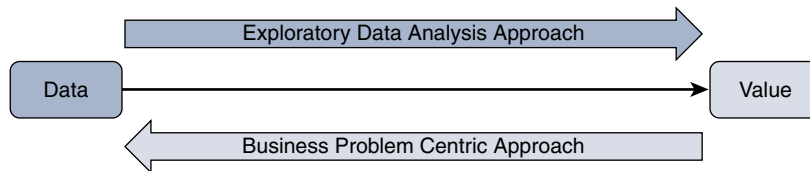


Figure 2-2 *Two Approaches to Developing Analytics Solutions*

These are the two main approaches that I use, and there is literature about many granular, systematic methodologies that support some variation of each of these approaches. Most analytics literature guides you to the problem-centric approach. If you are strongly aware of the data that you have but not sure how to use it to solve problems, you may find yourself starting in the statistically centered exploratory data analysis (EDA) space that is most closely associated with statistician John Tukey. This approach often has some quick wins along the way in finding statistical value in the data rollups and visualizations used to explore the data.

Most domain data experts tend to start with EDA because it helps you understand the data and get the quick wins that allow you to throw a bone to the stakeholders while digging into the more time-consuming part of the analysis. Your stakeholders often have hypotheses (and some biases) related to the data. Early findings from this side often sound like “You can see that issue X is highly correlated with condition Y in the environment; therefore, you should address condition Y to reduce the number of times you see issue X.” Most of my early successes in developing tools and applications for Cisco Advanced Services were absolutely data first and based on statistical findings instead of analytics models. There were no heavy algorithms involved, there was no machine learning, and there was no real data science. Sometimes, statistics are just as effective at telling interesting stories. Figure 2-3 shows how to view these processes as a comparison. There is no right or wrong side on which to start; depending on your analysis goals, either direction or approach is valid. Note that this model includes data acquisition, data transport, data storage, sharing, or streaming, and secure access to that data, all of which are things to consider if the model is to be implemented on a production data flow—or “operation-alized.” The previous, simpler model that shows a simple data and data science combination (refer to Figure 2-1) still applies for exploring a static data set or stream that you can play back and analyze using offline tools.

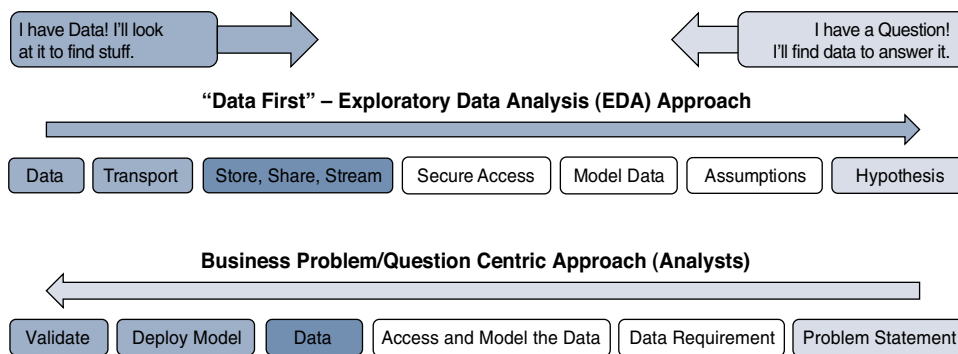


Figure 2-3 *Exploratory Data Versus Problem Approach Comparison*

Common Approach Walkthrough

While many believe that analytics is done only by math PhDs and statisticians, general analysts and industry subject matter experts (SMEs) now commonly use software to explore, predict, and preempt business and technical problems in their areas of expertise. You and other “citizen data scientists” can use a variety of software packages available today to find interesting insights and build useful models. You can start from either side when you understand the validity of both approaches. The important thing to understand is that many of the people you work with may be starting at the other end of the spectrum, and you need to be aware of this as you start sharing your insights with a wider audience. When either audience asks, “What problem does this solve for us?” you can present relevant findings.

Let's begin on the data side. During model *building*, you skip over the transport, store, and secure phases as you grab a batch of useful data, based on your assumptions, and try to test some hypothesis about it. Perhaps through some grouping and clustering of your trouble ticket data, you have seen excessive issues on your network routers with some specific version of software. In this case, you can create an analysis that proves your hypothesis that the problems are indeed related to the version of software that is running on the suspect network routers. For the data first approach, you need to determine the problems you want to solve, and you are also using the data to guide you to what is possible, given your knowledge of the environment.

What do you need in this suspect routers example? Obviously, you must get data about the network routers when they showed the issue, as well as data about the same types of routers that have not had the issue. You need both of these types of information in order to find the underlying factors that may or may not have contributed to the issue you are researching. Finding these factors is a form of inference, as you would like to infer something about all of your routers, based on comparisons of differences in a set of devices that exhibit the issue and a set of devices that do not. You will later use the same analytics model for prediction.

You can commonly skip the “production data” acquisition and transport parts of the model building phase. Although in this case you have a data set to work with for your analysis, consider here how to automate the acquisition of data, how to transport it, and where it will live if you plan to put your model into a fully automated production state so it can notify you of devices in the network that meet these criteria. On the other hand, full production state is not always necessary. Sometimes you can just grab a batch of data and run it against something on your own machine to find insights; this is valid and common. Sometimes you can collect enough data about a problem to solve that problem, and you can gain insight without having to implement a full production system.

Starting at the other end of this spectrum, a common analyst approach is to start with a known problem and figure out what data is required to solve that problem. You often need to seek things that you don't know to look for. Consider this example: Perhaps you have customers with service-level agreements (SLAs), and you find that you are giving them discounts because they are having voice issues over the network and you are not meeting the SLAs. This is costing your company money. You research what you need to analyze in order to understand why this happens, perhaps using voice drop and latency data from your environment. When you finally get these data, you build a proposed model that identifies that higher latency with specific versions of software on network routers is common on devices in the network path for customers who are asking for refunds. Then you deploy the model to flag these “SLA suckers” in your production systems and then validate that the model is effective as the SLA issues have gone away. In this case, *deploy* means that your model is watching your daily inventory data and looking for a device that matches the parameters that you have seen are problematic. What may have been a very complex model has a simple deployment.

Whether starting at data or at a business problem, ultimately solving the problem represents the value to your company and to you as an analyst. Both of these approaches follow many of the same steps on the analytics journey, but they often use different terminology. They are both about turning data into value, regardless of starting point, direction, or approach. Figure 2-4 provides a more detailed perspective that illustrates that these two approaches can work in the same environment on the same data and the very same problem statement. Simply put, all of the work and due diligence needs to be done to have a fully operational (with models built, tested, and deployed), end-to-end use case that provides real, continuous value.

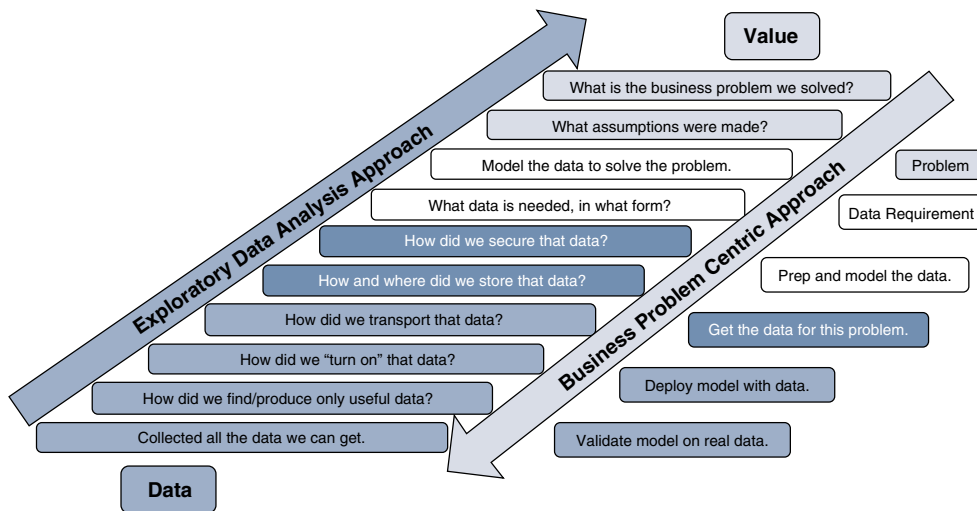


Figure 2-4 Detailed Comparison of Data Versus Problem Approaches

There are a wide variety of detailed approaches and frameworks available in industry today, such as CRISP-DM (cross-industry standard process for data mining) and SEMMA (Sample Explore, Modify, Model, and Assess), and they all generally follow these same principles. Pick something that fits your style and roll with it. Regardless of your approach, the primary goal is to create useful solutions in your problem space by combining the data you have with data science techniques to develop use cases that bring insights to the forefront.

Distinction Between the Use Case and the Solution

Let's slow down a bit and clarify a few terms. Basically, a *use case* is simply a description of a problem that you solve by combining data and data science and applying analytics. The underlying algorithms and models comprise the actual analytics solution. In the case of Amazon, for example, the use case is getting you to spend more money. Amazon does this by showing you what other people have also bought in addition to buying the same item that are purchasing. The intuition behind this is that you will buy more things because

other people like you needed those things when they purchased the same item that you did. The model is there to uncover that and remind you that you may also need to purchase those other things. Very helpful, right?

From the exploratory data approach, Amazon might want to do something with the data it has about what people are buying online. It can then collect the high patterns of common sets of purchases. Then, for patterns that are close but missing just a few items, Amazon may assume that those people just “forgot” to purchase something they needed because everyone else purchased the entire “item set” found in the data. Amazon might then use software implementation to find the people who “forgot” and remind them that they might need the other common items. Then Amazon can validate the effectiveness by tracking purchases of items that the model suggested.

From a business problem approach, Amazon might look at wanting to increase sales, and it might assume (or find research which suggests) that, if reminded, people often purchase common companion items to what they are currently viewing or have in their shopping carts. In order to implement this, Amazon might collect buying pattern data to determine these companion items. The company might then suggest that people may also want to purchase these items. Amazon can then validate the effectiveness by tracking purchases of suggested items.

Do you see how both of these approaches reach the same final solution?

The Amazon case is about increasing sales of items. In predictive analytics, the use case may be about predicting home values or car values. More simply, the use case may be the ability to predict a continuous number from historical numbers. No matter the use case, you can view analytics as simply the application of data and data science to the problem domain. You can choose how you approach finding and building the solutions either by using the data as a guide or by dissecting the stated problem.

Logical Models for Data Science and Data

This section discusses analytics solutions that you model and build for the purpose of deployment to your environment. When I was working with Cisco customers in the early days of analytics, it became clear that setting up the entire data and data science pipeline as a working application on a production network was a bit confusing to many customers, as well as to traditional Cisco engineers.

Many customers thought that they could simply buy network analytics software and install it onto the network as they would any other application—and they would have fully insightful analytics. This, of course, is not the case. Analytics packages integrate into the very same networks for which you build models to run. We can use this situation to introduce the concept of an *overlay*, which is a very important concept for understanding network data (covered in Chapter 3, “Understanding Networking Data Sources”). Analytics packages installed on computers that sit on networks can *build* the models as discussed earlier, but when it is time to *deploy* the models that include data feeds from network environments, the analytics packages often have tendrils that reach deep into

the network and IT systems. Further, these solutions can interface with business and customer data systems that exist elsewhere in the network. Designing such a system can be daunting because most applications on a network do not interact with the underlying hardware. A second important term you should understand is the *underlay*.

Analytics as an Overlay

So how do data and analytics applications fit within network architectures? In this context, you need to know the systems and software that consume the data, and you need to use data science to provide solutions as general applications. If you are using some data science packages or platforms today, then this idea should be familiar to you. These applications take data from the infrastructure (perhaps through a central data store) and combine it with other applications data from systems that reside within the IT infrastructure.

This means the solution is analyzing the very same infrastructure in which it resides, along with a whole host of other applications. In networking, an *overlay* is a solution that is abstracted from the underlying physical infrastructure in some way. Networking purists may not use the term *overlay* for applications, but it is used here because it is an important distinction needed to set up the data discussion in the next chapter. Your model, when implemented in production on a live network, is just an overlay instance of an application, much like other overlay application instances riding on the same network.

This concept of network layers and overlay/underlay is why networking is often blamed for fault or outage—because the network underlays all applications (and other network instances, as discussed in the next chapter). Most applications, if looked at from an application-centric view, are simply overlays onto the underlying network infrastructure. New networking solutions such as Cisco Application Centric Infrastructure (ACI) and common software-defined wide area networks (SD-WANs) such as Cisco iWAN+Viptela take overlay networking to a completely new level by adding additional layers of policy and network segmentation. In case you have not yet surmised, you probably should have a rock-solid underlay network if you want to run all these overlay applications, virtual private networks (VPNs), and analytics solutions on it.

Let's look at an example here to explain overlays. Consider your very own driving patterns (or walking patterns, if you are urban) and the roads or infrastructure that you use to get around. You are one overlay on the world around you. Your neighbor traveling is another overlay. Perhaps your overlay is “going to work,” and your neighbor's overlay for the day is “going shopping.” You are both using the same infrastructure but doing your own things, based on your interactions with the underlay (walkways, roads, bridges, home, offices, stores, and anything else that you interact with). Each of us is an individual “instance” using the underlay, much as applications are instances on networks. There could be hundreds or even thousands of these applications—or millions of people using the roadway system. The underlay itself has lots of possible “layers,” such as the physical roads and intersections and the controls such as signs and lights. Unseen to you, and therefore “virtual,” is probably some satellite layer where GPS is making decisions about how another application overlay (a delivery truck) should be using the underlay (roads).

This concept of overlays and layers, both physical and virtual, for applications as well as networks, was a big epiphany for me when I finally got it. The very networks themselves have layers and planes of operations. I recall it just clicking one day that the packets (routing protocol packets) that were being used to “set up” packet forwarding for a path in my network were using the same infrastructure that they were actually setting up. That is like me controlling the stoplights and walk signs as I go to work, while I am trying to get there. We’ll talk more about this “control plane” later. For now, let’s focus on what is involved with an analytics infrastructure overlay model.

By now, I hope that I have convinced you that this concept of some virtual overlay of functionality on a physical set of gear is very common in networking today. Let’s now look at an analytics infrastructure overlay diagram to illustrate that the data and data science come together to form the use cases of always-on models running in your IT environment. Note in Figure 2-5 how other data, such as customer, business, or operations data, is exported from other application overlays and imported into yours.

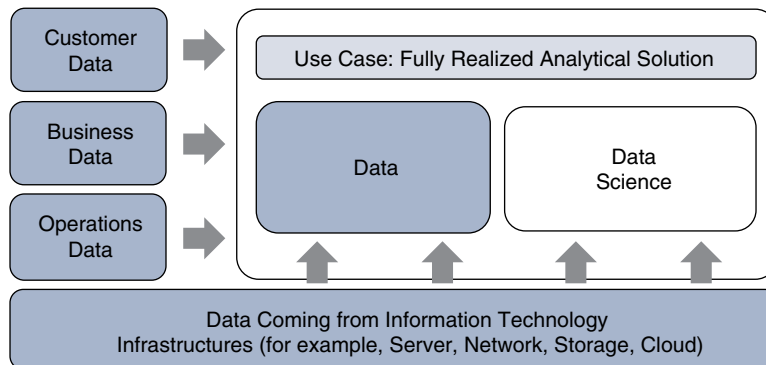


Figure 2-5 *Analytics Solution Overlay*

In today’s digital environment, consider that all the data you need for analysis is produced by some system that is reachable through a network. Since everyone is connected, this is the very same network where you will use some system to collect and store this data. You will most likely deploy your favorite data science tools on this network as well. Your role as the analytics expert here is to make sure you identify how this is set up, such that you successfully set up the data sources that you need to build your analytics use case. You must ensure these data sources are available to the proper layer—your layer—of the network.

The concept of customer, business, and operations data may be new, so let’s get right to the key value. If you used analytics in your customer space, you know who your valuable customers are (and, conversely, which customers are more costly than they are worth). This adds context to findings from the network, as does the business context (which network components have the greatest impact) and operations (where you are spending

excessive time and money in the network). Bringing all these data together allows you to develop use cases with relevant context that will be noticed by business sponsors and stakeholders at higher levels in your company.

As mentioned earlier in this chapter, you can build a model with batches of data, but deploying an active model into your environment requires planning and setup of the data sources needed to “feed” your model as it runs every day in the environment. This may also include context data from other customer or business applications in the network environment. Once you have built a model and wish to operationalize it, making sure that everything properly feeds into your data pipelines is crucial—including the customer, business, operations, and other applications data.

Analytics Infrastructure Model

This section moves away from the overlays and network data to focus entirely on building an analytics solution. (We revisit the concepts of layers and overlays in the next chapter, when we dive deeper into the data sources in the networking domain.) In the case of IT networking, there are many types of deep technical data sources coming up from the environment, and you may need to combine them with data coming from business or operations systems in a common environment in order to provide relevance to the business. You use this data in the data science space with maturity levels of usage, as discussed in Chapter 1. So how can you think about data that is just “out there in the ether” in such a way that you can get to actual analytics use cases? All this is data that you define or create. This is just one component of a model that looks at the required data and components of the analytics use cases.

Figure 2-6 is a simple model for thinking about the flow of data for building deployable, operationalized models that provide analytics solutions. We can call this a simple model for analytics infrastructure, and, as shown in the figure, we can contrast this model with a problem-centric approach used by a traditional business analyst.

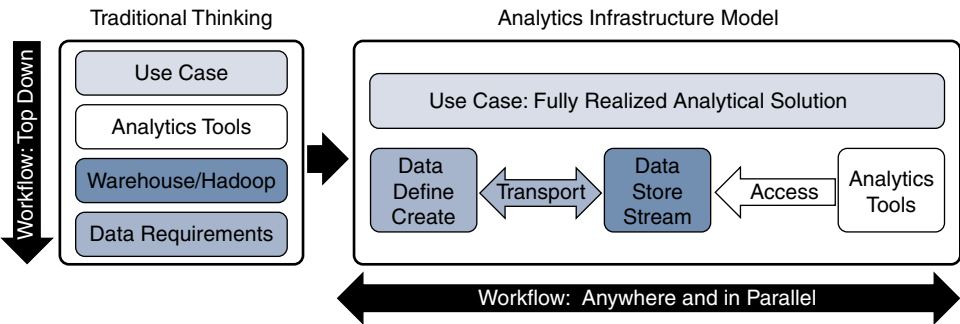


Figure 2-6 Traditional Analyst Thinking Versus Analytics Infrastructure Model

No, analytics infrastructure is not artificial intelligence. Due to the focus on the lower levels of infrastructure data for analytics usage, this analytics infrastructure name fits best. The goal is to identify how to build analytics solutions much the same way you have

built LAN, WAN, wireless, and data center network infrastructures for years. Assembling a full architecture to extract value from data to solve a business problem is an infrastructure in itself. This is very much like an end-to-end application design or an end-to-end networking design, but with a focus on analytics solutions only.

The analytics infrastructure model used in IT networking differs from traditional analyst thinking in that it involves always looking to build repeatable, reusable, flexible solutions and not just find a data requirement for a single problem. This means that once you set up a data source—perhaps from routers, switches, databases, third-party systems, network collectors, or network management systems—you want to use that data source for multiple applications. You may want to replicate that data pipeline across other components and devices so others in the company can use it. This is the “build once, use many” paradigm that is common in Cisco Services and in Cisco products. Solutions built on standard interfaces are connected together to form new solutions. These solutions are reused as many times as needed. Analytics infrastructure model components can be used as many times as needed.

It is important to use standards-based data acquisition technologies and perhaps secure the transport and access around the central data cleansing, sharing, and storage of any networking data. This further ensures the reusability of your work for other solutions. Many such standard data acquisition techniques for the network layer are discussed in Chapter 4, “Accessing Data from Network Components.”

At the far right of the model in Figure 2-6, you want to use any data science tool or package you can to access and analyze your data to create new use cases. Perhaps one package builds a model that is implemented in code, and another package produces the data visualization to show what is happening. The components in the various parts of the model are pluggable so that parts (for example, a transport or a database) could be swapped out with suitable replacements. The role and functionality of a component, not the vendor or type, is what is important.

Finally, you want to be able to work this in an Agile manner and not depend on the top-down Waterfall methods used in traditional solution design. You can work in parallel in any sections of this analytics infrastructure model to help build out the components you need to enable in order to operationalize any analytics model onto any network infrastructure. When you have a team with different areas of expertise along the analytics infrastructure model components, the process is accelerated.

Later in the book, this model is referenced as an aid to solution building. The analytics infrastructure model is very much a generalized model, but it is open, flexible, and usable across many different job roles, both technical and nontechnical, and allows for discussion across silos of people with whom you need to interface. All components are equally important and should be used to aid in the design of analytics solutions.

The analytics infrastructure model (shown enlarged in in Figure 2-7) also differs from many traditional development models in that it segments functions by job roles, which allows for the aforementioned Agile parallel development work. Each of these job roles may still use specialized models within its own functions. For example, a data scientist

might use a preferred methodology and analytics tools to explore the data that you provided in the data storage location. As a networking professional, defining and creating data (far left) in your domain of expertise is where you play, and it is equally as important as the setup of the big data infrastructure (center of the model) or the analysis of the data using specialized tools and algorithms (far right).

Analytics Infrastructure (AI) Model

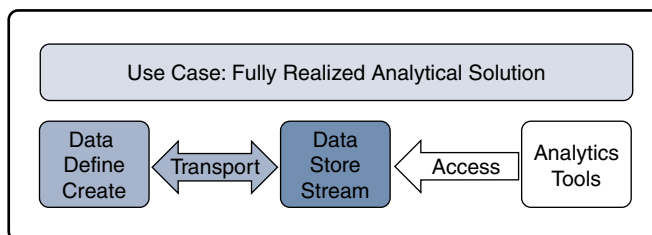


Figure 2-7 *Analytics Infrastructure Model for Developing Analytics Solutions*

Here is a simple elevator pitch for the analytics infrastructure model: “Data is defined, created, or produced in some system from which it is moved into a place where it is stored, shared, or streamed to interested users and data science consumers. Domain-specific solutions using data science tools, techniques, and methodologies provide the analysis and use cases from this data. A fully realized solution crosses all of the data, data storage, and data science components to deliver a use case that is relevant to the business.”

As mentioned in Chapter 1, this book spends little time on “the engine,” which is the center of this model, identified as the big data layer shown in Figure 2-8. When I refer to anything in this engine space, I call out the function, such as “store the data in a database” or “stream the data from the Kafka bus.” Due to the number of open source and commercial components and options in this space, there is an almost infinite combination of options and instructions readily available to build the capabilities that you need.

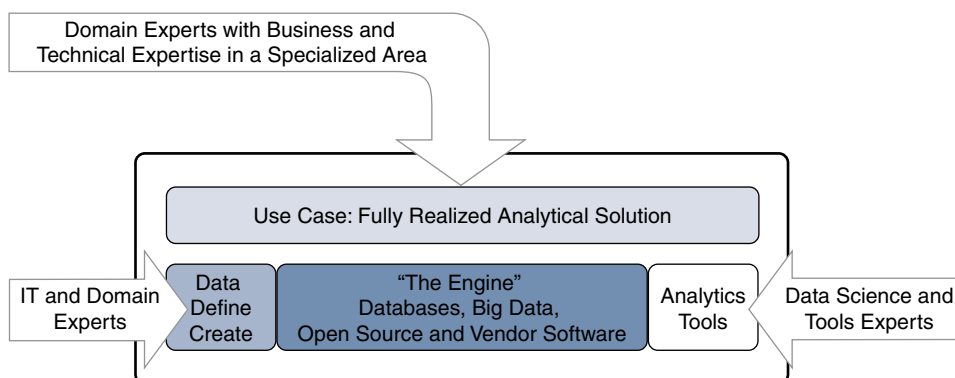


Figure 2-8 *Roles and the Analytics Infrastructure Model*

It is not important that you understand how “the engine” in this car works; rather, it is important to ensure that you can use it to drive toward analytics solutions. Whether using open source big data infrastructure or packages from vendors in this space, you can readily find instructions to transport, store, share, and stream and provide access to the data on the Internet. Run a web search on “data engineering pipelines” and “big data architecture,” and you will find a vast array of information and literature in the data engineering space.

The book aims to help you understand the job roles around the common big data infrastructure, along with data, data science, and use cases. The following are some of the key roles you need to understand:

- **Data domain experts**—These experts are familiar with the data and data sources.
- **Analytics or business domain experts**—These experts are familiar with the problems that need to be solved (or questions that need to be answered).
- **Data scientists**—These experts have knowledge of the tools and techniques available to find the answers or insights desired by the business or technical experts in the company.

The analytics infrastructure model is location agnostic, which is why you see callouts for data transport and data access. This overall model approach applies regardless of technology or location. Analytics systems can be on-premises, in the cloud, or hybrid solutions, as long as all the parts are available for use. Regardless of where the analytics is used, the networking team is usually involved in ensuring that the data is in the right place for the analysis. Recall from the overlay discussion earlier in the chapter that the underlay is necessary for the overlay to work. Parts of this analysis may exist in the cloud, other parts on your laptop, and other parts on captive customer relationship management (CRM) systems on your corporate networks. You can use the analytics infrastructure model to diagram a solution flow that results in a fully realized analytics use case.

Depending on your primary role, you may be involved in gathering the data, moving the data, storing the data, sharing the data, streaming the data, archiving the data, or providing the analytics analysis. You may be ready to build the entire use case. There are many perspectives when discussing analytics solutions. Sometimes you will wear multiple hats. Sometimes you will work with many people; sometimes you will work alone if you have learned to fill all the required roles. If you decide to work alone, make sure you have access to resources or expertise to validate findings in areas that are new to you. You don’t want to spend a significant amount of time uncovering something that is already general knowledge and therefore not very useful to your stakeholders.

Building your components using the analytics infrastructure model ensures that you have reusable assets in each of the major parts of the model. Sometimes you will spend many hours, days, or weeks developing an analysis, only to find that there are no interesting insights. This is common in data science work. By using the analytics infrastructure model, you can maintain some parts of your work to build other solutions in the future.

The Analytics Infrastructure Model In Depth

So what are the “reusable and repeatable components” touted in the analytics infrastructure model? This section digs into the details of what needs to happen in each part of the model. Let’s start by digging into the lower-left data component of the model, looking at the data that is commonly available in an IT environment. Data pipelines are big business and well covered in the “for fee” and free literature.

Building analytics models usually involves getting and modeling some data from the infrastructure, which includes spending a lot of time on research, data munging, data wrangling, data cleansing, ETL (Extract, Transform, Load), and other tasks. The true power of what you build is realized when you deploy your model into an environment and turn it on. As the analytics infrastructure model indicates, this involves acquiring useful data and transporting it into an accessible place. What are some examples of the data that you may need to acquire? Expanding on the data and transport sections of the model in Figure 2-9, you will find many familiar terms related to the combination of networking and data.

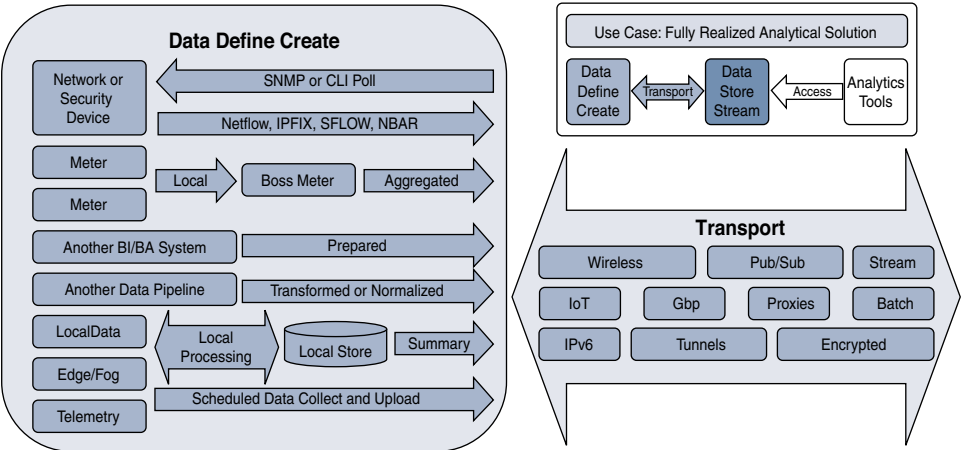


Figure 2-9 Analytics Infrastructure Model Data and Transport Examples

Implementing a model involves setting up a full pipeline of new data (or reusing a part of a previous pipeline) to run through your newly modeled use cases, and this involves “turning on” the right data and transporting it to where you need it to be. Sometimes this is kept local (as in the case of many Internet of Things [IoT] solutions), and sometimes data needs to be transported. This is all part of setting up the full data pipeline. If you need to examine data in flight for some real-time analysis, you may need to have full data streaming capabilities built from the data source to the place where the analysis happens.

Do not let the number of words in Figure 2-9 scare you; not all of these things are used. This diagram simply shares some possibilities and is in no way a complete set of everything that could be at each layer.

To illustrate how this model works, let's return to the earlier example of the router problem. If latency and sometimes router crashes are associated with a memory leak in some software versions of a network router, you can use a telemetry data source to access memory statistics in a router. Telemetry data, covered in Chapter 4, is a push model whereby network devices send periodic or triggered updates to a specified location in the analytics solution overlay. Telemetry is like a hospital heart monitor that gets constant updates from probes on a patient. Getting router memory-related telemetry data to the analytics layer involves using the components identified in white in Figure 2-10—for just a single stream. By setting this up for use, you create a reusable data pipeline with telemetry-supplied data. A new instance of this full pipeline must be set up for each device in the network that you want to analyze for this problem. The hard part—the “feature engineering” of building a pipeline—needs to happen only once. You can easily replicate and reuse that pipeline, as you now have your memory “heart rate monitor” set up for all devices that support telemetry. The left side of Figure 2-10 shows many ways data can originate, including methods and local data manipulations, and the arrow on the right side of the figure shows potential transport methods. There are many types of data sources and access methods.

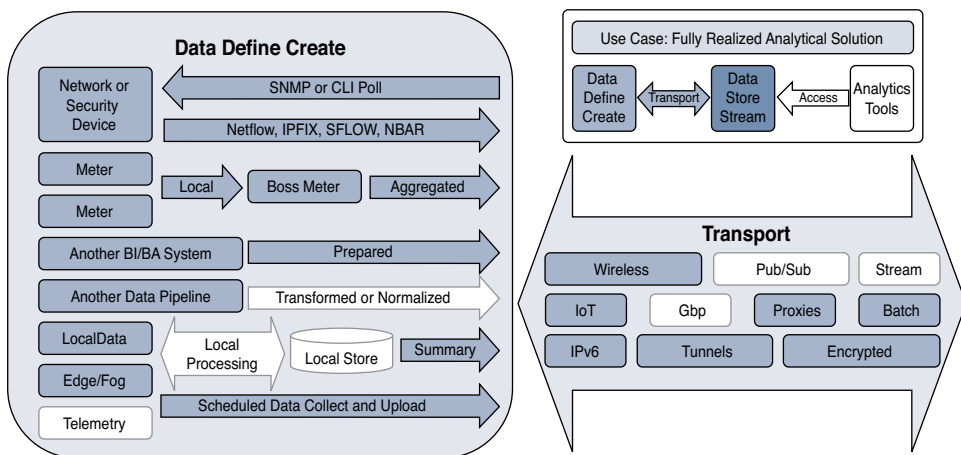


Figure 2-10 *Analytics Infrastructure Model Telemetry Data Example*

In this example, you are taking in telemetry data at the data layer, and you may also do some local processing of the data and store it in a localized database. In order to send the memory data upstream, you may standardize it to a megabyte or gigabyte number, standardize it to a “z” value, or perform some other transformation. This design work must happen once for each source. Does this data transformation and standardization stuff sound tedious to you? Consider that in 1999, NASA lost a \$125 million Mars orbiter due to a mismatch of metric to English units in the software. Standardization, transformation, and data design are important.

Now, assuming that you have the telemetry data you want, how do you send it to a storage location? You need to choose transport options. For this example, say that you choose to send a steady stream to a Kafka publisher/subscriber location by using Google Protocol Buffers (GPB) encoding. There are lots of capabilities, and lots of options, but after a one-time design, learning, and setup process, you can document it and use it over and over again. What happens when you need to check another router for this same memory leak? You call up the specification that you designed here and retrofit it for the new requirement.

While data platforms and data movement are not covered in detail in this book, it is important that you have a basic understanding of what is happening inside the engine, all around the “the data platform.”

The Analytics Engine

Unless you have a dedicated team to do this, much of this data storage work and setup may fall in your lap during model building. You can find a wealth of instruction for building your own data environments by doing a simple Internet search. Figure 2-11 shows many of the activities related to this layer. Note how the transport and data access relate to the configuration of this centralized engine. You need a destination for your prepared data, and you need to know the central location configuration so you can send it there. On the access side, the central data location will have access methods and security, which you must know or design in order to consume data from this layer.

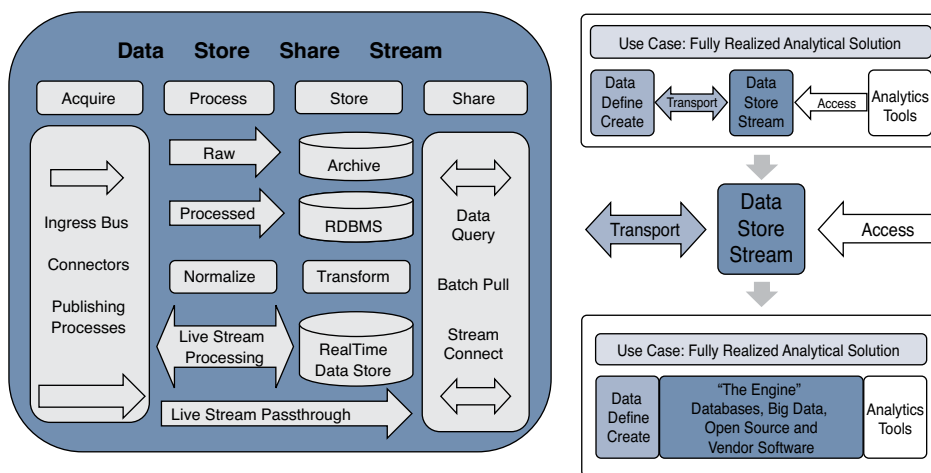


Figure 2-11 *The Analytics Infrastructure Model Data Engine*

Once you have defined the data parameters, and you understand where to send the data, you can move the data into the engine for storage, analysis, and streaming. From each individual source perspective, the choice comes down to push or pull mechanisms, as

per the component capabilities available to you in your data-producing entities. This may include pull methods using polling protocols such as Simple Network Management Protocol (SNMP) or push methods such as the telemetry used in this example.

This centralized data-engineering environment is where the Hadoop, Spark, or commercial big data platform lives. Such platforms are often set up with receivers for each individual type of data. The pipeline definition for each of these types of data includes the type and configuration of this receiver at the central data environment. Very common within analytics engines today is something called a publisher/subscriber environment, or “pub/sub” bus. Apache Kafka is a very common bus used in these engines today.

A good analogy for the pub/sub bus is broadcast TV channels with a DVR. Data feeds (through analytics infrastructure model transports) are sent to specific channels from data producers, and subscribers (data consumers) can choose to listen to these data feeds and subscribe (using some analytics infrastructure model access method, such as a Kafka consumer) to receive them. In this telemetry example, the telemetry receiver takes interesting data and copies or publishes it to this bus environment. Any package requiring data for doing analytics subscribes to a stream and has it copied to its location for analysis in the case of streaming data. This separation of the data producers and consumers makes for very flexible application development. It also means that your single data feed could be simultaneously used by multiple consumers.

What else happens here at the central environment? There are receivers for just about any data type. You can both stream into the centralized data environment and out of the centralized environment in real time. While this is happening, processing functions decode the stream, extract interesting data, and put the data into relational databases or raw storage. It is also common to copy items from the data into some type of “object” storage environment for future processing. During the transform process, you may standardize, summarize, normalize, and store data. You transform data to something that is usable and standardized to fit into some existing analytics use case. This centralized environment, often called the “data warehouse” or “data lake,” is accessed through a variety of methods, such as Structured Query Language (SQL), application programming interface (API) calls, Kafka consumers, or even simple file access, just to name a few.

Before the data is stored at the central location, you may need to adjust these data, including doing the following:

- Data cleansing to make sure the data matches known types that your storage expects
- Data reconciliation, including filling missing data, cleaning up formats, removing duplicates, or bounding values to known ranges
- Deriving or generating any new values that you want included in the records
- Splitting or combining data into meaningful values for the domain
- Standardizing the data ingress or splitting a stream to keep standardized and raw data

Now let's return to the memory example: These telemetry data streams (subject: memory leak) from the network infrastructure must now be made available to the analytics tools and data scientists for analysis or application of the models. This availability must happen through the analytics engine part of the analytics infrastructure model. Figure 2-12 shows what types of activities are involved if there is a query or request for this data stream from analytics tools or packages. This query is requesting that a live feed of the stream be passed through the publisher/subscriber bus architecture and a normalized feed of the same stream be copied to a database for batch analysis. This is all set up in the software at the central data location.

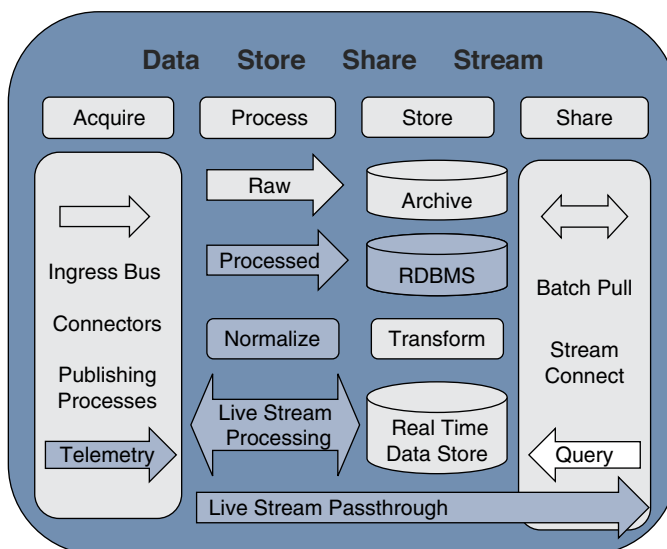


Figure 2-12 *Analytics Infrastructure Model Streaming Data Example*

Data Science

Data science is the sexy part of analytics. Data science includes the data mining, statistics, visualization, and modeling activities performed on readily available data. People often forget about the requirements to get the proper data to solve the individual use cases. The focus for most analysts is to start with the business problem first and then determine which type of data is required to solve or provide insights from the particular use case. Do not underestimate the time and effort required to set up the data for these use cases. Research shows that analysts spend 80% or more of their time on acquiring, cleaning, normalizing, transforming, or otherwise manipulating the data. I've spent upward of 90% on some problems.

Analysts must spend so much time because analytics algorithms require specific representations or encodings of the data. In some cases, encoding is required because the raw stream appears to be gibberish. You can commonly do the transformations, standardizations, and normalizations of data in the data pipeline, depending on the use case. First you need to figure out the required data manipulations through your model building phases; you will ultimately add them inline to the model deployment phases, as shown in the previous diagrams, such that your data arrives at the data science tools ready to use in the models.

The analytics infrastructure model is valuable from the data science tools perspective because you can assume that the data is ready, and you can focus clearly on the data access and the tools you need to work on that data. Now you do the data science part. As shown in Figure 2-13, the data science part of the model highlights tools, processes, and capabilities that are required to build and deploy models.

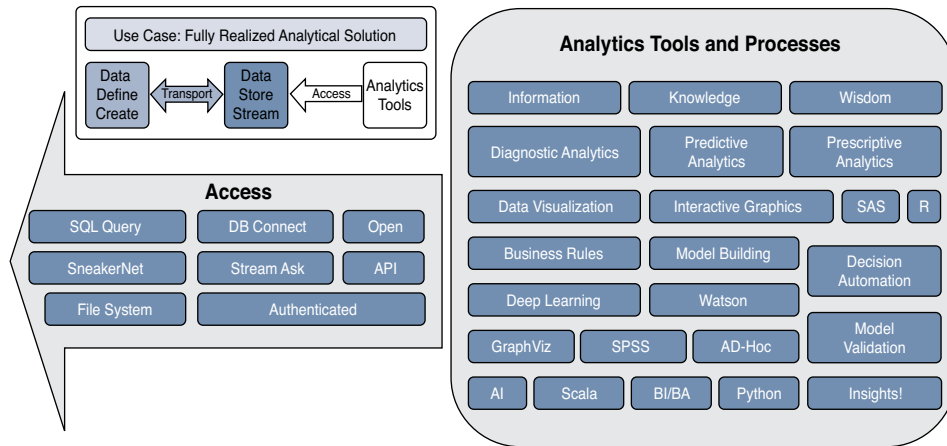


Figure 2-13 *Analytics Infrastructure Model Analytics Tools and Processes*

Going back to the streaming telemetry memory leak example, what should you do here? As highlighted in Figure 2-14, you use a SQL query to an API to set up the storage of the summary data. You also request full stream access to provide data visualization. Data visualization then easily shows both your technical and nontechnical stakeholders the obvious untamed growth of memory on certain platforms, which ultimately provides some “diagnostic analytics.” Insight: This platform, as you have it deployed, leaks memory with the current network conditions. You clearly show this with a data visualization, and now that you have diagnosed it, you can even build a predictive model for catching it before it becomes a problem in your network.

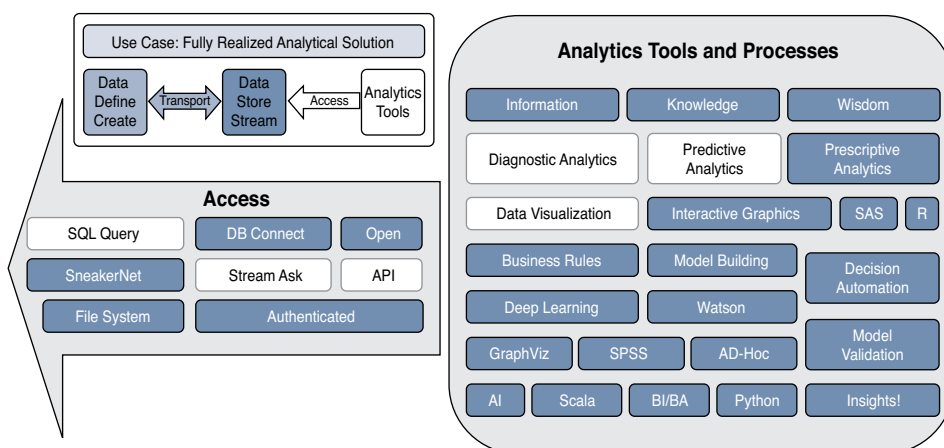


Figure 2-14 *Analytics Infrastructure Model Streaming Analytics Example*

Analytics Use Cases

The final section of the analytics infrastructure model is the use cases built on all this work that you performed: the “analytics solution.” Figure 2-15 shows some examples of generalized use cases that are supported with this example. You can build a predictive application for your memory case and use survival analysis techniques to determine which routers will hit this memory leak in the future. You can also use your analytics for decision support to management in order to prioritize activities required to correct the memory issue. Survival analysis here is an example of how to use common industry intuition to develop use cases for your own space. Survival analysis is about recognizing that something will not survive, such as a part in an industrial machine. You can use the very same techniques to recognize that a router will not survive a memory leak.

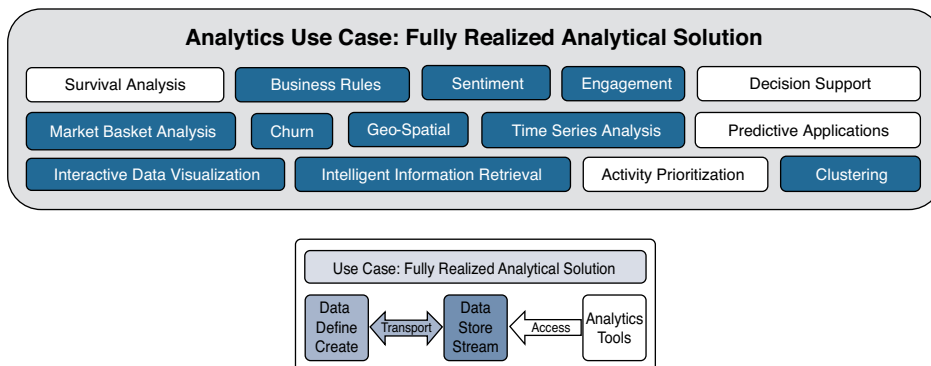


Figure 2-15 *Analytics Infrastructure Model Analytics Use Cases Example*

As you go through the analytics use cases in later chapters, it is up to you and your context bias to determine how far to take each of the use cases. Often simple descriptive

analytics or a picture of what is in the environment is enough to provide a solution. Working toward wisdom from the data for predictive, prescriptive, and preemptive analytics solutions is well worth the effort in many cases. The determination of whether it is worth the effort is highly dependent on the capabilities of the systems, people, process, and tools available in your organization (including you).

Figure 2-16 shows where fully automated service assurance is added to the analytics infrastructure model. When you combine the analytics solution with fully automated remediation, you build a full-service assurance layer. Cisco builds full-service assurance layers into many architectures today, in solutions such as Digital Network Architecture (DNA), Application Centric Infrastructure (ACI), Crosswork Network Automation, and more that are coming in the near future. Automation is beyond the scope of this book, but rest assured that your analytics solutions are a valuable source for the automated systems to realize full-service assurance.

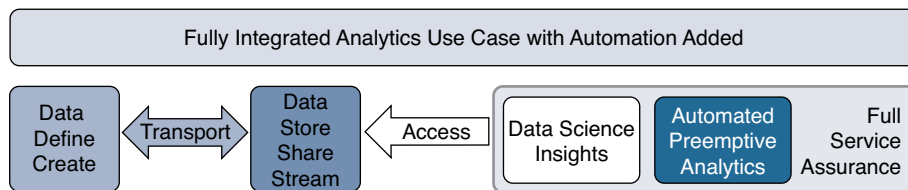


Figure 2-16 *Analytics Infrastructure Model with Service Assurance Attachment*

Summary

Now you understand that there is a method to the analytics madness. You also now know that there are multiple approaches you can take to data science problems. You understand that building a model on captive data in your own machine is an entirely different process from deploying a model in a production environment. You also understand different approaches to the process and that you and your stakeholders may each show preferences for different ones. Whether you are starting with the data exploration or the problem statement, you can find useful and interesting insights.

You may also have had your first introduction to the overlays and underlays concepts, which are important concepts as you go deeper into the data that is available to you from your network in the next chapter. Getting data to and from other overlay applications, as well as to and from other layers of the network is an important part of building complete solutions.

You now have a generalized analytics infrastructure model that helps you understand how the parts of analytics solutions come together to form a use case. Further, you understand that using the analytics infrastructure model allows you to build many different levels of analytics and provides repeatable, reusable components. You can choose how mature you wish your solution to be, based on factors from your own environment. The next few chapters take a deep dive into understanding the networking data from that environment.

This page intentionally left blank

Understanding Networking Data Sources

This chapter begins to examine the complexities of networking data. Understanding and preparing all the data coming from the IT infrastructure is part of the data engineering process within analytics solution building. Data engineering involves the setup of data pipelines from the data source to the centralized data environment, in a format that is ready for use by analytics tools. From there, data may be stored, shared, or streamed into dedicated environments where you perform data science analysis. In most cases, there is also a process of cleaning up or normalizing data at this layer. ETL (Extract, Transform, Load) is a carryover acronym from database systems that were commonly used at the data storage layer. ETL simply refers to getting data; normalizing, standardizing, or otherwise manipulating it; and “loading” it into the data layer for future use. Data can be loaded in structured or unstructured form, or it can be streamed right through to some application that requires real-time data. Sometimes analysis is performed on the data right where it is produced. Before you can do any of that, you need to identify how to define, create, extract, and transport the right data for your analysis, which is an integral part of the analytics infrastructure model, shown in Figure 3-1.

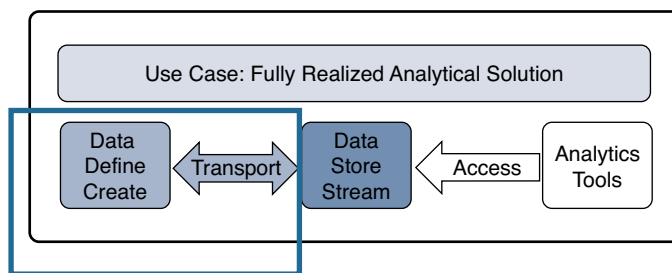


Figure 3-1 *The Analytics Infrastructure Model Focus Area for This Chapter*

Chapter 2, “Approaches for Analytics and Data Science,” provides an overlay example of applications and analytics that serves as a backdrop here. There are layers of virtual abstraction up and down and side by side in IT networks. There are also instances of applications and overlays side by side. Networks can be very complex and confusing. As I journeyed through learning about network virtualization, server virtualization, OpenStack, and network functions virtualization (NFV), it became obvious to me that it is incredibly important to understand the abstraction layers in networking. Entire companies can exist inside a virtualized server instance, much like a civilization on a flower in *Horton Hears a Who!* (If you have kids you will get this one.) Similarly, an entire company could exist in the cloud, inside a single server.

Planes of Operation on IT Networks

Networking infrastructures exist to provide connectivity for overlay applications to move data between components assembled to perform the application function. Perhaps this is a bunch of servers and databases to run the business, or it may be a collection of high-end graphics processing units (GPUs) to mine bitcoin. Regardless of purpose, such a network is made of routers, switches, and security devices moving data from node to node in a fully connected, highly resilient architecture. This is the lowest layer, and similar whether it is your enterprise network, any cloud provider, or the Internet. At the lowest layer are “big iron” routers and switches and the surrounding security, access, and wireless components.

Software professionals and other IT practitioners may see the data movement between nodes of architecture in their own context, such as servers to servers, applications to applications, or even applications to users. Regardless of what the “node” is for a particular context, there are multiple levels of data available for analysis and multiple “overlay perspectives” of the very same infrastructure. Have you ever seen books about the human body with clear pages that allow you to see the skeleton alone, and then overlay the muscles and organs and other parts onto the skeleton by adding pages one at a time? Networks have many, many top pages to overlay onto the picture of the physical connectivity.

When analyzing data from networking environments, it is necessary to understand the level of abstraction, or the page from which you source data. Recall that you are but an overlay on the roads that you drive. You could analyze the roads, you could analyze your car, or you could analyze the trip, and all these analyses could be entirely independent of each other. This same concept applies in networking: You can analyze the physical network, you can analyze individual packets flowing on that physical network, and you can analyze an application overlay on the network.

So how do all these overlays and underlays fit together in a data sense? In a networking environment, there are three major “planes” of activity. Recall from high school math class that a plane is not actually visible but is a layer that connects things that coexist in the same flat space. Here the term *planes* is used to indicate different levels of operation within a single physical, logical, or virtual entity described as a *network*. Each plane has its own transparency page to flip onto the diagram of the base network. We can

summarize the major planes of operation based on three major functions and assign a data context to each. From a networking perspective, these are the three major planes (see Figure 3-2):

- **Management plane**—This is the plane where you talk to the devices and manage the software, configuration, capabilities, and performance monitoring of the devices.
- **Control plane**—This is the plane where network components talk to each other to set up the paths for data to flow over the network.
- **Data plane**—This is the plane where applications use the network paths to share data.

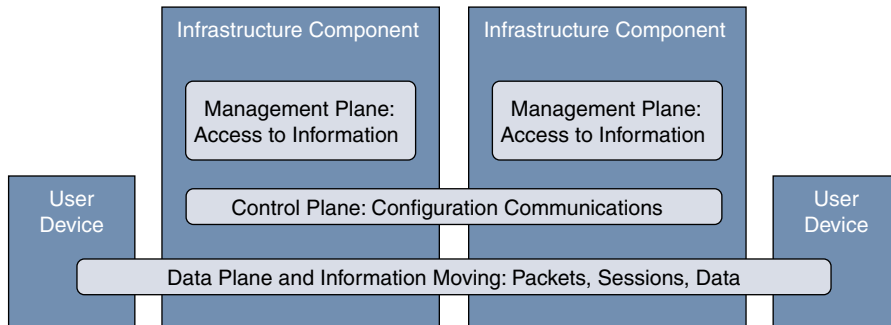


Figure 3-2 *Planes of Operation in IT Networks*

These planes are important because they represent different levels and types of data coming from your infrastructure that you will use differently depending on the analytics solution you are developing. You can build analytics solutions using data from any one or more of these planes.

The *management plane* provides the access to any device on your network, and you use it to communicate with, configure, upgrade, monitor, and extract data from the device. Some of the data you extract is about the *control plane*, which enables communication through a set of static or dynamic configuration rules in network components. These rules allow networking components to operate as a network unit rather than as individual components. You can also use the management plane to get data about the things happening on the *data plane*, where data actually moves around the network (for example, the analytics application data that was previously called an overlay). The software overlay applications in your environment share the data plane. Every network component has these three planes, accessible directly to the device or through a centralized controller that commands many such devices, physical or virtual.

This planes concept is extremely important as you start to work with analytics and more virtualized network architectures and applications. If you already know it, feel free to just skim or skip this section. If you do not, a few analogies in the upcoming pages will aid in your understanding.

In this first example, look at the very simple network diagram shown in Figure 3-3, where two devices are communicating over a very simple routed network of two routers. In this case, you use the management plane to ask the routers about everything in the little deployment—all devices, the networks, the addressing, MAC addresses, IP addresses, and more. The routers have this information in their configuration files.

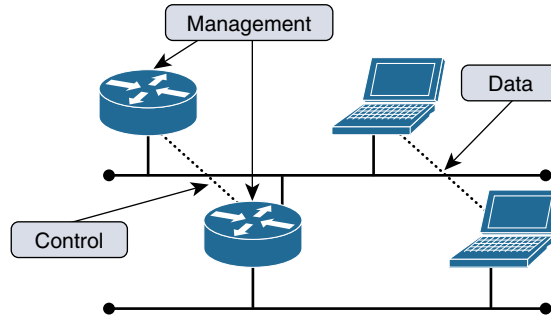


Figure 3-3 Sample Network with Management, Control, and Data Planes Identified

For the two user laptop devices to communicate, they must have connectivity set up for them. The routers on the little network communicate with each other, creating an instance of control plane traffic in order to set up the common network such that the two hosts are communicating with each other. The routers communicate with each other using a routing protocol to share any other networks that each knows about. A type of communication used to configure the devices to forward properly is *control plane communication*—communication between the participating network components to set up the environment for proper data forwarding operation.

I want to add a point of clarification. The routers have a configuration item that instructs them to run the routing protocol. You find this in the configuration you extract using the management plane, and it is a “feature” of the device. This particular feature creates the need to generate control plane traffic communications. The feature configuration is not in the control plane, but it tells you what you should see in terms of control plane activity from the device. Sometimes you associate feature information with the control plane because it is important context for what happens on the control plane communications channels.

The final area here is the data plane, which is the communications plane between the users of the little network. They could be running an analytics application or running Skype. As long as the control plane does its work, a path through the routers is available here for the hosts to talk together on a common *data plane*, enabling the application overlay instance between the two users to work. If you capture the contents of the Skype session from the data plane, you can examine the overlay application Skype in a vacuum. In most traditional networks, the control plane communication is happening across the same data plane paths (unless a special design dictates a completely separate path).

Next, let's look at a second example that is a little more abstract. In this example, a pair of servers provides cloud functionality using OpenStack cloud virtualization, as shown in Figure 3-4. OpenStack is open source software used to build cloud environments on common servers, including virtualized networking components used by the common servers. Everything exists in software, but the planes concept still applies.

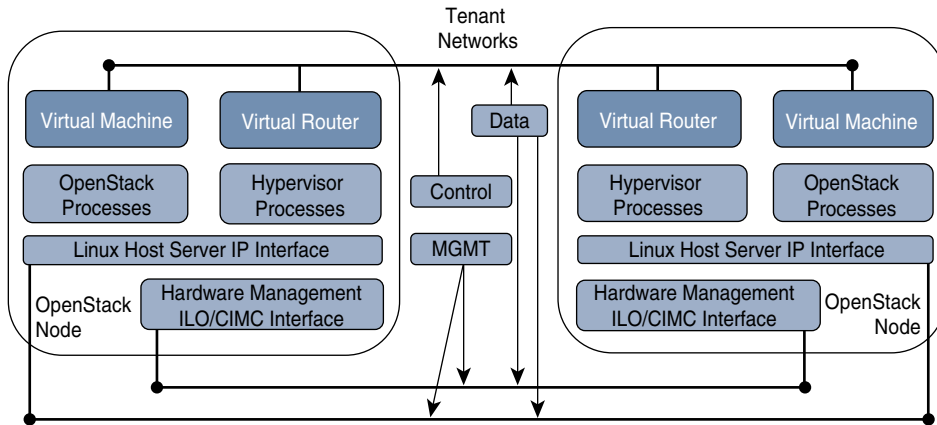


Figure 3-4 *Planes of Operation and OpenStack Nodes*

The management plane is easy, and hopefully you understand this one: The management plane is what you talk to, and it provides information about the other planes, as well as information about the network components (whether they are physical or virtual, server or router) and the features that are configured. Note that there are a couple of management plane connections here now: A Linux operating system connection was added, and you need to talk to the management plane of the server using that network.

In cloud environments, some interfaces perform both management and control plane communications, or there may be separate channels set up for everything. This area is very design specific. In network environments, the control plane communication often uses the data plane path, so that the protocols have actual knowledge of working paths and the experience of using those paths (for example, latency, performance). In this example, these concepts are applied to a server providing OpenStack cloud functionality. The control plane in this case now includes the Linux and OpenStack processes and functions that are required to set up and configure the data plane for forwarding. There could be a lot of control plane, at many layers, in cloud deployments.

A cloud control plane sets up data planes just as in a physical network, and then the data plane communication happens between the virtual hosts in the cloud. Note that this is shown in just a few nodes here, but these are abstracted planes, which means they could extend into hundreds or thousands of cloud hosts just like the ones shown.

When it comes to analytics, each of these planes of activity offers a different type of data for solving use cases. It is common to build solutions entirely from management plane

data, as you will see in Chapter 10, “Developing Real Use Cases: The Power of Statistics,” and Chapter 11, “Developing Real Use Cases: Network Infrastructure Analytics.” Solutions built entirely from captured data plane traffic are also very popular, as you will see in Chapter 13, “Developing Real Use Cases: Data Plane Analytics.” You can use any combination of data from any plane to build solutions that are broader, or you can use focused data from a single plane to examine a specific area of interest.

Things can get more complex, though. Once the control plane sets things up properly, any number of things can happen on the data plane. In cloud and virtualization, a completely new instance of the control plane for some other, virtualized network environment may exist in the data plane. Consider the network and then the cloud example we just went through. Two virtual machines on a network communicate their private business over their own data plane communications. They encrypt their data plane communications. At first glance, this is simply data plane traffic between two hosts, which could be running a Skype session. But then, in the second example, those computers could be building a cloud and might have their own control plane and data plane inside what you see as just a data plane. If one of *their* customers is virtualizing those cloud resources into something else.... Yes, this rabbit hole can go very deep. Let’s look at another analogy here to explore this further.

Consider again that you and every one of your neighbors uses the same infrastructure of roads to come and go. Each of you has your own individual activities, and therefore your behavior on that shared road infrastructure represents your overlays—your “instances” using the infrastructure in separate ways. Your activities are data plane entities there, much like packets and applications riding your corporate networks, or the data from virtual machines in an OpenStack environment. In the roads context, the management plane is the city, county, or town officials that actually build, clean, clear, and repair the roads. Although it affects you at times (everybody loves road repair and construction), their activity is generally separate from yours, and what they care about for the infrastructure is different from your concerns.

The control plane in this example is the communications system of stoplights, stop signs, merge signs, and other components that determine the “rules” for how you use paths on the physical infrastructure. This is a case where the control plane has a dedicated channel that is not part of the data plane. As in the cloud tenant example, you may also have your own additional “family control plane” set of rules for how *your* cars use those roads (for example, 5 miles per hour under the speed limit), which is not related at all to the rules of the other cars on the roads. In this example, you telling your adolescent driver to slow down is control plane communication within your overlay.

Review of the Planes

Before going deeper, let’s review the three planes.

The *management plane* is the part of the infrastructure where you access all the components to learn information about the assets, components, environment, and some

applications. This may include standard items such as power consumption, central processing units (CPUs), memory, or performance counters related to your environment. This is a critical plane of operation as it is the primary mechanism for configuring, monitoring, and getting data from the networking environment—even if the data is describing something on the control or data planes. In the server context using OpenStack, this plane is a combination of a Hewlett-Packard iLO (Integrated Lights Out) or Cisco IMC (Integrated Management Controller) connection, as well as a second connection to the operating system of the device.

The *control plane* is the configuration activity plane. Control plane activities happen in the environment to ensure that you have working data movement across the infrastructure. The activities of the control plane instruct devices about how to forward traffic on the data plane (just as stoplights indicate how to use the roads). You use standard network protocols to configure the data plane forwarding. The communications traffic between these protocols is control plane traffic. Protocol examples are Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) and, at a lower level of the network, Spanning Tree Protocol (STP). Each of these common control plane protocols has both an operating environment and a configured state of features, both of which produce interesting data for analysis of IT environments. Management plane features (configuration items) are often associated with the control plane activities.

The *data plane* consists of actual traffic activity from node to node in an IT networking infrastructure. This is also a valuable data source as it represents the actual data movement in the environment. When looking at data plane traffic, there are often external sensors, appliances, network taps, or some “capture” mechanisms to evaluate the data and information movement. Behavioral analytics and other user-related analysis account for one “sub-plane” that looks at what the users are doing and how they are using the infrastructure. Returning to the traffic analysis analogy, by examining all traffic on the data plane by counting cars at an intersection, it may be determined that a new traffic control device is required at that intersection. Based on examining one sub-plane of traffic, it may be determined that the sub-plane needs some adjustment. Behavioral analysis on your sub-plane or overlay as a member of the all cars data plane may result in you getting a speeding ticket!

I recall first realizing that these planes exist. At first, they were not really a big deal to me because every device was a single entity and performed a single purpose (and I had to walk to work uphill both ways in the snow to work on these devices). But as I started to move into network and server virtualization environments, I realized the absolute necessity of understanding how these planes work because we could all be using the same infrastructure for entirely different purposes—just as my neighbors and I drive the same roads in our neighborhoods to get to work or stores or the airport. If you want to use analytics to find insights about virtualized solutions, you need to understand these planes. The next section goes even deeper and provides a different analogy to bring home the different data types that come from these planes of operation.

Data and the Planes of Operation

You now know about three levels of activity—the three planes of operation in a networking environment. Different people see data from various perspectives, depending on their backgrounds and their current context. If you are a sports fan, the data you see may be the statistics such as batting average or points scored. If you are from a business background, the data you see may be available in business intelligence (BI) or business analytics (BA) dashboards. If you are a network engineer, the data you see may be inventory, configuration, packet, or performance data about network devices, network applications, or users of your network.

Data that comes from the business or applications reporting functions in your company is not part of these three planes, but it provides important context that you may use in analysis. Context is a powerful addition to any solution. Let’s return to our neighbor analogy: Think of you and your family as an application riding on the network. How much money you have in the bank is your “business” data. This has nothing to do with how you are using the infrastructure (for example, roads) or what your application might be (for example, driving to sports practice), but it is very important nonetheless as it has an impact on what you are driving and possible purposes for your being out there on the infrastructure. As more and more of the traditional BI/BA systems are modernized with machine learning, you can use business layer data to provide valuable context to your infrastructure-level analysis. At the time of this writing, net neutrality has been in the news. Using business metrics to prioritize applications on the Internet data plane by interacting directly with the control plane seems like it could become a reality in the near future. The important thing to note is that context data about the business and the applications is outside the foundational network data sources and the three planes (see Figure 3-5). The three planes all provide data about the infrastructure layer only.

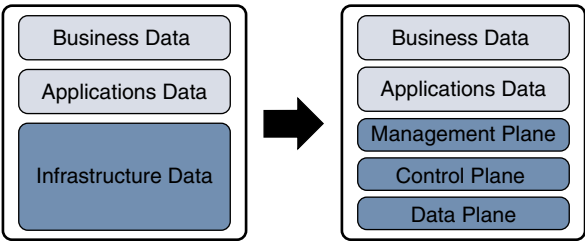


Figure 3-5 *Business and Applications Data Relative to Network Data*

When talking about business, applications, or network data, the term *features* is often used to distinguish between the actual traffic that is flowing on the network and things that are known about the application in the traffic streams. For example, “myApp version 1.0” is a feature about an application riding on the network. If you want to see how much traffic is actually flowing from a user to myApp, you need to analyze the network data plane. If you want to see the primary path for a user to get to myApp, you need to examine the control plane configuration rules. Then you can validate your configuration intent by asking

questions of the management plane, and you can further validate that it is operating as instructed by examining the data plane activity with packet captures.

In an attempt to clarify this complex topic, let's consider one final analogy related to sports. Say that the "network" is a sports league, and you know a player playing within it (much like a router, switch, or server sitting in an IT network). Management plane conversations are analogous to conversations with sports players to gain data. You learn a player's name, height, weight, and years of experience. In fact, you can use his or her primary communication method (the management plane) to find out all kinds of features about the player. Combining this with the "driving on roads" infrastructure analogy, you use the management plane to ask the player where he or she is going. This can help you determine what application (such as going to practice) the player is using the roads infrastructure for today.

Note that you have not yet made any assessment of how good a player is, how good your network devices are, or how good the roads in your neighborhood look today. You are just collecting data about a sports player, an overlay application, or your network of roads. You are collecting features. The mappings in Figure 3-6 show how real-world activities of a sports player map to the planes.

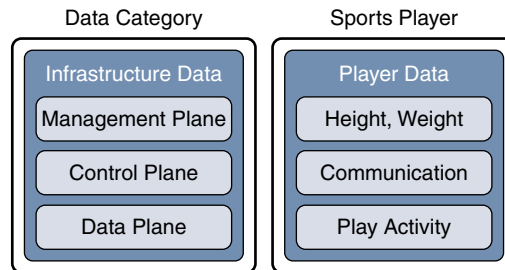


Figure 3-6 *Planes Data Sports Player Analogy*

The control plane in a network is like player communication with other players in sports to set up a play or an approach that the team will try. American football teams line up and run through certain plays against defensive alignments in order to find the optimal or best way to run a play. The same thing happens in soccer, basketball, hockey, and any other sport where there are defined plays. The control plane is the layer of communication used between the players to ensure that everybody knows his or her role in the upcoming activity. The control plane on a network, like players communicating during sports play, is always on and always working in reaction to current conditions.

That last distinction is very important for understanding the control plane of the network. Like athletes practicing plays so that they know what to do given a certain situation, network components share a set of instructions for how the network components should react to various conditions on the network. You may have heard of Spanning Tree Protocol, OSPF, or BGP, which are like plays where all the players agree on what happens at game time. They all have a "protocol" for dealing with certain types of situations. Your traffic goes across your network because some control plane protocol made a decision

about the best way to get you from your source to your destination; more importantly, the protocol also set up the environment to make it happen. If we again go back to the example of you as a user of the network of roads in your neighborhood, the control plane is the system of instructions that happened between all of the stoplights to ensure orderly and fair sharing of the roads.

You will find that a mismatch between the control plane instruction and the data plane forwarding is one of the most frustrating and hard-to-find problems in IT networks. Just gaining an understanding that this type of problem exists will help you in your everyday troubleshooting. Imagine the frustration of a coach who has trained his sports players to run a particular play, but on game day, they do something different from what he taught them. That is like a control plane/data plane mismatch, which can be catastrophic in computer networks. When you have checked everything, and it all seems to be correct, look at the data plane to see if things are moving as instructed.

How do you know that the data plane is performing the functions the way you intended them to happen? For our athletes, the truth comes out on the dreaded film day with coach after the game. For your driving, cameras at intersections may provide the needed information. For networks, data plane analysis tells the story. Just as you know how a player performed, or just as you can see how you used an intersection while driving, you can determine how the data packets your network devices moved. Further, you can see many details about those packets. The data plane is where you get all the network statistics that everyone is familiar with. How much traffic is moving through the network? What applications are using the network? What users are using the network? Where is this traffic actually flowing on my network? Is this data flowing the way it was intended to flow when the environment was set up? Examine the data plane to find out.

Planes Data Examples

This section provides some examples of the data that you can see from the various planes. Table 3-1 shows common examples of management plane data.

Table 3-1 Management Plane Data Examples

Source	Data	What It Tells You	Example
Management plane command output	Product family	Broad category of device	Cisco Nexus 5500 Series switches
Management plane command output	Product identification	Exact device type	N5K_C5548P
Management plane command output	Physical type	Component type	Chassis
Management plane command output	Software version	Software version running on the component	5.1(3)N2(1)
Management plane configuration file	Configured routing protocol 1	A configuration entry for a routing protocol	Router OSPF x

Source	Data	What It Tells You	Example
Management plane command output	OSPF neighbors	Number of current OSPF neighbors configured	Neighbor x.x.x.x
Management plane configuration file	Configured routing protocol 2	A configuration entry for a routing protocol	Router BGP xxxxx
Management plane command output	Number of CPU cores	Data about the physical CPU configuration	8
Management plane command output	CPU utilization	CPU utilization at some point in time	30%
Management plane command output	Memory	Amount of memory in the device	16 GB
Management plane command output	Memory utilization	Amount of memory consumed given the current processes at some point in time	5 GB
Management plane command output	Interfaces	Number of interfaces in the device	50
Management plane command output	Interface utilization	Percentage of utilization of any given interface	45%
Management plane command output	Interface packet counters	Number of packets that have been forwarded by this interface since it was last cleared	1,222,333
Observed value or ask the town	Road surface	From the road analogy, describes the road surface	asphalt
Asking the player	Player weight	From the sports player analogy, describes the player	200 lb
Asking the player	Player 1 position 1	Describes the role of the player	Running-back player
Asking the player	Player 1 position 2	Describes another role of the player	Signal caller

In the last two rows of Table 3-1, note that the same player performs multiple functions: This player plays multiple positions on the same team. Similarly, single network devices perform multiple roles in a network and appear to be entirely different devices. A single cab driver can be part of many “going somewhere” instances. This also happens when you are using network device contexts. This is covered later in this chapter, in the section “A Wider Rabbit Hole.”

Notice that some of the management plane information (OSPF and packets) is *about* control plane and data plane information. This is still a “feature” because it is not communication (control plane) or actual packets (data plane) flowing through the device. This is

simply state information at any given point in time or features you can use as context in your analysis. This is information about the device, the configuration, or the traffic.

The control plane, where the communication between devices occurs, sets up the forwarding in the environment. This differs from management plane traffic, as it is communication between two or more entities used to set up the data plane forwarding. In most cases, these packets do not use the dedicated management interfaces of the devices but instead traverse the same data plane as the application overlay instances. This is useful for gathering information about the path during the communication activity. Control plane protocols examine speed, hop counts, latency, and other useful information as they traverse the data plane environments from sender to receiver. Dynamic path selection algorithms use these data points for choosing best paths in networks. Table 3-2 provides some examples of data plane traffic that is control plane related.

Table 3-2 *Control Plane Data Examples*

Source	Data	What It Tells You	Example
Captured traffic from the data plane	OSPF neighbor packets between two devices	Control plane communication between these two devices for an instance of OSPF	Router LSA (link-state advertisement) packets
Captured traffic from the data plane	BGP neighbor packets between two devices	Control plane communication between these two devices for an instance of BGP	BGP keepalives
Captured traffic from the data plane	Spanning-tree packets	Communication between neighboring devices to set up Layer 2 environments	Spanning-tree BPDUs (bridge protocol data units)
Municipality activity logs	Roads example stoplight system	Communications between intersection stoplights to ensure that all lights are never green at same time	Electronic communication that is not part of the data plane (the roads) but is part of the traffic system
Listening to the communications during an ongoing play	Sports player communication—football	Communications between the players to set up the environment	Play calls in a huddle or among players prior or during the play
Listening to the communications during an ongoing play	Same sports player communication—baseball	Communications between the players to set up the environment	Hand signals to fielders about what pitch is coming

The last two items in Table 3-2 are interesting in that the same player plays two sports! Recall from the management plane examples in Table 3-1 that the same device can perform multiple roles in a network segmentation scenario, as a single node or as multiple nodes split into virtual contexts. This means that they could also be participating in

multiple control planes, each of which may have different instructions for instances of data plane forwarding. A cab driver as part of many “going somewhere” instances has many separate and unrelated control plane communications throughout a typical day.

As you know, the control plane typically uses the same data plane paths as the data plane traffic. Network devices distinguish and prioritize known control plane protocols over other data plane traffic because correct path instruction is required for proper forwarding. Have you ever seen a situation in which one of the sports players in your favorite sport did not hear the play call? In such a case, the player does not know what is happening and does not know how to perform his or her role, and mistakes happen. The same type of thing can happen on a network, which is why networks prioritize these communications based on known packet types. Cisco also provides quality-of-service (QoS) mechanisms to allow this to be configurable for any custom “control plane protocols” you want to define that network devices do not already prioritize.

The data plane is the collection of overlay instance packets that move across the networks in your environment (including control plane communications). As discussed in Chapter 2, when you build an overlay analytics solution, all of the required components from your analytics infrastructure model comprise a single application instance within the data plane. When developing network analytics solutions, some of your data feeds from the left of the analytics infrastructure model may be reaching outside your application instance and back into the management plane of the same network. In addition, your solution may be receiving event data such as syslog data, as well as data and statistics about other applications running within the same data plane. For each of these applications, you need to gather data from some higher entity that has visibility into that application state or, more precisely, is communicating with the management plane of each of the applications to gather data about the application so that you can use that summary analysis in your solution. Table 3-3 provides some examples of data plane information.

Table 3-3 *Data Plane Data Examples*

Source	Data	What It Tells You	Example
Data plane packet capture	Your analytics streaming data packets between a data source and your data storage	Packets from a single application overlay instance	Packets from a network source you have set up to the receiver you have set up, such as a Kafka bus
Data plane packet capture	Your email application	Email packets, and email details inside the packets	Packets from your email server to all users with email clients
Data plane packet capture	Your music streaming services	A streaming music session outside the packets and where and what you are listening to inside the packets	Pandora or Amazon music session packets between your listening device and the service location

Source	Data	What It Tells You	Example
Data plane packet capture	Your browser session	Packets between you and the Internet for a single session (you may have several of these)	A single session between you and www.cisco.com
Data plane packet capture	Routing protocol session between two router devices	A data plane application overlay instance outside the packets (your control plane analysis is based on the data about and inside these packets)	An OSPF routing communications session between two of your core routers
Observing and recording the activity	Sports player 1 activity	Information about a single player performing the activity he/she has been instructed to do by the control plane communication	Running, throwing, blocking
Observing and recording the activity	Sports player 2 activity	Information about a second player performing the activity he/she has been instructed to do by the control plane communication	Running, throwing, blocking
Tracking your car along the path	Roads analogy 1	Information about you and your family using the roads system to go to work	Your car and driving activity on the various roads while going to work
Tracking your car along the path	Roads analogy 2	Information about you and your family using the roads system to go to the grocery store	Your car and driving activity on the various roads while going to the store
Data plane packet capture	Management plane for a network overlay	A session that uses your network data plane to reach inside an encapsulated network session	A Virtual Extensible LAN (VXLAN)-encapsulated virtual network instance running over your environment
Data plane packet capture	Control plane for a network overlay	A communications session between two network components, physical or virtual, that are tunneling through your networks	A session between virtual routers running in servers and using VXLAN encapsulation as part of an entire network “instance” running in your data plane

What are the last two items in Table 3-3? How are the management plane and somebody else's control plane showing up on your data plane? As indicated in the management and control plane examples, a single, multitasked player can play multiple roles side by side, just as a network device can have multiple roles, or contexts, and a cab driver can move many different people in the same day.

If you drill down into a single overlay instance, each of these roles may contain data plane communications that include the management, control, and data planes of other, virtualized instances. If your player is also a coach and has players of his own, then for his coaching role, he has entire instances of new players. Perhaps you have a management plane to your servers that have virtual networking as an application. Virtual network components within this application all have control plane communications for your virtual networks to set up a virtual data plane. This all exists within your original data plane. If the whole thing exists in the cloud, these last two are you.

Welcome to cloud networking. Each physical network typically has one management and control plane at the root. You can segment this physical network to adjacent networks where you treat them separately. You can virtualize instances of more networks over the same physical infrastructure or segment.

Within each of these adjacent networks, at the data plane, it is possible that one or more of the data plane overlays is a complete network in itself. Have you ever heard of Amazon Web Services (AWS), Azure, NFV, or VPC (Virtual Packet Core)? Each of these has its own management, control, and data planes related to the physical infrastructure but support creation of full network instances inside the data plane, using various encapsulation or tunneling mechanisms. Each of these networks also has its own planes of operation. Adjacent roles is analogous to a wider rabbit hole, and more instances of networks within each of them is analogous to a deeper rabbit hole.

A Wider Rabbit Hole

Prior to that last section, you understood the planes of data that are available to you, right? Ten years ago, you could have said yes. Today, with segmentation, virtualization, and container technology being prevalent in the industry, the answer may still be no. The rabbit hole goes much wider and much deeper. Let's first discuss the "wider" direction.

Consider your sports player again. Say that you have gone deep in understanding everything about him. You understand that he is a running back on a football team, and you know his height and weight. You trained him to run your special off-tackle plays again and again, based on some signal called out when the play starts (control plane). You have looked at films to find out how many times he has done it correctly (data plane). Excellent. You know all about your football player.

What if your athlete also plays baseball? What if your network devices are providing multiple independent networks? If you treat each of these separately, each will have its own set of management, control, and data planes. In sports, this is a multi-sport athlete. In networking, this is network virtualization. Using the same hardware and software to provide multiple, adjacent networks is like the same player playing multiple sports. Each of these

has its own set of data, as shown Figure 3-7. You can also split physical network devices into contexts at the hardware level, which is a different concept. (We would be taking the analogy too far if we compared this to a sports player with multiple personalities.)

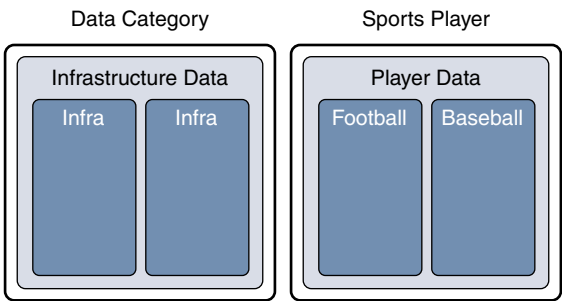


Figure 3-7 *Network Virtualization Compared to a Multisport Player*

In this example showing the network split into adjacent networks (via contexts and/or virtualization), now you need to have an entirely different management conversation about each. Your player’s management plane data about position and training for baseball is entirely different from his position and training in football. The control plane communications for each are unique to each sport. Data such as height and weight are not going to change. Your devices still have a core amount of memory, CPU, and capacity. The things you are going to measure at the player’s data plane, such as his performance, need to be measured in very different ways (yards versus pitches or at bats). Welcome to the world of virtualization of the same resource—using one thing to perform many different functions, each of which has its own management, control, and data planes (see Figure 3-8).

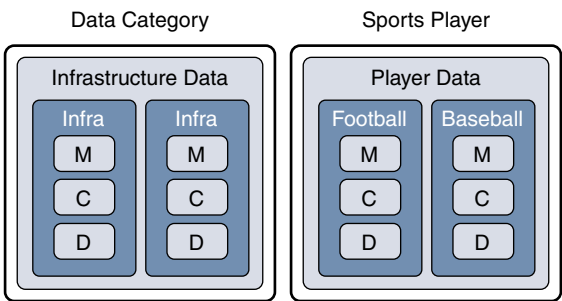


Figure 3-8 *Multiple Planes for Infrastructure and a Multisport Player*

This scenario can also be applied to device contexts for devices such as Cisco Nexus or ASA Firewall devices. Go a layer deeper: Virtualizing multiple independent networks within a device or context is called *network virtualization*. Alternatively, you can slice the same component into multiple “virtual” components or contexts, and each of these

components has an instance of the three necessary planes for operation. From a data perspective, this also means you must gather data that is relative to each of these environments. From a solutions perspective, this means you need to know how to associate this data with the proper environment. You need to keep all data from each of the environments in mind as you examine individual environments. Conversely, you must be aware of the environment(s) supported by a single hardware device if you wish to aggregate them all for analysis of the underlying hardware.

Most network components in your future will have the ability to perform multiple functions, and therefore there will often be a root management plane and many sub-management planes. Information at the root may be your sports player's name, age, height and weight, but there may be multiple management, control, and data planes per function for which your sports player or your network component performs. For each function, your sports player is part of a larger, spread-out network, such as a baseball team or a football team. Some older network devices do not support this; consider the roads analogy. It is nearly impossible to split up some roads for multiple purposes. Have you ever seen a parade that also has regular traffic using the same physical roads?

The ability to virtualize a component device into multiple other devices is common for cloud servers. For example, you might put software on a server that allows you to carve it into virtual machines or containers. You may have in your network Cisco Nexus switches that are deployed as contexts today. To a user, these contexts simply look like some device performing some services that are needed. As you just learned, you can use one physical device to provide multiple purposes, and each of these individual purposes has its own management, control, and data planes. Now recall the example from the data plane table (Table 3-3), where full management, control, and data planes exist within each of the data planes of these virtualized devices. The rabbit hole goes deeper, as discussed in the next section.

A Deeper Rabbit Hole

Have you ever seen the picture of a TV on a TV on a TV on a TV that appears to go on forever? Some networks seem to go to that type of depth.

You can create new environments entirely in software. The hardware management and control planes remain, but your new environment exists entirely within the data plane. This is the case with NFV and cloud networks, and it is also common in container, virtual machine, or microservices architectures. For a sports analogy to explain this, say that your athlete stopped playing and is now coaching sports. He still has all of his knowledge of both sports, as well as his own stats. Now he has multiple players playing for him, as shown in Figure 3-9, all of which he treats equally on his data plane activity of coaching.

Each of these players has his or her own set of data, too. There is a management plane to find out about the players, a communications plane where they communicate with their teammates, and a data plane to examine the players' activity and judge performance.

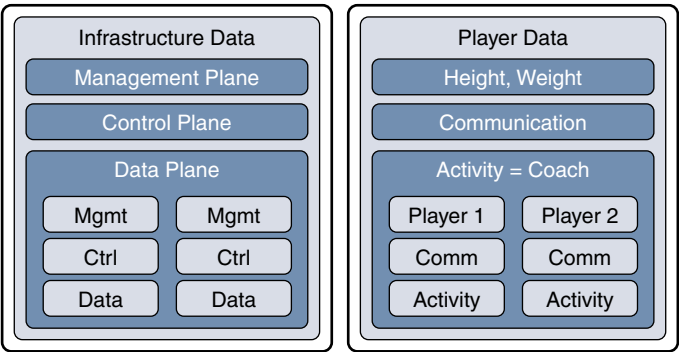


Figure 3-9 *Networks Within the Data Plane*

Figure 3-10 shows an example of an environment for NFV. You design virtual environments in these “pod” configurations such that you can add blocks of capacity as performance and scale requirements dictate. The NFV infrastructure exists entirely within the data plane of the physical environment because it exists within software, on servers on the right side of the diagram.

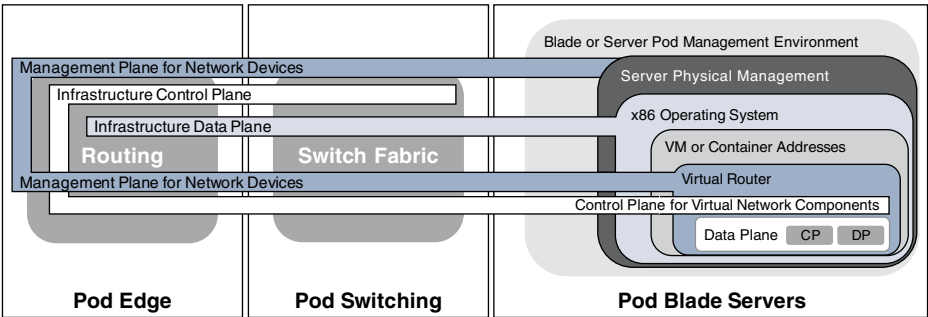


Figure 3-10 *Combining Planes Across Virtual and Physical Environments*

In order for the physical and virtual environments to function as a unit, you may need to extend the planes of operation. In this example, the pod is the coach, and each instance of an NFV function within the data plane environment is like another player on his team. Each team is a virtual network function that may have multiple components or players. NFV supports many different virtual network functions at the same time, just as your coach can coach multiple teams at the same time. Although rare, each of these virtual network functions may also have an additional control plane and data plane within the virtual data planes shown in Figure 3-10. Unless the virtual network function is providing an isolated, secure function, you connect this very deep control and data plane to the hybrid infrastructure planes. This is one server. As you saw in the earlier OpenStack example, these planes could extend to hundreds or thousands of servers.

Summary

At this point, you should understand the layers of abstraction and the associated data. Why is it important to understand the distinction? With the sports player, you determine the size, height, weight, role, and build of your player at the management plane; however, this reveals nothing about what the player communicates during his role. You learn that by watching his control plane. You analyze what network devices communicate to each other by watching the control plane activity between the devices.

Now let's move to the control plane. For your player, this is his current communication with his current team. If he is playing one sport, it is the on-field communications with his peers. However, if he is playing another sport as well, he has a completely separate instance that is a different set of control plane communications. Both sports have a data plane of the "activity" that may differ. You can virtualize network devices and entire networks into multiple instances—just like a multisport player and just as in the NFV example. Each of your application overlays could have a control plane, such as your analytics solution requesting traffic from a data warehouse.

If your player activity is "coaching," he has multiple players who each has his own management, control, and data planes with which he needs to interact so they have a cohesive operation. If he is coaching multiple teams, the context of each of the management, control, and data planes may be different within each team, just as different virtual network functions in an NFV environment may perform different functions. Within each slice (team), this coach has multiple players, just as a network has multiple environments within each slice, each of which has its own management, control, and data planes. If your network is "hosting," then the same concepts apply.

Chapter 4, "Accessing Data from Network Components," discusses how to get data from network components. Now you know that you must ensure that your data analysis is context aware, deep down into the layers of segmentation and virtualization. Why do you care about these layers? Perhaps you have implemented something in the cloud, and you wish to analyze it. Your cloud provider is like the coach, and that provider has its own management, control, and data planes, which you will never see. You are simply one of the provider's players on one of its teams (maybe team "Datacenter East"). You are an application running inside the data plane of the cloud provider, much like a Little League player for your sports coach. Your concern is your own management (about your virtual machines/containers), control (how they talk to each other), and data planes (what data you are moving among the virtual machines/containers). Now you can add context.

This page intentionally left blank

Accessing Data from Network Components

This chapter dives deep into data. It explores the methods available for extracting data from network devices and then examines the types of data used in analytics. In this chapter you can use your knowledge of planes from Chapter 3, “Understanding Networking Data Sources,” to decode the proper plane of operation as it relates to your environment. The chapter closes with a short section about transport methods for bringing that data to a central location for analysis.

Methods of Networking Data Access

This book does not spend much time on building the “big data engine” of the analytics process, but you do need to feed it gas and keep it oiled—with data—so that it can drive your analytics solutions. Maybe you will get lucky, and someone will hand you a completely cleaned and prepared data set. Then you can pull out your trusty data science books, apply models, and become famous for what you have created. Statistically speaking, finding clean and prepared data sets is an anomaly. Almost certainly you will have to determine how to extract data from the planes discussed in Chapter 3. This chapter discusses some of the common methods and formats that will get you most of the way there. Depending on your specific IT environment, you will most likely need to fine-tune and be selective about the data acquisition process.

As noted in Chapter 3, you obtain a large amount of data from the management plane of each of your networks. Many network components communicate to the outside world as a secondary function (the primary function is moving data plane packets through the network), through some specialized interface for doing this, such as an out-of-band management connection. *Out-of-band (OOB)* simply means that no data plane traffic will use the interface—only management plane traffic, and sometimes control plane traffic, depending on the vendor implementation. You need device access to get data from devices.

While data request methods are well known, “pulling” or “asking” for device data are not your only options. You can “push” data from a device on-demand, by triggering it, or on a schedule (for example, event logging and telemetry). You receive push data at a centralized location such as a syslog server or telemetry receiver, where you collect information from many devices. Why are we seeing a trend toward push rather than pull data? For each pull data stream, you must establish a network connection, including multiple exchanges of connection information, before you ask the management plane for the information you need. If you already know what you need, then why not just tell the management plane to send it to you on a schedule? You can avoid the niceties and protocol handshakes by using push data mechanisms, if they are available for your purpose.

Telemetry data is push data, much like the data provided by a heart rate monitor. Imagine that a doctor has to come into a room, establish rapport with the patient, and then take the patient’s pulse. This process is very inefficient if it must happen every 5 minutes. You would get quite annoyed if the doctor asked the same opening questions every 5 minutes upon coming into the room. A more efficient process would be to have a heart rate monitor set up to “send” (display in this case) the heart rate to a heart rate monitor. Then, the doctor could avoid the entire “Hi, how are you?” exchange and just get the data needed where it is handy. This is telemetry. Pull data is still necessary sometimes, though, as when a doctor needs to ask about a specific condition.

For data plane analysis, you use the management plane to gain information about the data flows. Tools such as NetFlow and IP Flow Information Export (IPFIX) provide very valuable summary data plane statistics to describe the data packets forwarded through the device. These tools efficiently describe what is flowing over the environment but are often sampled, so full granularity of data plane traffic may not be available, especially in high-speed environments.

If you are using deep packet inspection (DPI) or some other analysis that requires a look into the protocol level of the network packets, you need a dedicated device to capture these packets. Unless the forwarding device has onboard capturing capability, full packet data is often captured, stored, and summarized by some specialized data plane analysis device. This device captures data plane traffic and dissects it. Solutions such as NetFlow and IPFIX only go to a certain depth in packet data.

Finally, consider adding aggregate, composite, or derived data points where they can add quality to your analysis. Data points are atomic, and by themselves they may not represent the state of a system well. When you are collecting networking data points about a system whose state is known, you end up with a set of data points that represents a known state. This in itself is very valuable in networking as well as analytics. If you compare this to human health, a collection of data points such as your temperature, blood pressure, weight, and cholesterol counts is a group that in itself may indicate a general condition of healthy or not. Perhaps your temperature is high and you are sweating and nauseated, have achy joints, and are coughing. All of these data points together indicate some known condition, while any of them alone, such as sweating, would not be exactly predictive. So when considering the data, don’t be afraid to put on your subject matter expert (SME) hat and enter a new, known-to-you-only data point along the way, such as “has bug X,” “is crashed,” or “is lightly used.” These points provide valuable context for future analysis.

The following sections go through some common examples of data access methods to help you understand how to use each of them for gathering data. As you drill down into virtual environments, consider the available data collection options and the performance impact that each will have given the relative location in the environment. For example, a large physical router with hardware capacity built in for collecting NetFlow data exhibits much less performance degradation than a software-only instance of a router configured with the same collection. You can examine a deeper virtual environments by capturing data plane traffic and stripping off tunnel headers that associate the packets to the proper virtualized environment.

Pull Data Availability

This section discusses available methods for pulling data from devices by asking questions of the management plane. Each of these methods has specific strength areas, and these methods underpin many products and commercially available packages that provide services such as performance management, performance monitoring, configuration management, fault detection, and security. You probably have some of them in place already and can use them for data acquisition.

SNMP

Simple Network Management Protocol (SNMP), a simple collection mechanism that has been around for years, can be used to provide data about any of the planes of operation. The data is available only if there is something written into the component software to collect and store the data in a Management Information Base (MIB). If you want to collect and use SNMP data and the device has an SNMP agent, you should research the supported MIBs for the components from which you need to collect the data, as shown in Figure 4-1.

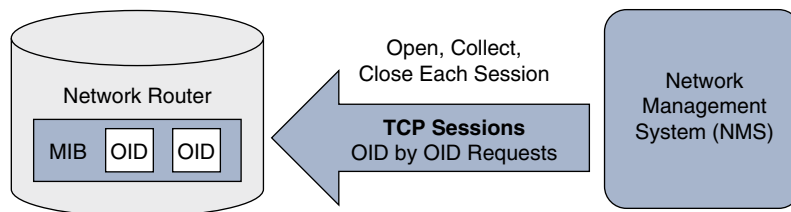


Figure 4-1 *SNMP Data Collection*

SNMP is a connection-oriented client/server architecture in which a network component is polled for a specific question for which it is known to have the answer (a MIB object exists that can provide the required data). There are far too many MIBs available to provide a list here, but Cisco provides a MIB locator tool you can use to find out exactly which data points are available for polling: <http://mibs.cloudapps.cisco.com/ITDIT/MIBS/MainServlet>.

Consider the following when using SNMP and polling MIBs:

- SNMP is standardized and widely available for most devices produced by major vendors, and you can use common tools to extract data from multivendor networks.
- MIBs are data tables of object identifiers (OIDs) that are stored on a device, and you can access them by using the SNMPv1, SNMPv2, or SNMPv3 mechanism, as supported by the device and software that you are using. Data that you would like for your analysis may not be available using SNMP. Research is required.
- OIDs are typically point-in-time values or current states. Therefore, if trending over time is required, you should use SNMP polling systems to collect the data at specific time intervals and store it in time series databases.
- Newer SNMP versions provide additional capabilities and enhanced security. SNMPv1 is very insecure, SNMPv2 added security measures, and SNMPv3 has been significantly hardened. SNMPv2 is common today.
- Because SNMP is statically defined by the available MIBs and sometimes has significant overhead, it is not well suited to dynamic machine-to-machine (M2M) communications. Other protocols have been developed for M2M use.
- Each time you establish a connection to a device for a polling session, you need to first establish the connection and then request specific OIDs by using network management system (NMS) software.
- Some SNMP data counters clear on poll, so be sure to research what you are polling and how it behaves. Perform specific data manipulation on the collector side to ensure that the data is right for analysis.
- Some SNMP counters “roll over”; for example, 32-bit counters on very large interfaces max out at 4294967295. 64-bit counters ($2^{64}-1$) extend to numbers as high as 18446744073709551615. If you are tracking delta values (which change from poll to poll), this rollover can appear to be negative numbers in your data.
- Updating of the data in data tables you are polling is highly dependent on how the device software is designed in terms of MIB update. Well-designed systems are very near real-time, but some systems may update internal tables only every minute or so. Polling 5-second intervals for a table that updates every minute is just a waste of collection resources.
- There will be some level of standard data available for “discovery” about the device’s details in a public MIB if the SNMP session is authenticated and properly established.
- There are public and private (vendor-specific) MIBs. There is a much deeper second level of OIDs available from the vendor for devices that are supported by the NMS. This means the device MIB is known to the NMS, and vendor-specific MIBs and OIDs are available.
- Periodic SNMP collections are used to build a model of the device, the control plane configuration, and the data plane forwarding environment. SNMP does not perform data plane packet captures.

There are many SNMP collectors available today, and almost every NMS has the capability to collect available SNMP data from network devices. For the router memory example from Chapter 3, the SNMP MIB that contains the memory OID that reports memory utilization is polled.

If you want data about something where there is no MIB, you need to find another way to get the data. For example, say that your sports player from Chapter 3 has been given a list of prepared questions prior to an interview, and you can only ask questions from the prepared sheet. If you ask a question outside of the prepared sheet, you just get a blank stare. This is like trying to poll a MIB that does not exist. So what can you do?

CLI Scraping

If you find the data that you want by running a command on a device, then it is available to you with some creative programming. If the data is not available using SNMP or any other mechanisms, the old standby is command-line interface (CLI) scraping. It may sound fancy, but CLI scraping is simply connecting to a device with a connection client such as Telnet or Secure Shell (SSH), capturing the output of the command that contains your data, and using software to extract the values that you want from the output provided. For the router memory example, if you don't have SNMP data available you can scrape the values from periodic collections of the following command for your analysis:

```
Router#show proc mem
Processor Pool Total: 766521544 Used: 108197380 Free: 658324164
I/O Pool Total: 54525952 Used: 23962960 Free: 30562992
```

While CLI scraping seems like an easy way to ensure that you get anything you want, there are pros and cons. Some key factors to consider when using CLI scraping include the following:

- The overhead is even higher for CLI scraping than for SNMP. A connection must be established, the proper context or prompt on the device must be established, and the command or group of commands must be pulled.
- Once you pull the commands, you must write a software parser to extract the desired values from the text. These parsers often include some complex regular expressions and programming.
- For commands that have device-specific or network-specific parameters, such as IP addresses or host names, the regular expressions must account for varying length values while still capturing everything else in the scrape.
- If there are errors in the command output, the parser may not know how to handle them, and empty or garbage values may result.
- If there are changes in the output across component versions, you need to update or write a new parser.
- It may be impossible to capture quality data if the screen is dynamically updating any values by refreshing and redrawing constantly.

YANG and NETCONF

YANG (Yet Another Next Generation) is an evolving alternative to SNMP MIBs that is used for many high-volume network operations tasks. YANG is defined in RFC 6020 (<https://tools.ietf.org/html/rfc6020>) as a data modeling language used to model configuration and state data. This data is manipulated by the Network Configuration Protocol (NETCONF), defined in RFC 6241 (<https://tools.ietf.org/html/rfc6241>)

Like SNMP MIBs, YANG models must be defined and available on a network device. If a model exists, then there is a defined set of data that can be polled or manipulated with NETCONF remote procedure calls (RPCs). Keep in mind a few other key points about YANG:

- YANG is the model on the device (such as an SNMP MIB), and NETCONF is the mechanism to poll and manipulate the YANG models (for example, to get data).
- YANG is extensible and modular, and it provides additional flexibility and capability over legacy SNMP.
- NETCONF/YANG performs many configuration tasks that are difficult or impossible with SNMP.
- NETCONF/YANG supports many new paradigms in network operations, such as the distinction between configuration (management plane) and operation (control plane) and the distinction between creating configurations and applying these configurations as modifications.
- You can use NETCONF/YANG to provide both configuration and operational data that you can use for model building.
- RESTCONF (<https://tools.ietf.org/html/rfc8040>) is a Representational State Transfer (REST) interface that can be reached through HTTP for accessing data defined in YANG using data stores defined in NETCONF.

YANG and NETCONF are being very actively developed, and there are many more capabilities beyond those mentioned here. The key points here are in the context of acquiring data for analysis.

NETCONF and YANG provide configuration and management of operating networks at scale, and they are increasingly common in full-service assurance systems. For your purpose of extracting data, NETCONF/YANG represents another mechanism to extract data from network devices, if there are available YANG models.

Unconventional Data Sources

This section lists some additional ways to find more network devices or to learn more about existing devices. Some protocols, such as Cisco Discovery Protocol (CDP), often send identifying information to neighboring devices, and you can capture this information from those devices. Other discovery mechanisms provided here aid in identifying all devices on a network. The following are some unconventional data sources you need to know about:

<https://t.me/learningnets>

- Link Layer Discovery Protocol (LLDP) is an industry standard protocol for device discovery. Devices communicate to other devices over connected links. If you do not have both devices in your data, LLDP can help you find out more about missing devices.
- You can use an Address Resolution Protocol (ARP) cache of devices that you already have. ARP maps hardware MAC addresses to IP addresses in network participants that communicate using IP. Can you account for all of the IP entries in your “known” data sets?
- You can examine MAC table entries from devices that you already have. If you are capturing and reconciling MAC addresses per platform, can you account for all MAC addresses in your network? This can be a bit challenging, as every device must have a physical layer address, so there could be a large number of MAC addresses associated to devices that you do not care about. Virtualization environments set up with default values may end up producing duplicate MAC addresses in different parts of the network, so be aware.
- Windows Management Instrumentation (WMI) for Microsoft Windows servers provides data about the server infrastructure.
- A simple ping sweep of the management address space may uncover devices that you need to use in your analysis if your management IP space is well designed.
- Routing protocols such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), and Enhanced Interior Gateway Routing Protocol (EIGRP) have participating neighbors that are usually defined within the configuration or in a database stored on the device. You can access the configuration or database to find unknown devices.
- Many devices today have REST application programming interface (API) instrumentation, which may have some mechanism for requesting the available data to be delivered by the API. Depending on the implementation of the API, device and neighbor device data may be available. If you are polling a controller for a software-defined networking (SDN) environment, you may find a wealth of information by using APIs.
- In Linux servers used for virtualization and cloud building, there are many commands to scrape. Check your operating system with `cat /etc/*release` to see what you have, and then search the Internet to find what you need for that operating system.

Push Data Availability

This section describes push capability that enables a device to tell you what is happening. You can configure push data capability on the individual components or on interim systems that you build to do pull collection for you.

SNMP Traps

In addition to the client server polling method, SNMP also offers some rudimentary event notification, in the form of SNMP traps, as shown in Figure 4-2.

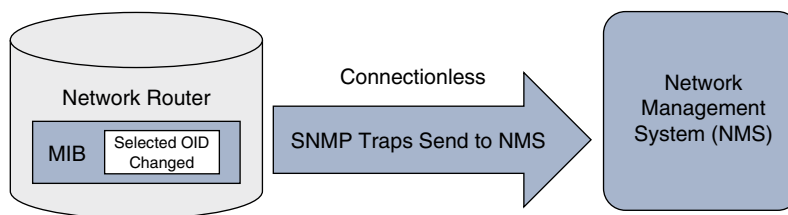


Figure 4-2 *SNMP Traps Architecture*

The number of available traps is limited. Even so, using traps allows you to be notified of a change in a MIB OID value. For example, a trap can be generated and sent if a connected interface goes down (that is, if the data plane is broken) or if there is a change in a routing protocol (that is, there is a control plane problem). Most NMSs also receive SNMP traps. Some OID values are numbers and counters, and many others are descriptive and do not change often. Traps are useful in this latter case.

Syslog

Most network and server devices today support syslog capability, where system-, program-, or process-level messages are generated by the device. Figure 4-3 shows a syslog example from a network router.

```

*Jan 2 14:55:07.631: c3600_scp_set_dstaddr2_idb(184)add = 80 name is Embedded-Service-Engine0/0
*Jan 2 14:55:08.935: %VPN_HW-6-INFO_LOC: Crypto engine: onboard 0 State changed to: Initialized
*Jan 2 14:55:09.111: %VPN_HW-6-INFO_LOC: Crypto engine: onboard 0 State changed to: Enabled
*Jan 2 14:55:16.547: %PVM-6-UNSUPPORTED: Codec G.723 is not supported on PVM3.
*Jan 2 14:55:30.071: %MAINBOARD_SFP-6-INSERTED: Transceiver module 1000BaseSX SFP is inserted in GigabitEthernet0/1 port.
*Jan 2 14:55:30.147: %MAINBOARD_SFP-6-INSERTED: Transceiver module 1000BaseSX SFP is inserted in GigabitEthernet0/2 port.
*Jan 2 14:55:30.147: %LINEPROTO-5-UPDOWN: Line protocol on Interface VoIP-Null0, changed state to up
*Jan 2 14:55:30.147: %LINEPROTO-5-UPDOWN: Line protocol on Interface LI-Null0, changed state to up
*Jan 2 14:55:30.147: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed state to up
*Jan 2 14:55:30.147: %LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to up
*Jan 2 14:55:30.147: %LINK-3-UPDOWN: Interface GigabitEthernet0/2, changed state to down
*Jan 2 14:55:31.147: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up
*Jan 2 14:55:31.147: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up
*Jan 2 14:55:31.147: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/2, changed state to down
  
```

Figure 4-3 *Syslog Data Example*

Syslog messages are stored locally for troubleshooting purposes, but most network components have the additional capability built in (or readily available in a software package) to send these messages off-box to a centralized syslog server. This is a rich source of network intelligence, and many analysis platforms can analyze this type of data to a very deep level. Common push logging capabilities include the following:

- Network and server syslogs generally follow a standardized format, and many facilities are available for storing and analyzing syslogs. Event message severities range from detailed debug information to emergency level.
- Servers such as Cisco Unified Computing System (UCS) typically have system event logs (SELs), which detail the system hardware activities in a very granular way.

- Server operating systems such as Windows or Linux have detailed logs to describe the activities of the operating system processes. There are often multiple log files if the server is performing many activities.
- If the server is virtualized, or sliced, there may be log files associated with each slice, or each virtual component, such as virtual machines or containers.
- Each of these virtual machines or containers may have log files inside that are used for different purposes than the outside system logs.
- Software running on the servers typically has its own associated log files describing the activities of the software package. These packages may use the system log file or a dedicated log file, or they may have multiple log files for each of the various activities that the software performs.
- Virtualized network devices often have two logs each. A system may have a log that is about building and operating the virtualized router or switch, while the virtualized device (recall a player on the coach's team?) has its own internal syslog mechanism (refer to the first bullet in this list).

Note that some components log by default, and others require that you explicitly enable logging. Be sure to check your components and enable logging as a data source. Logging is asynchronous, and if nothing is happening, then sometimes no logs are produced. Do not confuse this with logs that are not making it to you or logs that cannot be sent off a device due to a failure condition. For this purpose, and for higher-value analytics, have some type of periodic log enabled that always produces data. You can use this as a logging system “test canary.”

Telemetry

Telemetry, shown in Figure 4-4, is a newer push mechanism whereby network components periodically send specific data feeds to specific telemetry receivers in the network. You source telemetry sessions from the network device rather than poll with NMS. There can be multiple telemetry events, as shown in Figure 4-4. Telemetry sessions may be configured on the router, or the receiver may configure the router to send specific data on a defined schedule; either way, all data is pushed.

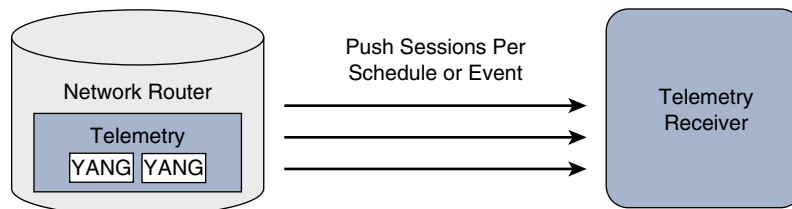


Figure 4-4 *Telemetry Architecture Example*

Like a heart rate monitor that checks pulse constantly, as in the earlier doctor example, telemetry is about sending data from a component to an external analysis system.

Telemetry capabilities include the following:

- Telemetry on Cisco routers can be configured to send the value of individual counters in 1-second intervals, if desired, to create a very granular data set with a time component.
- Much as with SNMP MIBs, a YANG-formatted model must exist for the device so that the proper telemetry data points are identified.
- You can play back telemetry data to see the state of the device at some point in the past. Analytics models use this with time series analysis to create predictive models.
- Model-driven telemetry (MDT) is a standardized mechanism by which common YANG models are developed and published, much as with SNMP MIBs. Telemetry uses these model elements to select what data to push on a periodic schedule.
- Event-driven telemetry (EDT) is a method by which telemetry data is sent only when some change in a value is detected (for example, if you want to know when there is a change in the up/down state of an interface in a critical router). You can collect the interface states of all interfaces each second, or you can use EDT to notify you of changes.
- Telemetry has a “dial-out” configuration option, with which the router initiates the connection pipe to the centralized capture environment. The management interface and interim firewall security do not need to be opened to the router to enable this capability.
- Telemetry also has a “dial-in” configuration option, with which the device listens for instructions from the central environment about the data streams and schedules for those data streams to be sent to a specific receiver.
- Because you use telemetry to produce steady streams of data, it allows you to use many common and standard streaming analytics platforms to provide very detailed analysis and insights.
- When using telemetry, although counters can be configured as low as 1 second, you should learn the refresh rate of the underlying table to maximize efficiency in the environment. If the underlying data table is updated by the operating system only every 1 minute, polling every 5 seconds has no value.

For networks, telemetry is superior to SNMP in many regards, and where it can be used as a replacement, it reduces the overhead for your data collection. The downside is that it is not nearly as pervasive as SNMP, and the required YANG-based telemetry models are not yet as readily available as are many common MIBs.

Make sure that every standard data source in your environment has a detailed evaluation and design completed for the deployment phase so that you know what you have to work with and how to collect and make it available. Recall that repeatable and reusable components (data pipelines) are a primary reason for taking an architecture approach to analytics and using a simple model like the analytics infrastructure model.

NetFlow

NetFlow, shown in Figure 4-5, was developed to capture data about the traffic flows on a network and is well suited for capturing data plane IPv4 and IPv6 flow statistics.

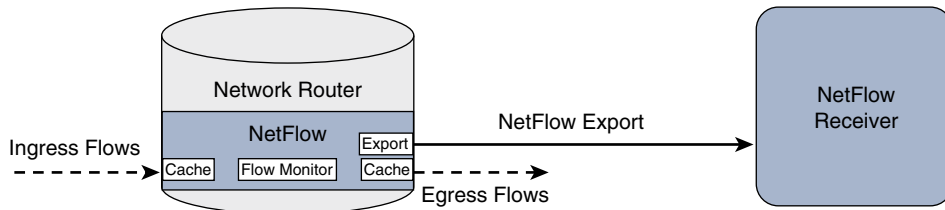


Figure 4-5 *NetFlow Architecture Example*

NetFlow is a very useful management plane method for data plane analysis in that NetFlow captures provide very detailed data about the actual application and control plane traffic that is flowing in and out of the connections between the devices on the network. NetFlow is heavily used for data plane statistics because of the rich set of data that is learned from the network packets as they are being forwarded through the device. An IPv4 or IPv6 data packet on a computer network has many fields from which to collect data, and NetFlow supports many of them. Some examples of the packet details are available later in this chapter, in the “Packet Data” section. Some important characteristics of NetFlow include the following:

- A minimum flow in IP terminology is the *5-tuple*—the sender, the sending port, the receiver, the receiving port, and the protocol used to encapsulate the data. This is the minimum NetFlow collection and was used in the earliest versions of NetFlow.
- Over the years, additional fields were added to subsequent versions of NetFlow, and predominant versions of NetFlow today are v5 and v9. NetFlow now allows you to capture dozens of fields.
- NetFlow v5 has a standardized list of more than a dozen fields and is heavily used because it is widely available in most Cisco routers on the Internet today.
- NetFlow v9, called Flexible NetFlow, has specific field selection within the standard that can be captured while unwanted fields are ignored.
- NetFlow capture is often unidirectional on network devices. If you want a full description of a flow, you can capture packet statistics in both directions between packet sender and receiver and associate them at the collector.
- NetFlow captures data about the traffic flows, and not the actual traffic that is flowing. NetFlow does not capture the actual packets.
- Many security products, including Cisco Stealthwatch, make extensive use of NetFlow statistics.

- NetFlow is used to capture all traffic statistics if the volume is low, or it can sample traffic in high-volume environments if capturing statistics about every packet would cause a performance impact.
- NetFlow by definition captures the statistics on the network device into NetFlow records, and a NetFlow export mechanism bundles up sets of statistics to send to a NetFlow collector.
- NetFlow exports the flow statistics when flows are finished or when an aging timer triggers the capture of data flows as aging time expires.
- NetFlow sends exports to NetFlow collectors, which are dedicated appliances for receiving NetFlow statistics from many devices.
- Deduplication and stitching together of flow information across network device information is important in the collector function so that you can analyze a single flow across the entire environment. If you collect data from two devices in the same application overlay path, you will see the same sessions on both of them.
- Cloud providers may have specific implementations of flow collection that you can use. Check with your provider to see what is available to you.

NetFlow v5 and v9 are Cisco specific, but IPFIX is a standards-based approach used by multiple vendors to perform the same flexible flow collection.

IPFIX

IP Flow Information Export (IPFIX) is a standard created by the IETF (Internet Engineering Task Force) that provides a NetFlow-alternative flow capture mechanism for Cisco and non-Cisco network devices. IPFIX is closely related to NetFlow as the original standard was based on NetFlow v9, so the architecture is generally the same. The latest IPFIX version is often referred to as NetFlow v10, and Cisco supports IPFIX as well. Some capabilities of IPFIX, in addition to those of NetFlow, include the following:

- IPFIX includes syslog information in a semi-structured format. By default, syslog information is sent as unstructured text in the push mechanism described earlier in this chapter.
- IPFIX includes SNMP MIB OIDs in the exports.
- IPFIX has a vendor ID field that a vendor can use for anything.
- Because IPFIX integrates extra data, it allows for some variable-length fields, while NetFlow has only fixed-length fields.
- IPFIX uses templates to tell the collector how to decode the fields in the updates, and these templates can be custom defined; in NetFlow, the format is fixed, depending on the NetFlow version.
- Templates can be crowdsourced and shared across customers using public repositories.

When choosing between NetFlow and IPFIX, consider the granularity of your data requirements. Basic NetFlow with standardized templates may be enough if you do not require customization.

sFlow

sFlow is a NetFlow alternative that samples network packets. sFlow offers many of the same types of statistics as NetFlow but differs in a few ways:

- sFlow involves sampled data by definition, so only a subset of the packet statistics are analyzed. Flow statistics are based on these samples and may differ greatly from NetFlow or IPFIX statistics.
- sFlow supports more types of protocols, including older protocols such as IPX, than NetFlow or IPFIX.
- As with NetFlow, much of the setup is often related to getting the records according to the configurable sampling interval and exporting them off the network device and loaded into the data layer in a normalized way.
- sFlow is built into many forwarding application-specific integrated circuits (ASICs) and provides minimal central processing unit (CPU) impact, even for high-volume traffic loads.

Most signs indicate that IPFIX is a suitable replacement for sFlow, and there may not be much further development on sFlow.

Control Plane Data

The control plane “configuration intent” is located by interacting with the management plane, while “activity traffic” is usually found within the data plane traffic. Device-level reporting from the last section (for example, telemetry, NetFlow, or syslog reporting) also provides data about control plane activity. What is the distinction between control plane analysis using management plane traffic and using data plane traffic? Figure 4-6 again shows the example network examined in Chapter 3.

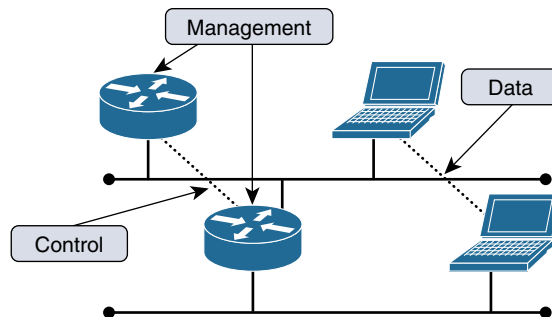


Figure 4-6 Sample Network Control Plane Example

Consider examining two network devices that should have a “relationship” between them, using a routing relationship as an example. Say that you determine through management plane polling of configuration items that the two routers are configured to be neighbors to each other. You may be able to use event logs to see that they indeed established a neighbor relationship because the event logging system was set up to log such activities.

However, how do you know that the neighbor relationship is always up? Is it up right now? Configuration shows the intent to be up, and event logs tell you when the relationship came up and when it went down. Say that the last logs you saw indicated that the relationship came up. What if messages indicating that the relationship went down were lost before they got to your analysis system?

You can validate this control plane intent by examining data plane traffic found on the wire between these two entities. (“On the wire” is analogous to capturing packets or packet statistics.) You can use this traffic to determine if regular keepalives, part of the routing protocol, are flowing at expected intervals. This analysis shows two-way communication and successful partnership of these routers. After you have checked configuration, confirmed with event logs, and validated with traffic from the wire, you can rest assured that your intended configuration for these devices to be neighbors was realized.

Data Plane Traffic Capture

If you really want to understand what is using your networks and NetFlow and IPFIX do not provide the required level of detail, packet inspection on captured packets may be your only option. You perform this function on dedicated packet analysis devices, on individual security devices, or within fully distributed packet analysis environments.

For packet capture on servers (if you are collecting traffic from virtualized environments and don’t have a network traffic capture option), there are a few good options for capturing all packets or filtering sets of packets from one or more interfaces on the device.

- NTOP (<https://www.ntop.org>) is software that runs on servers and provides a NetFlow agent, as well as full packet capture capabilities.
- Wireshark (<https://www.wireshark.org>) is a popular on-box packet capture tool and analyzer that works on many operating systems. Packet data sets are generated using standard filters.
- tcpdump (<https://www.tcpdump.org>) is a command-line packet capture tool available on most UNIX and Linux systems.
- Azure Cloud has a service called Network Watcher (<https://azure.microsoft.com/en-us/services/network-watcher/>).

You can export files from servers by using a software script if historical batches are required for model building. You can perform real-time analysis and troubleshooting on the server, and you can also save files for offline analysis on your own environment.

On the network side, capturing the massive amounts of full packet data that are flowing through routers and switches typically involves a two-step process. First, the device must

be explicitly configured to send a copy of the traffic to a specific interface or location (if the capture device is not in line with the typical data plane). Second, there must be a receiver capability ready to receive, store, and analyze that data. This is often part of an existing big data cluster as packet capture data can be quite large. The following sections describe some methods for sending packet data from network components.

Port Mirroring and SPAN

Port mirroring is a method of identifying the traffic to capture, such as from an interface or a VLAN, and mirroring that traffic to another port on the same device. Mirroring means that you have the device create another copy of the selected traffic. Traffic that enters or leaves VLANs or ports on a switch can use Switched Port Analyzer (SPAN).

RSPAN

Remote SPAN (RSPAN) provides the ability to define a special VLAN to capture and copy traffic from multiple switches in an environment to that VLAN. At some specified location, the traffic is copied to a physical switch port, which is connected to a network analyzer.

ERSPAN

Encapsulated Remote Switched Port Analyzer (ERSPAN) uses tunneling to take the captured traffic copy to an IP addressable location in the network, such as the interface of a packet capture appliance, or your machine.

TAPs

A very common way to capture network traffic is through the use of passive network terminal access points (TAPs), which are minimum three-port devices that are put between network components to capture packets. Two ports simply provide the in and out, and the third port (or more) is used for mirroring the traffic to a packet capture appliance.

Inline Security Appliances

In some environments, it is possible to have a dedicated security appliance in the traffic path. Such a device acts as a Layer 2 transparent bridge or as a Layer 3 gateway. An example is a firewall that is inspecting every packet already.

Virtual Switch Options

In virtualized environments, virtual switches or other forms of container networking exist inside each of the servers used to build the virtualized environments. For any traffic leaving the host and entering another host, it is possible to capture that traffic at the network layer.

However, sometimes the traffic leaves one container or virtual machine and enters another container or virtual machine within the same server host using local virtual switching, and the traffic is not available outside the single server. In many cases, capturing the data from virtual switches is not possible due to the performance implications on

virtual switching, but in some cases, this is possible if there is a packet analysis virtual machine on the same device. Following are some examples of known capabilities for capturing packet data inside servers:

- Hyper-V provides port mirroring capabilities if you can install a virtual machine on the same device and install capture software such as Wireshark. You can go to the virtual machine from which you want to monitor the traffic and configure it to mirror the traffic.
- For a VMware standard vSwitch, you can make an entire port group promiscuous, and a virtual machine on the same machine receives the traffic, as in the Hyper-V example. This essentially turns the vSwitch into a hub, so other hosts are receiving (and most are dropping) the traffic. This clearly has performance implications.
- For a VMware distributed switch, one option is to configure a distributed port mirroring session to mirror the virtual machine traffic from one virtual machine to another virtual machine on the same distributed switch.
- A VMware distributed switch also has RSPAN capability. You can mirror traffic to a network RSPAN VLAN as described previously and then dump the traffic to a packet analyzer connected to the network where the RSPAN VLAN is sent out a physical switch port. Layer 2 connectivity is required.
- A VMware distributed switch also has ERSPAN capability. You can send the encapsulated traffic to a remote IP destination for monitoring. The analysis software on the receiver, such as Wireshark, recognizes ERSPAN encapsulation and removes the outer encapsulation layer, and the resulting traffic is analyzed.
- It is possible to capture traffic from one virtual machine to another virtual machine on a local Open vSwitch switch. To do this, you install a new Open vSwitch switch, add a second interface to a virtual machine, and bridge a generic routing encapsulation (GRE) session, much as with ERSPAN, to send the traffic to the other host. Or you can configure a dedicated mirror interface to see the traffic at Layer 2.

Only the common methods are listed here. Because you do this capture in software, other methods are sure to evolve and become commonplace in this space.

Packet Data

You can get packet statistics from flow-based collectors such as NetFlow and IPFIX. These technologies provide the capability to capture data about most fields in the packet headers. For example, an IPv4 network packet flowing over an Ethernet network has the simple structure shown in Figure 4-7.



Figure 4-7 IPv4 Packet Format

Not too bad, right? If you expand the IP header, you can see that it provides a wealth of information, with a number of possible values, as shown in Figure 4-8.

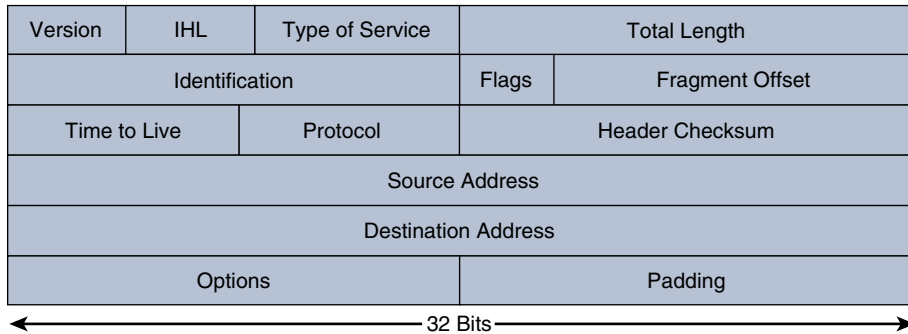


Figure 4-8 *Detailed IPv4 Packet Format*

NetFlow and IPFIX capture data from these fields. And you can go even deeper into a packet and capture information about the Transmission Control Protocol (TCP) portion of the packet, which has its own header, as shown in Figure 4-9.

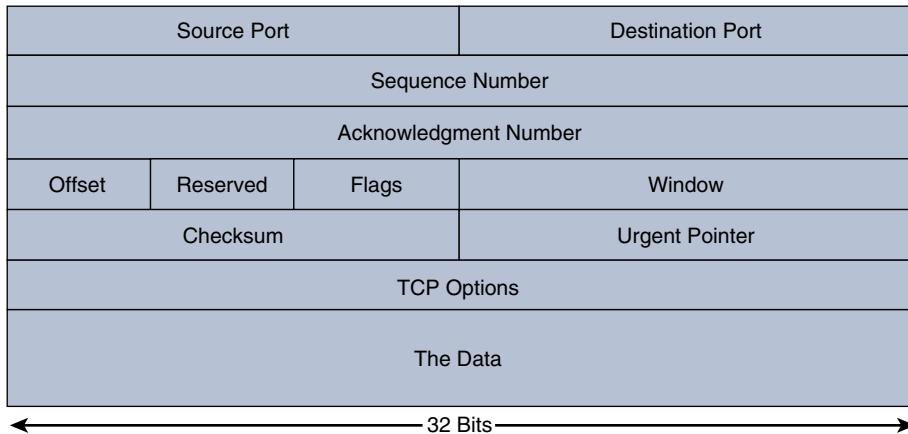


Figure 4-9 *TCP Packet Format*

Finally, if the data portion of the packet is exposed, you can gather more details from there, such as the protocols in the payload. An example of Hypertext Transfer Protocol (HTTP) that you can get from a Wireshark packet analyzer is shown in Figure 4-10. Note that it shows the IPv4 section, the TCP section, and the HTTP section of the packet.

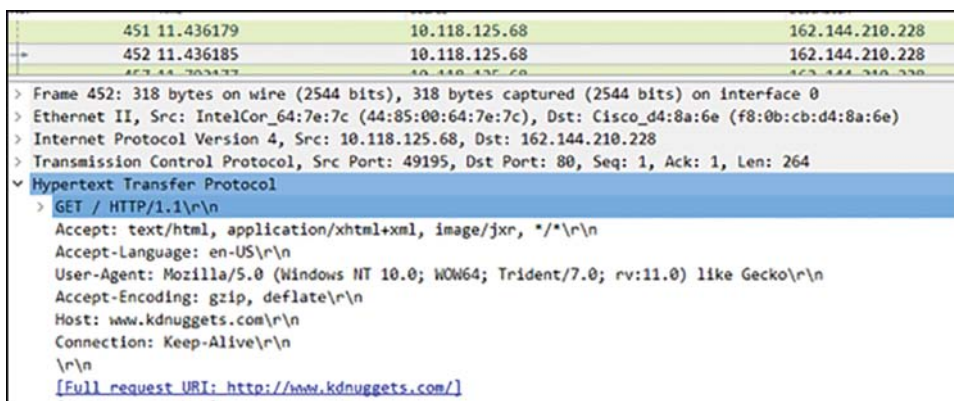


Figure 4-10 HTTP Packet from a Packet Analyzer

Figure 4-11 shows the IPv4 section from Figure 4-10 opened up. Notice the fields for the IPv4 packet header, as identified earlier, in Figure 4-8.

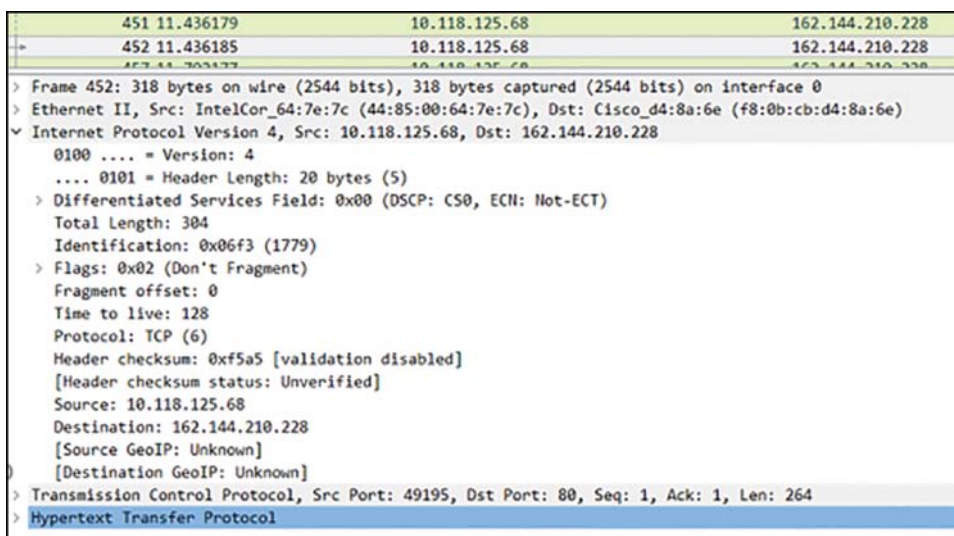


Figure 4-11 IPv4 Packet Header from a Packet Analyzer

In the final capture in Figure 4-12, notice the TCP header, which is described in Figure 4-9.

You have just seen what kind of details are provided inside the packets. NetFlow and IPFIX capture most of this data for you, either implemented in the network devices or using some offline system that receives a copy of the packets.

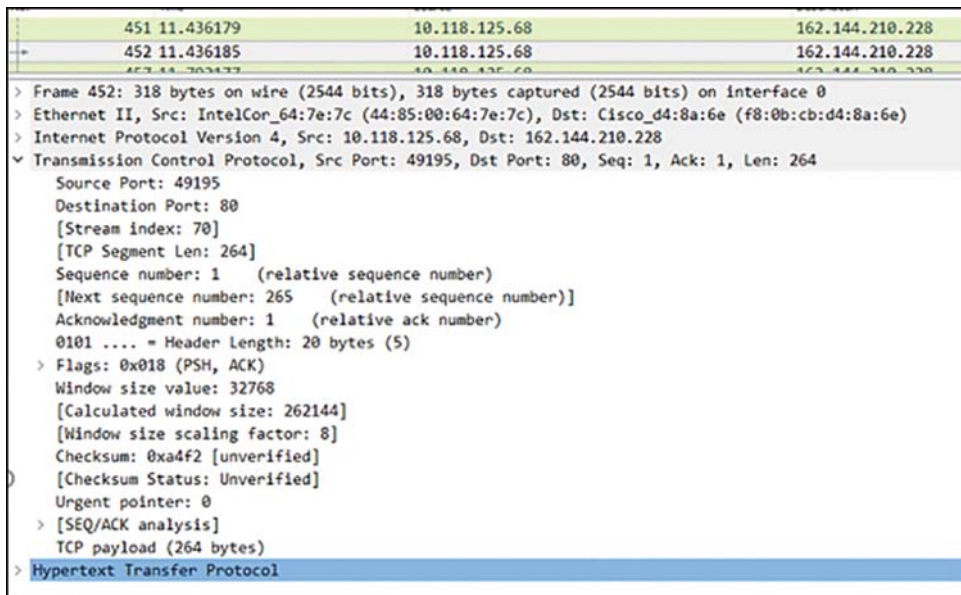


Figure 4-12 TCP Packet Header from a Packet Analyzer

Packet data can get very complex when it comes to security and encryption. Figure 4-13 shows an example of a packet that is using Internet Protocol Security (IPsec) transport mode. Note that the entire TCP header and payload section are encrypted; you cannot analyze this encrypted data.

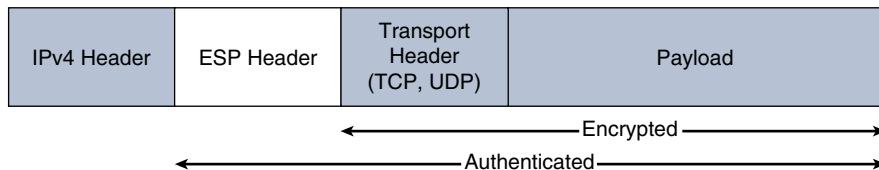


Figure 4-13 IPsec Transport Mode Packet Format

IPsec also has a tunnel mode, which even hides the original source and destination of the internal packets with encryption, as shown in Figure 4-14.

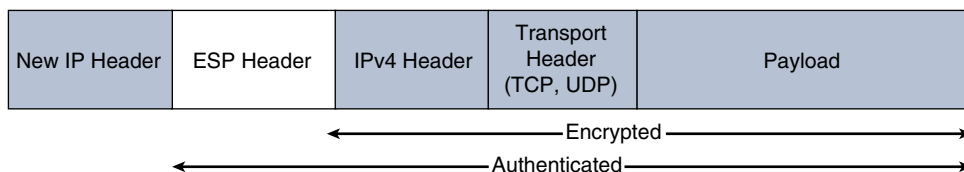


Figure 4-14 IPsec Tunnel Mode Packet Format

What does encrypted data look like to the analyzer? In the case of HTTPS, or Secure Sockets Layer (SSL)/Transport Layer Security (TLS), just the HTTP payload in a packet is encrypted, as shown in the packet sample in Figure 4-15.

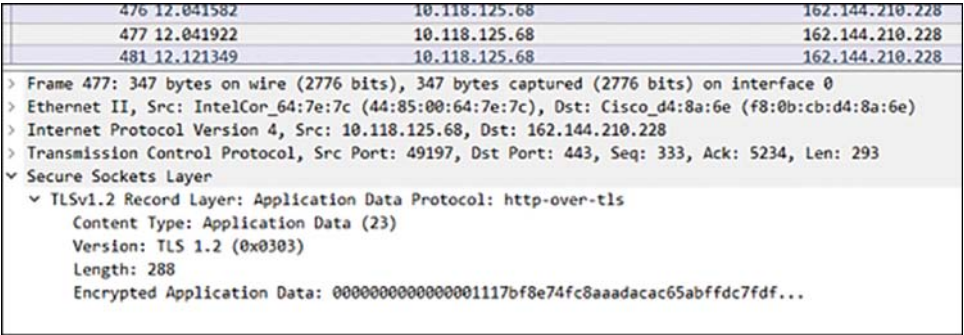


Figure 4-15 SSL Encrypted Packet, as Seen by a Packet Analyzer

In the packet encryption cases, analytics such as behavior analysis using Cisco Encrypted Threat Analytics must be used to glean any useful information from packet data. If they are your packets, gather packet data before they enter and after they leave encrypted sessions for useful data.

Finally, for the cases of network overlays (application overlays exist within network overlays), using tunnel packets such as Virtual Extensible LAN (VXLAN) is a common encapsulation method. Note in Figure 4-16 that there are multiple sets of IP headers inside and out, as well as a VXLAN portion of the packets that define the mapping of packets to the proper network overlay. Many different application instances, or “application overlays,” could exist within the networks defined inside the VXLAN headers.

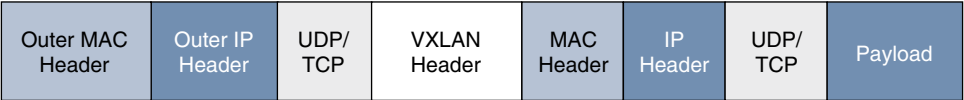


Figure 4-16 VXLAN Network Overlay Packet Format

Other Data Access Methods

You have already learned about a number of common methods for data acquisition. This section looks at some uncommon methods that are emerging that you should be aware of.

Container on Box

Many newer Cisco devices have a native Linux environment on the device, separate from the configuration. This environment was created specifically to run Linux containers such that local services available in Linux are deployed at the edge (which is useful for

fog computing). With this option, you may not have the resources you typically have in a high end server, but it is functional and useful for first-level processing of data on the device. When coupled with model application in a deployment example, the containers make local decisions for automated configuration and remediation.

Internet of Things (IoT) Model

The Global Standards Initiative on Internet of Things defines IoT as a “global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.” Interconnecting all these things means there is yet more data available—sensor data.

IoT is very hot technology right now, and there are many standards bodies defining data models, IoT platforms, security, and operational characteristics. For example, oneM2M (<http://www.onem2m.org>) develops technical specifications with a goal of a common M2M service layer to embed within hardware and software for connecting devices in the field with M2M application servers worldwide. The European Telecommunications Standards Institute (ETSI) is also working on M2M initiatives for standardizing component interfaces and IoT architectures (<http://www.etsi.org/technologies-clusters/technologies/internet-of-things>). If you are working at the edge of IoT, you can go much deeper into IoT by reading the book *Internet of Things—From Hype to Reality*, by Ammar Rayes and Samer Salam.

IoT environments are generally custom built, and therefore you may not have easy access to IoT protocols and sensor data. If you do, you should treat it very much like telemetry data, as discussed earlier in this chapter. In some cases, you can work with your IT department to bring this data directly into your data warehouse from a data pipeline to the provider connection. In other cases, you may be able to build models from the data right in the provider cloud.

Sensor data may come from a carrier that has done the aggregation for you. Large IoT deployments produce massive amounts of data. Data collection and aggregation schemes vary by industry and use case. In the analytics infrastructure model data section in Figure 4-17, notice the “meter” and “boss meter” examples. In one utility water meter use case, every house has a meter, and every neighborhood has a “boss meter” that aggregates the data from that neighborhood. There may be many levels of this aggregation before the data is aggregated and provided to you. Notice how to use the data section of the analytics infrastructure model in Figure 4-17 to identify the relevant components for your solution. You can grow your own alternatives for each section of the analytics infrastructure model as you learn more.

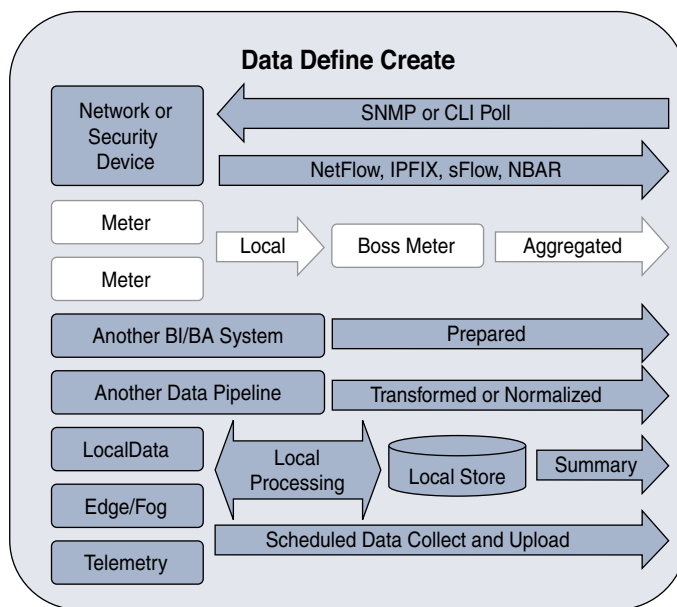


Figure 4-17 *Analytics Infrastructure Model IoT Meters Example*

This is just one example of IoT data requirements. IoT, as a growing industry, defines many other mechanisms of data acquisition, but you only need to understand what comes from your IoT data provider unless you will be interfacing directly with the devices. The IoT industry coined the term *data gravity* to refer to the idea that data attracts more data. This immense volume of IoT data attracts systems and more data to provide analysis where it resides, causing this gravity effect. This volume of available data can also increase latency when centralizing, so you need to deploy models and functions that act on this data very close to the edge to provide near-real-time actions. Cisco calls this edge processing *fog computing*.

One area of IoT that is common with networking environments is event processing. Much of the same analysis and collection techniques used for syslog or telemetry data can apply to events from IoT devices. As you learned in Chapter 2, *Approaches for Analytics and Data Science*, you can build these models locally and deploy them remotely if immediate action is necessary.

Finally, for most enterprises, the wireless network may be a source of IoT data for things that exist within the company facilities. In this case, you can treat IoT devices like any other network component with respect to gathering data.

Data Types and Measurement Considerations

Data has fundamental properties that are important for determining how to use it in analytics algorithms. As you go about identifying and collecting data for building your

solutions, it is important to understand the properties of the data and what to do with those properties. The two major categories of data are nominal and numerical. Nominal (categorical) data is either numbers or text. Numbers (numerical) have a variety of meanings and can be interpreted as continuous/discrete numerical values, ordinals, ratios, intervals, and higher-order numbers.

The following sections examine considerations about the data, data types, and data formats that you need to understand in order to properly extract, categorize, and use data from your network for analysis.

Numbers and Text

The following sections look at the types of numbers and text that you will encounter with your collections. The following sections also share a data science and programming perspective for how to classify this data when using it with algorithms. As you will learn later in this chapter, the choice of algorithm often determines the data type requirement.

Nominal (Categorical)

Nominal data, such as names and labels, are text or numbers in mutually exclusive categories. You can also call nominal values *categorical* or *qualitative* values. The following are a few examples of nominal data and possible values:

Hair color:

- Black
- Brown
- Red
- Blond

Router type:

- 1900
- 2900
- 3900
- 4400

If you have an equal number of Cisco 1900 series routers and Cisco 2900 series routers, can you say that your average router is a Cisco 2400? That does not make sense. You cannot use the 1900 and 2900 numbers that way because these are categorical numbers. Categorical values are either text or numbers, but you cannot do any valid math with the numbers. In data networking, categorical data provides a description of features of a component or system. When comparing categorical values to numerical values, it is clear

that a description such as “blue” is not numerical. You have to be careful when doing analysis when you have a list such as the following:

Choose a color:

- 1—Blue
- 2—Red
- 3—Green
- 4—Purple

Categorical values are descriptors developed using data mining to assign values, text analytics, or analytics-based classification systems that provide some final classification of a component or device. You often choose the label for this classification to be a simple list of numbers that do not have numerical meaning.

Device types:

- 1—Router
- 2—Switches
- 3—Access points
- 4—Firewalls

For many of the algorithms used for analytics, categorical values are codified in numerical form in one way or another, but they still represent a categorical value and therefore should not be thought of as numbers. Keeping the values as text and not codifying into numbers in order to eliminate confusion is valid and common as well.

The list of device types just shown represents an encoding of a category to a number, which you will see in Chapters 11, 12, and 13, “Developing Real Use Cases: Network Infrastructure Analytics,” “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry,” “Developing Real Use Cases: Data Plane Analytics.” You must be careful when using algorithms with this encoding because the numbers have no valid comparison. A firewall (4) is not four times better than a router (1). This encoding is done for convenience and ease of use.

Continuous Numbers

Continuous data is defined in mathematical context as being infinite in range. In networking, you can consider continuous data a continuous set of values that fall within some range related to the place from which it originated. For many numbers, there is a minimum, a maximum, and a full range of values in between. For example, a Gigabit Ethernet interface can have a bandwidth measurement that falls anywhere between 0 and 1,000,000,000 (1 GB). Higher and lower places on the scale have meaning here.

In the memory example in Chapter 3, if you develop a prediction line using algorithms that predict continuous variables, the prediction at some far point in the future may well exceed the amount of memory in the router. That is fine: You just need to see where it hits that 80%, 90%, and 100% consumed situation.

Discrete Numbers

Discrete numbers are a list of numbers where there are specific values of interest, and other values in the range are not useful. These could be counts, binned into ordinal categories such as survey averages on a 10-point rating scale. In other cases, the order may not have value, but the values in the list cannot take on *any* value in the group of possible numbers—just a select few values. For example, you might say that the interface speeds on a network device range from 1 Gbps to 100 Gbps, but a physical interface of 50 Gbps does not exist. Only discrete values in the range are possible. Order may have meaning in this case if you are looking at bandwidth. If you are looking at just counting interfaces, then order does not matter.

Gigabit interface bandwidth:

- 10
- 40
- 100

Sometimes you want to simplify continuous outputs into discrete values. “Discretizing,” or binning continuous numbers into discrete numbers, is common. Perhaps you want to know the number of megabits of traffic in whole numbers. In this case, you can round up the numbers to the closest megabyte and use the results as your discrete values for analysis.

Ordinal Data

Ordinal data is categorical, like nominal data, in that it is qualitative and descriptive; however, with ordinal data, the order matters. For example, in the following scale, the order of the selections matters in the analysis:

How do you feel about what you have read so far in this book?

- 1—Very unsatisfied
- 2—Slightly unsatisfied
- 3—I’m okay
- 4—Pleased
- 5—Extremely pleased

These numbers have no real value; adding, subtracting, multiplying, or dividing with them makes no sense.

The best way to represent ordinal values is with numbers such that order is useful for mathematical analysis (for example, if you have 10 of these surveys and want to get the “average” response). For network analysis, ordinal data is very useful for “bucketing” continuous values to use in your analysis as indicators to provide context.

Bandwidth utilization:

- 1—Average utilization less than or equal to 500 Mbps
- 2—Average utilization greater than 500 Mbps but less than less than 1 Gbps
- 3—Average utilization greater than 1 Gbps but less than less than 5 Gbps
- 4—Average utilization greater than 5 Gbps but less than less than 10 Gbps
- 5—Average utilization greater than 10 Gbps

In ordinal variables used as numeric values, the difference between two values does not usually make sense unless the categories are defined with equal spacing, as in the survey questions. Notice in this bandwidth utilization example that categories 3 and 4 are much larger than the other categories in terms of the range of bandwidth utilization. However, the buckets chosen with the values 1 through 5 may make sense for what you want to analyze.

Interval Scales

Interval scales are numeric scales in which order matters and you know the exact differences between the values. Differences in an interval scale have value, unlike with ordinal data. You can define bandwidth on a router as an interval scale between zero and the interface speed. The bits per second increments are known, and you can add and subtract to find differences between values. Statistical central tendency measurements such as mean, median, mode, and standard deviation are valid and useful. You clearly know the difference between 1 Gbps and 2 Gbps bandwidth utilization.

A challenge with interval data is that you cannot calculate ratios. If you want to compare two interfaces, you can subtract one from the other to see the difference, but you should not divide by an interface that has a value of zero to get a ratio of how much higher one interface bandwidth is compared to the other. Interval values are best defined as variables where taking an average makes sense.

Interval values are useful in networking when looking at average values over date and time ranges, such as a 5-minute processor utilization, a 1-minute bandwidth utilization, or a daily, weekly, or monthly packet throughput calculation. The resulting values of these calculations produce valid and useful data for examining averages.

Ratios

Ratio values have all the same properties as interval variables, but the zero value must have meaning and must not be part of the scale. A zero means “this variable does not exist” rather than having a real value that is used for differencing, such as a zero bandwidth count. You can multiply and divide ratio values, which is why the zero cannot be part of the scale, as multiplying by any zero is zero, and you cannot divide by zero.

There are plenty of debates in the statistical community about what is interval only and what can be ratio, but do not worry about any of that. If you have analysis with zero values and the interval between any two of those values is constant and equal, you can sometimes just add one to everything to eliminate any zeros and run it through some algorithms for validation to see if it provides suitable results. A common phrase used in analytics comes from George Box: “All models are wrong, but some are useful.” “Off by one” is a nightmare in programming circles but is useful when you are dealing with calculations and need to eliminate a zero value.

Higher-Order Numbers

The “higher orders” of numbers and data is a very important concept for advanced levels of analysis. If you are an engineer, then you had calculus at some point in your career, so you may already understand that you can take given numbers and “derive” new values (derivatives) from the given numbers. Don’t worry: This book does not get into calculus. However, the concept still remains valid. Given any of the individual data points that you collect from the various planes of operation, higher-order operations may provide you with additional data from those points. Let’s use the router memory example again and the “driving to work” example to illustrate:

1. You can know the memory utilization of the router at any given time. This is simply the values that you pull from the data. You also know your vehicle position on the road at any point in time, based on your GPS data. This is the first level of data. Use first-level numbers to capture the memory available in a router or the maximum speed you can attain in the car from the manufacturer.
2. How do you know your current speed, or velocity, in the car? How do you know how much memory is currently being consumed (leaked in this case) between any two time periods? You derive this from the data that you have by determining your memory value (or vehicle location) at point A and at point B, determining distance with a $B - A$ calculation, and divide by the time it took you to get there. Now you have a *new* value for your analysis: the “rate of change” of your initial measured value. Add this to your existing data or create a new data set. If the speed is not changing, use this first derivative of your values to predict the time it will take you to reach a given distance or the time to reach maximum memory with simple extrapolation.
3. Maybe the rate of change for these values is not the same for each of these measured periods; it is not constant. Maybe your velocity from measurement is changing because you are stepping on the gas pedal. Maybe conditions in your network are changing the rates of memory loss in your router from period to period. This is acceleration, which is the third level (the rate of change again) derived from the second-level speed that you already calculated. In this case, use these third-level values to develop a functional analysis that predicts where you will reach critical thresholds, such as the speed limit or the available memory in your router.

- 4. There are even higher levels related to the amount of pressure you apply to the gas pedal or steering wheel (it’s called jerk) or the amount of instant memory draw from the input processes that consume memory, but those levels are deeper than you need to go when collecting and deriving data for learning initial data science use cases.

Data Structure

The following sections look at how to gather and share collections of the atomic data points that you created in the previous section.

Structured Data

Structured data is data that has a “key = value” structure. Assume that you have a spreadsheet containing the data shown in Table 4-1. There is a column heading (often called a *key*), and there is a *value* for that heading. Each row is a *record*, with the value of that *instance* for that column header key. This is an example of structured data. *Structured data* means it is formed in a way that is already known. Each value is provided, and there is a label (key) to tell what that value represents.

Table 4-1 Structured Data Example

Device	Device Type	IP Address	Number of Interfaces
Device1	Router	10.1.1.1	2
Device2	Router	10.1.1.2	2
Device3	Switch	10.1.1.3	24

If you have structured spreadsheet data, then you can usually just save it as a comma-separated values (CSV) file and load it right into an analytics package for analysis. Your data could also be in a database, which has the same headers, and you could use database calls such as Structured Query Language (SQL) queries to pull this from the data engine part of the design model right into your analysis. You may pull this from a relational database management system (RDBMS). Databases are very common sources for structured data.

JSON

You will often hear the term *key/value pairs* when referencing structured data. When working with APIs, using JavaScript Object Notation (JSON) is a standardized way to move data between systems, either for analysis or for actual operation of the environment. You can have an API layer that pulls from your database and, instead of giving you a CSV, delivers data to you record by record. What is the difference? JSON provides the data row by row, in pairs of keys and values.

Here is a simple an example of some data in a JSON format, which translates well from a row in your spreadsheet to the Python dictionary format **Key: Value**:

```
{ "productFamily": "Cisco_ASR_9000_Series_Aggregation_Services_Routers",
  "productType": "Routers",
  "productId": "ASR-9912" }
```

As with the example of planes within planes earlier in the chapter, it is possible that the value in a **Key: Value** pair is another key, and that key value is yet another key. The value can also be lists of items. Find out more about JSON at one of my favorite sites for learning web technologies: https://www.w3schools.com/js/js_json_intro.asp.

Why use JSON? By standardizing on something common, you can use the data for many purposes. This follows the paradigm of building your data pipelines such that some new and yet-to-be-invented system can come along and plug into the data platform and provide you with new insights that you never knew existed.

Although it is not covered in this book, Extensible Markup Language (XML) is another commonly used data source that delivers key/value pairs. YANG/NETCONF is based on XML principles. Find more information about XML at <https://www.w3schools.com/xml/default.asp>.

Unstructured Data

This paragraph is an example of unstructured data. You do not have labels for anything in this paragraph. If you are doing CLI scraping, the results from running the commands come back to you as unstructured data, and you must write a parser to select values to put into your database. Then these values with associated fields (keys or labels) can be used to query known information. You create the keys and assign values that you parsed. Then you have structured data to work with.

In the real world, you see this kind of data associated with tickets, cases, emails, event logs, and other areas where humans generate information. This kind of data requires some kind of specialized parsing to get any real value from it.

You do not have to parse unstructured data into databases. Packages such as Splunk practice “schema on demand,” which simply means that you have all the unstructured text available, and you parse it with a query language to extract what you need, when you need it. Video is a form of unstructured data. Imagine trying to collect and parse video pixels from every frame. The processing and storage requirements would be massive. Instead, you save it as unstructured data and parse it when you need it.

For IT networking data, often you do not know which parts have value, so you store full “messages” for schema parsing on demand. A simple example is syslog messages. It is impossible to predict all combinations of values that may appear in syslog messages such that you can parse them into databases on receipt. However, when you *do* find a new value of interest, is it extremely powerful to be able to go back through the old messages

and “build a model”—or a search query in this case—to identify that value in future messages. With products such as Splunk, you can even deploy your model to production by building a dashboard that presents the findings in your search and analysis related to this new value found in the syslog messages. Perhaps it is a log related to low memory on a routing device.

Semi-Structured Data

In some cases, such as with the syslog example just discussed, data may come in from a specific host in the network. While the message is stored in a field with a name like “the whole unstructured message,” the sending host is stored in a field with the sending host name. So your host name and the blob of message text together are structured data, but the blob of message text is unstructured within. The host that you got it from has a label. You can ask the system for all messages from a particular host, or perhaps your structured fields also have the type of device, such as a router. In that case, you can do analysis on the unstructured blob of message text in the context of all routers.

Data Manipulation

Many times you will use the data you collect as is, but other times you will want to manipulate the data or add to it.

Making Your Own Data

So far, atomic data points and data that you extract, learn, or otherwise infer from instances of interest have been discussed. When doing feature engineering for analytics, sometimes you have a requirement to “assign your own” data or take some of the atomic values through an algorithm or evaluation method and use the output of that method as a value in your calculation. For example, you may assign network or geographic location, criticality, business unit, or division to a component.

Here is an example of made-up data for device location (all of which could be the same model of device):

- Core network
- Subscriber network
- Corporate internal WAN
- Internet edge environment

Your “algorithm” for producing this data in this location example may simply be parsing regular expressions on host names if you used location in your naming scheme. For building models, you can use the regex to identify all locations that have the device names that represent characteristics of interest.

If you decide to use an algorithm to define your new data, it may be the following:

- Aggregate bandwidth utilization
- Calculated device health score
- Probability to hit a memory leak
- Composite MTBF (mean time between failures)

This enrichment data is valuable for analysis as you recognize areas of your environment that are in different “populations” for analysis. Because an analytics model is a generalization, it is important to have qualifiers that allow you to identify the characteristics of the environments that you want to generalize. Context is very useful with analytics.

Standardizing Data

Standardizing data involves taking data that may have different ranges, scales, and types and putting it into a common format such that comparison is valid and useful. When looking at the memory utilization example earlier in this chapter, note that you were using percentage as a method of standardization. Different components have differing amounts of available memory, so comparing the raw memory values does not provide a valid comparison across devices, and you may therefore standardize to percentage.

In statistics and analytics, you use many methods of data standardization, such as relationship to the mean or mode, zero-to-one scaling, z-scores, standard deviations, or rank in the overall range. You often need to rescale the numbers to put them on a finite scale that is useful for your analysis.

For categorical standardization, you may want to compare routers of a certain type or all routers. You can standardize the text choices as “router,” “switch,” “wireless,” or “server” for the multitude of components that you have. Then you can standardize to other subgroups within each of those. There are common mechanisms for standardization, or you can make up a method to suit your needs. You just need to ensure that they provide a valid comparison metric that adds value to your analysis.

Cisco Services standardizes categorical features by transforming data observations to a matrix or an array and using encodings such as simple feature counts, one-hot encoding, or term frequency divided by inverse document frequency (TF/IDF). Then it is valid to represent the categorical observations relative to each other. These encoding methods are explained in detail in Chapter 8, “Analytics Algorithms and the Intuition Behind Them.”

You may also see the terms *data normalization*, *data munging*, and *data regularization* associated with standardization. Each of these has its own particular nuances, but the theme is the same: They all involve getting data into a form that is usable and desired for storage or use with algorithms.

Missing Data

Missing and unavailable data is a very common problem when working with analytics. We have all had spreadsheets that are half full of data and hard to understand. It is even harder for machines to understand these spreadsheets. For data analytics, missing data often means a device needs to be dropped from the analysis. You can sometimes generate the missing data yourself. This may involve adding inline scripting or programming to make sure it goes into the data stores with your data, or you can add it after the fact. You can use the analytics infrastructure model to get a better understanding of your data pipeline flow and then choose a spot to insert a new function to change the data. Following are some ideas for completing incomplete data sets:

- Try to infer the data from other data that you have about the device. For example, the software name may contain data about the device type.
- Sometimes an educated guess works. If you know specifics about what you are collecting, sometimes you may already know missing values.
- Find a suitable proxy that delivers the same general meaning. For example, you can replace counting active interfaces on an optical device with looking at the active interface transceivers.
- Take the average of other devices that you cluster together as similar to that device. If most other values match a group of other devices, take the mean, mode, or median of those other device values for your variable.
- Instead of using the average, use the mode, which is the most common value.
- Estimate the value by using an analytics algorithm, such as regression.
- Find the value based on math, using other values from the same entity.

This list is not comprehensive. When you are the SME for your analysis, you may have other creative ways to fill in the missing data. The more data you have, the better you can be at generalizing it with analytics. Filling missing data is usually worth the effort.

You will commonly encounter the phrase *data cleansing*. Data cleansing includes addressing missing data, as just discussed, as well as removing outliers and values that would decrease the effectiveness of the algorithms you will use on the data. How you handle data cleansing is algorithm specific and something that you should revisit when you have your full analytics solution identified.

Key Performance Indicators

Throughout all of the data sources mentioned in this chapter, you will find or create many data values. You and your stakeholders will identify some of these as key performance indicators (KPIs). These KPIs could be atomic collected data or data created by you. If you do not have KPIs, try to identify some that resonate with you, your management, and the key users of the solutions that you will provide. Technical KPIs (not business KPIs, such as revenue and expense) are used to gauge health, growth, capacity, and

other factors related to your infrastructure. KPIs provide your technical and nontechnical audiences with something that they can both understand and use to improve and grow the business. Do you recall mobile carriers advertising about “most coverage” or “highest speeds” or “best reliability”? Each of these—coverage, speed, and reliability—is a technical KPI that marketers use to promote companies and consumers use to make buying choices.

You can also compare this to the well-known business KPIs of sales, revenue, expense, margins, or stock price to get a better idea of what they provide and how they are used. One on hand, a KPI is a simple metric that people use to make a quick comparison and assessment, but on the other, it is a guidepost for you for building analytics solutions. Which solutions can you build to improve the KPIs for your company?

Other Data Considerations

The following sections provide a few additional areas for you to consider as you set up your data pipelines.

Time and NTP

Time is a critical component of any analysis that will have a temporal component. Many of the push components push their data to some dedicated receiving system. Timestamps on the data should be subject to the following considerations during your data engineering phase:

- For the event that happened, what time is associated with the exact time of occurrence?
- Is the data for a window of time? Do I have the start and stop times for that window?
- What time did the sending system generate and send the data?
- What time did the collection system receive the data?
- If I moved the data to a data warehouse, is there a timestamp associated with that? I do not want to confuse this with any of the previous timestamps.
- What is the timestamp when I accessed the data? Again, I do not want to use this if I am doing event analysis and the data has timestamps within.

Some of these considerations are easy, and data on them is provided, but sometimes you will need to calculate values (for example, if you want to determine the time delta between two events).

Going back to the discussion of planes of operation, also keep in mind awareness of the time associated with each plane and which level of infrastructure it originated within. As shown in the diagram in Figure 4-18, each plane commonly has its own associated configuration for time, DNS, logging, and many other data sources. Ensure that a common time source is available and used by all of the systems that provide data.

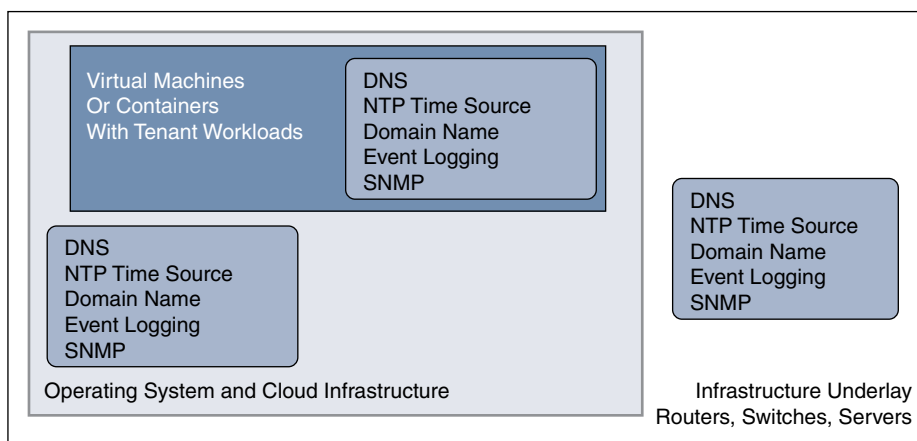


Figure 4-18 *NTP and Network Services in Virtualized Architectures*

The Observation Effect

As more and more devices produce data today, the observation effect comes into play. In simple terms, the observation effect refers to changes that happen when you observe something—because you observed it. Do you behave differently when someone is watching you?

For data and network devices, data generation could cause this effect. As you get into the details of designing your data pipelines, be sure to consider the impact that your collection will have on the device and the surrounding networks. Excessive polling of devices, high rates of device data export, and some protocols can consume resources on the device. This means that you affect the device from which you are extracting data. If the collection is a permanent addition, then this is okay because it is the “new normal” for that component. In the case of adding a deep collection method for a specific analysis, you could cause a larger problem than you intend to solve by stressing the device too much with data generation.

Panel Data

Also called longitudinal data, panel data is a data set that is captured over time about multiple components and multiple variables for those components of interest. Sensor data from widespread environments such as IoT provides panel data. You often see panel data associated with collections of observations of people over time for studies of differences between people in health, income, and aging. Think of panel data in terms of collection from your network as the set of all network devices with the same collection over and over again and adding a time variable to use for later trending. When you want to look at a part of the population, you slice it out. If you want to compare memory utilization behavior in different types of routers, slice the routers out of the panel data and perform analysis that compares one group to others, such as switches, or to members of the same group, such as other routers. Telemetry data is a good source of panel data.

External Data for Context

As you have noticed in this chapter, there is specific lingo in networking and IT when it comes to data. Other industries have their own lingo and acronyms. Use data from your customer environment, your business environment, or other parts of your business to provide valuable context to your analysis. Be sure that you understand the lingo and be sure to standardize where you have common values with different names.

You might assume that external data for context is sitting in the data store for you, and you just need to work with your various departments to gain access. If you are not a domain expert in the space, you may not know what data to request, and you may need to enlist the help of some SME peers from that space.

Data Transport Methods

Are you tired of data yet? This section finally moves away from data and takes a quick run through transports and getting data to your data stores as part of the analytics infrastructure model shown in Figure 4-19.

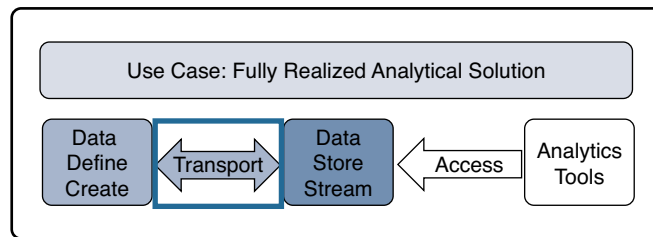


Figure 4-19 *Analytics Infrastructure Model Data Transports*

For each of the data acquisition technologies discussed so far, various methods are used for moving the data into the right place for analysis. Some data provides a choice between multiple methods, and for some data there is only a single method and place to get it. Some derivation of data from other data may be required. For the major categories already covered, let's now examine how to set up transport of that data back to a storage location.

Once you find data that is useful and relevant, and you need to examine this data on a regular basis, you can set up automated data pulling and storage on a central location that is a big data cluster or data warehouse environment. You may only need this data for one purpose now, but as you grow in your capabilities, you can use the data for more purposes in the future. For systems such as NMSs or NetFlow collectors that collect data into local stores, you may need to work with your IT developers to set up the ability to move or copy the data to the centralized data environment on an automated, regular basis. Or you might choose to leave the data resident in these systems and access it only when you need it. In some cases, you may take the analysis to the data, and the data may never need to be moved. This section is for data that will be moved.

Transport Considerations for Network Data Sources

Cisco Services distinguishes between the concepts high-level design (HLD) and low-level design (LLD). HLD is about defining the big picture, architecture, and major details about what is needed to build a solution. The analytics infrastructure model is very much about designing the big picture—the architecture—of a full analytics overlay solution. The LLD concept is about uncovering all the details needed to support a successful implementation of the planned HLD. This building of the details needed to fully set up the working solution includes data pipeline engineering, as shown in Figure 4-20.

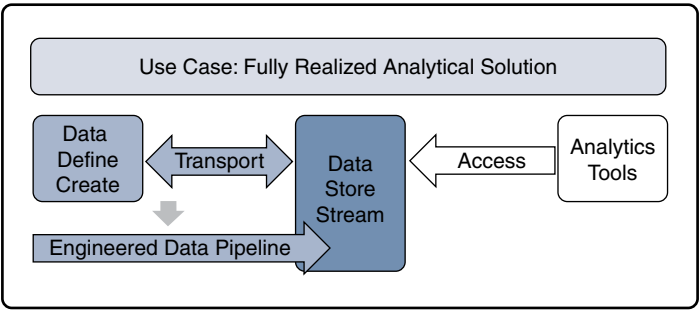


Figure 4-20 *Data Pipeline Engineering*

Once you use the generalized analytics infrastructure model to uncover your major requirements, engineering the data pipeline is the LLD work that you need to do. It is important to document in detail during this pipeline engineering as you commonly reuse components of this work for other solutions.

The following sections explore the commonly used transports for many of the protocols mentioned earlier. Because it is generally easy to use alternative ports in networks, this is just a starting point for you, and you may need to do some design and engineering for your own solutions. Some protocols do not have defined ports, while others do. Determine your options during the LLD phase of your pipeline engineering.

SNMP

The first transport to examine is SNMP, because it is generally well known and a good example to show why the data side of the analytics infrastructure model exists. (Using something familiar to aid in developing something new is a key innovation technique that you will want to use in the upcoming chapters.) Starting with SNMP and the components shown in Figure 4-21, let's go through a data engineering exercise.

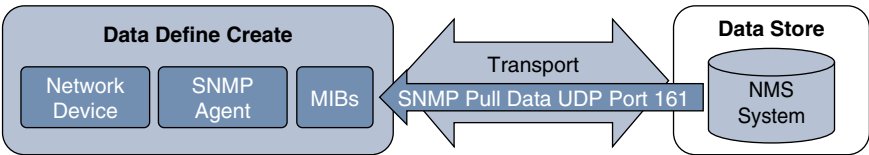


Figure 4-21 *SNMP Data Transport*

You have learned (or already knew) that network devices have SNMP agents, and the agents have specific information available about the environment, depending on the MIBs that are available to each SNMP agent. By standard, you know that NMSs use User Datagram Protocol (UDP) as a transport, and SNMP agents are listening on port 161 for your NMS to initiate contact to poll the device MIBs. This is the HLD of how you are going to get polled SNMP data. This is where simplified “thinking models” such as the analytics infrastructure model are designed to help—and also where they stop. Now you need to uncover the details.

So how does the Cisco Services HLD/LLD concept apply to the SNMP example? Perhaps from an HLD/analytics infrastructure perspective, you have determined that SNMP provides the data you want, so you want to get that data and use the SNMP mechanisms to do so. Now consider that you need to work on the details, the following LLD items, for every instance where you need it, in order to have a fully engineered data pipeline set up for analysis and reuse:

1. Is the remote device already configured for SNMP as I need it?
2. What SNMP version is running? What versions are possible?
3. Can I access the device, given my current security environment?
4. Do I need the capabilities of some other version?
5. How would I change the environment to match what I need?
6. Are my MIBs there, or do I need to put them there?
7. Can I authenticate to the device?
8. What mechanism do I need to use to authenticate?
9. Does my authentication have the level of access that I need?
10. What community strings are there?
11. Do I need to protect any sessions with encryption?
12. Do I need to set up the NMS, or is there one readily available to me?
13. What are the details for accessing and using that system?
14. Where is the system storing the data I need?
15. Can I use the data in place? Do I need to copy it?
16. Can I set up an environment where I will always have access to the latest information from this NMS?
17. Can I access the required information all the time, or do I need to set up sharing/moving with my data warehouse/big data environment?
18. If I need to move the data from the NMS, do I need push or pull mechanisms to get the data into my data stores?

19. How will I store the data if I need to move it over? Will it be raw? In a database?
20. Do I need any data cleansing on the input data before I put it into the various types of stores (unstructured raw, parsed from an RDBMS, pulled from object storage)?
21. Do I need to standardize the data to any set of known values?
22. Do I need to normalize the data?
23. Do I need to transform/translate the data into other formats?
24. Will I publish to a bus for others to consume as the data comes into my environment? Would I publish what I clean?
25. How will I offer access to the data to the analytics packages for production deployment of the analysis that I build?

Your data engineering, like Cisco LLD, should answer tens, hundreds, or thousands of these types of questions. We stop at 25 questions here, but you need to capture and answer all questions related to each of your data sources and transports in order to build a resilient, reusable data feed for your analytics efforts today and into the future.

The remainder of this section identifies the analytics infrastructure model components that are important for the HLD of each of these data sources. Since this is only a single chapter in a book focused on the analytics innovation process, doing LLD for every one of these sources would add unnecessary detail and length. Defining the lowest-level parts of each data pipeline design is up to you as you determine the data sources that you need. In some cases, as with this SNMP example, you will find that the design of your current, existing NMS has already done most of the work for you, and you can just identify what needs to happen at the central data engine or NMS part of the analytics infrastructure model.

CLI Scraping

For CLI scraping, the device is accessed using some transport mechanism such as SSH, Telnet, or an API. The standard SSH port is TCP port 22, as shown in the example in Figure 4-22. Telnet uses TCP 25, and API calls are according to the API design but are typically at something at or near port 80 or 443 if secured and at ports 8000, 8080, or 8443 if obscured.

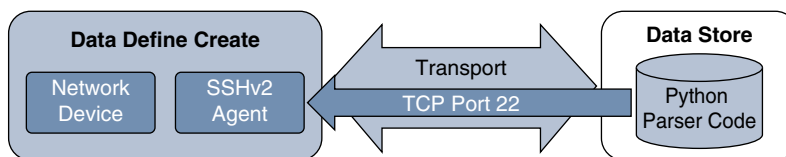


Figure 4-22 *SSHv2 Transport*

Other Data (CDP, LLDP, Custom Labels, and Tags)

Other data defined here is really context data about your device that comes from sources that are *not* your device. This data may come from neighboring devices where you use the previously discussed SNMP, CLI, or API mechanisms to retrieve the data, or it may come from data sets gathered from outside sources and stored in other data stores, such as a monetary value database, as in the example shown in Figure 4-23.

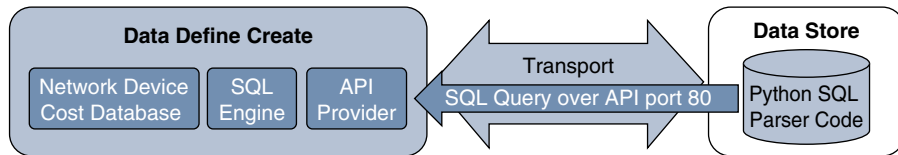


Figure 4-23 *SQL Query over API*

SNMP Traps

SNMP traps involve data pushed by devices. Traps are selected events, as defined in the MIBs, sent from the device using UDP on port 162 and usually stored in the same NMS that has the SNMP polling information, as shown in Figure 4-24.

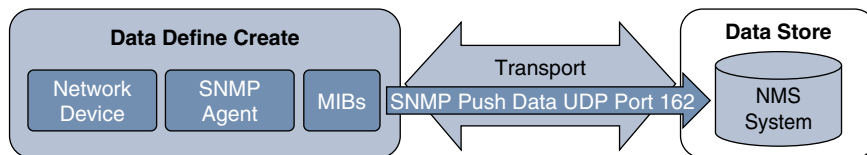


Figure 4-24 *SNMP Traps Transport*

Syslog and System Event Logs

Syslog is usually stored on the device in files, and syslog export to standard syslog servers is possible and common. Network devices (routers, switches, or servers providing network infrastructure) copy this traffic to a remote location using standard UDP port 514. For server devices and software instances, a software package such as rsyslog (www.rsyslog.com) or syslog-ng (<https://syslog-ng.org>) and special configuration for the package for each log file may need to be set up.

Much as with NMS, there are also dedicated systems designed to receive large volumes of syslog from many devices at one time. An example of a syslog pipeline for servers is shown in Figure 4-25.

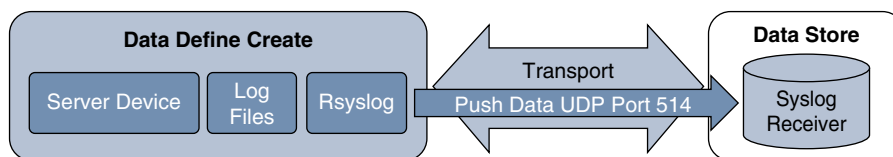


Figure 4-25 *Syslog Transport*

Telemetry

Telemetry capability is available in all newer Cisco software and products, such as IOS XR, IOS XE, and NX-OS. Most work in telemetry at the time of this writing is focused on YANG model development and setting up the push from the device for specific data streams. Whether configured manually by you or using an automation system, this is push capability, as shown in Figure 4-26. Configuring this way is called a “dial-out” configuration.

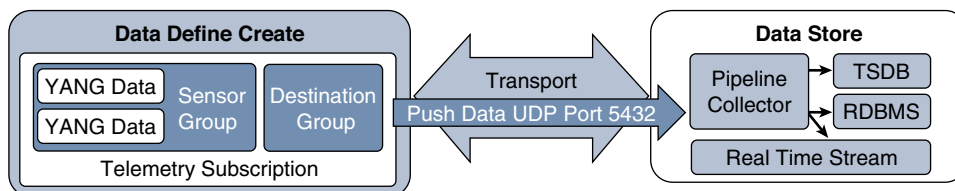


Figure 4-26 *Telemetry Transport*

You can extract telemetry data from devices by configuring the available YANG models for data points of interest into a sensor group, configuring collector destinations into a destination group, and associating it all together with a telemetry subscription, with the frequency of export defined.

NetFlow

NetFlow data availability is enabled by first identifying the interfaces on the network device that participate in NetFlow to capture these statistics and then packaging up and exporting these statistics to centralized NetFlow collectors for analysis. An alternative to doing this on the device is to use your packet capture devices offline from the device. NetFlow has a wide range of commonly used ports available, as shown in Figure 4-27.

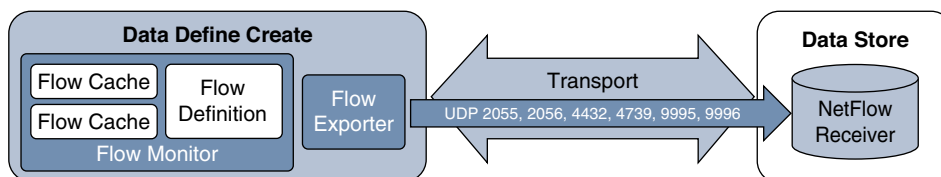


Figure 4-27 *NetFlow Transport*

IPFIX

As discussed earlier in this chapter, IPFIX is a superset of the NetFlow capabilities and is commonly called NetFlow v10. NetFlow is bound by the data capture capabilities for each version, but IPFIX adds unique customization capabilities such as variable-length fields, where data such as long URLs are captured and exported using templates. This makes IPFIX more extensible over other options but also more complex. IPFIX, shown in Figure 4-28, is an IETF standard that uses UDP port 4739 for transport by default.

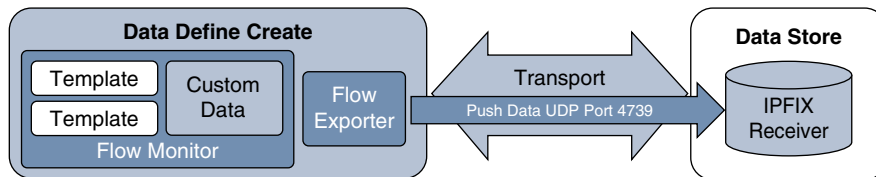


Figure 4-28 *IPFIX Transport*

You can use custom templates on the sender and receiver sides to define many additional fields for IPFIX capture.

sFlow

sFlow, defined in RFC 3176 (<https://www.ietf.org/rfc/rfc3176.txt>), is sampling technology that works at a much lower level than IPFIX or NetFlow. sFlow captures more than just IP packets; for example, it also captures Novell IPX packets. sFlow capture is typically built into hardware, and a sampling capture itself takes minimal effort for the device. As with NetFlow and IPFIX, the export process with sFlow consumes system resources.

Recall that sFlow, shown in Figure 4-29, is sampling technology, and it is useful for understanding what is on the network for network monitoring purposes. NetFlow and IPFIX are for true accounting. Use them to get full packet counts and detailed data about those packets.

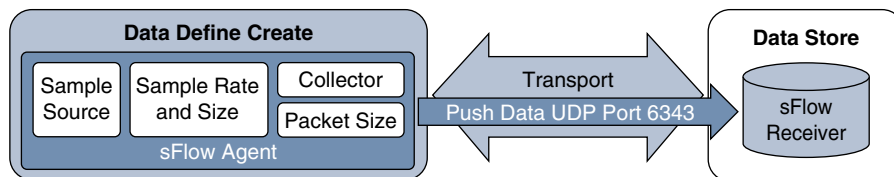


Figure 4-29 *sFlow Transport*

Summary

In this chapter, you have learned that there are a variety of methods for accessing data from devices. You have also learned that all data is not created the same way or used the same way. The context of the data is required for good analysis. “One” and “two” could be the gigabytes of memory in your PC, or they could be descriptions of doors on a game show. Doing math to analyze memory makes sense, but you cannot do math on door numbers. In this chapter you have learned about many different ways to extract data from networking environments, as well as common ways to manipulate data.

You have also learned that as you uncover new data sources, you should build data catalogs and documentation for the data pipelines that you have set up. You should document where data is available, what it signifies, how you used it. You have seen that multiple innovative solutions come from unexpected places when you combine data from disparate sources. You need to provide other analytics teams access to data that they have not had before, and you can watch and learn what they can do. Self-service is here, and citizen data science is here, too. Enabling your teams to participate by providing them new data sources is an excellent way to multiply your effectiveness at work.

In this chapter you have learned a lot about raw data, which is either structured or unstructured. You know now that you may need to add, manipulate, derive, or transform data to meet your requirements. You have learned all about data types and scales used by analytics algorithms. You have also received some inside knowledge about how Cisco uses HLD and LLD processes to work through the data pipeline engineering details. And you have learned about the details that you will gather in order to create reusable data pipelines for yourself and your peers.

The next chapter steps away from the details of methodologies, models, and data and starts the journey through cognitive methods and analytics use cases that will help you determine which innovative analytics solutions you want to develop.

Mental Models and Cognitive Bias

This chapter and Chapter 6, “Innovative Thinking Techniques,” zoom way out from the data details and start looking into techniques for fostering innovation. In an effort to find that “next big thing” for Cisco Services, I have done extensive research about interesting mechanisms to enhance innovative thinking. Many of these methods involve the use of cognitive mechanisms to “trick” your brain into another place, another perspective, another mode of thinking. When you combine these cognitive techniques with data and algorithms from the data science realm, new and interesting ways of discovering analytics use cases happen. As a disclaimer, I do not have any formal training in psychology, nor do I make any claims of expertise in these areas, but certain things have worked for me, and I would like to share them with you.

So what is the starting point? What is your current mindset? If you have just read Chapter 4, “Accessing Data from Network Components,” then you are probably deep in the mental weeds right now. Depending on your current mindset, you may or may not be very rigid about how you are viewing things as you start this chapter. From a purely technical perspective, when building technologies and architectures to certain standards, rigidity in thinking is an excellent trait for engineers. This rigidity can be applied to building mental models drawn upon for doing architecture, design, and implementation.

Sometimes mental models are not correct representations of the world. The models and lenses through which we view the business requirements from our roles and careers are sometimes biased. Cognitive biases are always lurking, always happening, and biases affect innovative thinking. Everyone has them to some degree. The good news is that they need not be permanent; you can change them. This chapter explores how to recognize biases, how to use bias to your advantage, and how to undo bias to see a new angle and gain a new perspective on things.

A clarification about the bias covered in this book: Today, many talks at analytics forums and conferences are about removing *human* bias from mathematical models—specifically race or gender bias. This type of bias is not discussed in this book, nor is much time spent discussing the purely mathematical bias related to error terms in mathematics models or

neural networks. This book instead focuses on well-known cognitive biases. It discusses cognitive biases to help you recognize them at play, and it discusses ways to *use* the biases in unconventional ways, to stretch your brain into an open net. You can then use this open net in the upcoming chapters to catch analytics insights, predictions, use cases, algorithms, and ideas that you can use to innovate in your organization.

Changing How You Think

This chapter is about you and your stakeholders, about how you think as a subject matter expert (SME) in your own areas of experience and expertise. Obviously, this strongly correlates to what you do every day. It closely correlates to the areas where you have been actively working and spending countless hours practicing skills (otherwise known as doing your job). You have very likely developed a strong competitive advantage as an expert in your space, along with an ability to see some use cases intuitively. Perhaps you have noticed that others do not see these things as you do. This area is your value-add, your competitive advantage, your untouchable value chain that makes you uniquely qualified to do your job, as well as any adjacent jobs that rely on your skills, such as developing analytics for your area of expertise. You are uniquely qualified to bring the SME perspective for these areas right out of the gate. Let's dive into what comes with this mode of thinking and how you can capitalize on it while avoiding the cognitive pitfalls that sometimes come with the SME role. This chapter examines the question "How are you so quick to know things in your area of expertise?"

This chapter also looks at the idea that being quick to know things is not always a blessing. Sometimes it gives impressions that are wrong, and you may even blurt them out. Try this example: As fast as you can, answer the following questions and jot down your answers. If you have already encountered any of them, quickly move on to the next one.

1. If a bat and ball cost \$1.10, and the bat costs \$1 more than the ball, how much does the ball cost?
2. In a lake, there is a patch of lily pads. Every day, the patch doubles in size. If it takes 48 days for the patch to cover the entire lake, how long would it take for the patch to cover half of the lake?
3. If it takes 5 machines 5 minutes to make 5 widgets, how long would it take 100 machines to make 100 widgets?

These are well-known questions from the Cognitive Reflection Test (CRT), created by Shane Frederick of MIT as part of his cognitive psychology research. The following are the correct answers as well as the common answers. Did your quick thinking fail you?

1. Did you say the ball costs 10 cents? The correct answer is that the ball cost 5 cents.
2. Did you say 24 days? The correct answer is 47 days.
3. Did you say 1 minute? The correct answer is 5 minutes.

If you see any of these questions after reading this chapter, your brain will recognize the trickery and take the time to think through the correct answers. Forcing you to stop and think is the whole point of this chapter and Chapter 6. The second part of this chapter reviews common biases. It looks into how these cognitive biases affect your ability to think about new and creative analytics use cases. As I researched ways to find out why knowledge of bias worked for me, I discovered that many of my successes related to being able to use them for deeper understanding of myself. Further, understanding these biases provided insights about my stakeholders when it came time to present my solutions to them or find new problems to solve.

Domain Expertise, Mental Models, and Intuition

What makes you a domain expert or SME in your area of expertise? In his book *Outliers: The Story of Success*, Malcolm Gladwell identifies many examples showing that engaging in 10,000 hours of deliberate practice can make you an expert in just about anything. If you relax a bit on Gladwell's *deliberate* part, you can make a small leap that you are somewhat of an expert in anything that you have been actively working on for 4 or 5 years at 2000 to 2500 hours per year. For me, that is general networking, data center, virtualization, and analytics. What is it for you? Whatever your answer, this is the area where you will be most effective in terms of analytics expertise and use-case development in your early efforts.

Mental Models

What makes you an “expert” in a space? In his book *Smarter, Faster, Better: The Secrets of Being Productive in Life and Business*, Charles Duhigg describes the concept of “mental models” using stories about nurses and airplane pilots.

Duhigg shares a story of two nurses examining the same baby. One nurse does not notice anything wrong with the baby, based on the standard checks for babies, but the second nurse cannot shake the feeling that the baby is unhealthy. This second nurse goes on to determine that the baby is at risk of death from sepsis. Both nurses have the same job role; both have been in the role for about the same amount of time. So how can they see the same baby so differently?

Duhigg also shares two pilot stories: the terrible loss of Air France flight 447 and the safe landing of Qantas Airways flight 32. He details how some pilots inexplicably find a way to safely land, even if their instruments are telling them information that conflicts with what they are feeling.

So how did the nurse and pilot do what they did? Duhigg describes using a mental model as holding a mental picture, a mental “snapshot of a good scenario,” in your brain and then being able to recognize factors in the current conditions that do and do not match that known good scenario. Often people cannot identify *why* they see what they see but just know that something is not right. Captain Chesley Sullenberger, featured in the movie *Sully*, mentioned in this book's introduction, is an airplane pilot with finely tuned

mental models. His commercial plane with 155 people on board struck a flock of geese just after leaving New York City's LaGuardia Airport in January 2009, causing loss of all engine power. He had to land the plane, and he was over New York City. Although the conditions may have warranted that he return to an airport, Sully just *knew* his plane would not make it to the New York or New Jersey airports. He safely landed flight 1549 on the Hudson River. The Qantas Airways flight 32 pilot and the nurse who found the baby's sepsis were in similar positions: Given the available information and the situation, they intuitively knew the right things to do.

So do you have any mental models? When there is an emergency, a situation, or a critical networking condition, when do you engage? When do you get called in to quickly find a root cause that nobody else sees? You may be able to find the issues and then use your skills to address the deficiencies or highlight the places where things are not matching your mental models well. Is this starting to sound familiar? You probably do this every day in your area of expertise. You just *know* when things are not right.

Whether your area of expertise is routing and switching, data center, wireless, server virtualization, or some other area of IT networking, your experiences to this point in your life have rewarded you with some level of expertise that you can combine with analytics techniques to differentiate yourself from the crowd of generalized data scientists. As a networking or IT professional, this area of mental models is where you find use cases that set you apart from others. Teaching data science to you is likely to be much easier and quicker than finding data scientists and teaching them what you know.

We build our mental models over time through repetition, which for you means hands-on experience in networking and IT. I use the term *hands-on* here to distinguish between active engagement and simple time in role. We all know folks who coast through their jobs; they have fewer and different mental models than the people who actively engage, or *deliberately practice*, as Gladwell puts it.

Earlier chapters of this book compare overlays on a network to a certain set of roads you use to get to work. Assuming that you have worked in the same place for a while, because you have used those roads so many times, you have built a mental model of what a *normal* commute looks like. Can you explain the turns you took today, the number of stop signs you encountered, and the status of the traffic lights? If the trip was uneventful, then probably not. In this case, you made the trip through intuition, using your "autopilot." If there was an accident at the busiest intersection of your routine trip, however, and you had to take a detour, you would remember the details of this trip.

When something changes, it grabs your attention and forces you to apply a mental spotlight to it so that you can complete the desired goal (getting to work in this case). Every detailed troubleshooting case you have worked on in your career has been a mental model builder. You have learn how things *should* work and now, while troubleshooting, you can recall your mental models and diagrams to determine where you have a deviation from the "known good" in your head. Every case strengthens your mental models.

My earliest recollection of using my mental models at work was during a data center design session for a very large enterprise customer. A lot of architecture and planning

work had been put in over the previous year, and a cutting-edge data center design was proposed by a team from Cisco. The customer was on the path to developing a detailed low-level design (LLD) from the proposed high-level architecture (HLA). The customer accepted the architecture, and Cisco Services was building out the detailed design and migration plans; I was the newly appointed technical lead. On my first day with the customer, in my first meeting with the customer's team, I stood in front of the entire room of 20-plus people and stated aloud, "I don't like this design." Ouch. Talk about foot in mouth. ...I had forgotten to engage the filter between my mental model and my mouth.

First, let me tell you that this was *not* the proper way to say, "I have some reservations about what you are planning to deploy" (which they had been planning for a year). At dinner that evening, my project manager said that there was a request to remove me from the account as a technical lead. I said that I was okay with that because I was not going to be the one to deploy a design that did not fit the successful mental models in my head. I was in meetings all day, and I needed to do some research, but something in my data center design mental models was telling me that there was an issue with this design. Later that night, I confirmed the issue that was nagging me and gathered the necessary evidence required to present to the room full of stakeholders.

The next day, I presented my findings to the room full of arms-crossed, leaned-back-in-chairs engineers, all looking to roast the new guy who had called their baby ugly in front of management the previous day. After going through the technical details, I was back in the game, and I kept my technical lead role. All the folks on the technical team agreed that the design would not have worked, given my findings. There was a limitation in the spanning-tree logical port/MAC table capacity of the current generation of switches. This limitation would have had disastrous consequences had the customer deployed this design in the highly virtualized data center environment that was planned.

The design was changed. After the deployment and migration was successful for this data center, two more full data centers with the new design were deployed over the next three years. The company is still running much of this infrastructure today. I had a mental model that saved years of suboptimal performance and a lot of possible downtime and enabled a lot of stability and new functionality that is still being used today.

Saving downtime is cool, but what about the analytics, you ask? Based on this same mental model, anytime I evaluate a customer data center, I now know to check MAC addresses, MAC capacity, logical ports, virtual LANs (VLANs), and many other Layer 2 networking factors from my mental models. I drop them all into a simple "descriptive analytics" table to compare the top counts in the entire data center. Based on experience, much of this is already in my head, and I intuitively see when something is not right—when some ratio is wrong or some number is too high or too low.

How do you move from a mental model to predictive analytics? Do you recall the next steps in the phases of analytics in Chapter 1, "Getting Started with Analytics"? Once you know the reasons based on diagnostic analytics, you can move to predictive analytics as a next possible step by encoding your knowledge into mathematical models or algorithms. On the analytics maturity curve, you can move from simple proactive to predictive once you build these models and algorithms into production. You can then add fancy analytics

models like logistic regression or autoregressive integrated moving average (ARIMA) to predict and model behaviors, and then you can validate what the models are showing. Since I built my mental model of a data center access design, I have been able to use it hundreds of times since then and for many purposes.

As an innovative thinker in your own area of expertise, you probably have tens or hundreds of these mental models and do not even realize it. This is your prime area for innovation. Take some time and make a list of the areas where you have spent detailed time and probably have a strong mental model. Apply anomaly detection on your own models, from your own head, and also apply what-if scenarios. If you are aware of current challenges or business problems in your environment, mentally run through your list of mental models to see if you can apply any of them.

This chapter introduces different aspects of the brain and your cognitive thinking processes. If your goal here is to identify and gather innovative use cases, as the book title suggests, then now is a good time to pause and write down any areas of your own expertise that have popped into your mind while reading this section. Write down anything you just “know” about these environments as possible candidates for future analysis. Try to move your mode of thinking all over the place in order to find new use cases but do not lose track of any of your existing ones along the way. When you are ready, continue with the next section, which takes a deeper dive into mental models.

Daniel Kahneman’s System 1 and System 2

Where does the concept of mental models come from? In his book *Thinking Fast and Slow* (a personal favorite), Daniel Kahneman identifies this expert intuition—common among great chess players, fire fighters, art dealers, expert drivers, and video game-savvy kids—as one part of a simple two-part mental system. This intuition happens in what Kahneman calls *System 1*. It is similar to Gladwell’s concept of *deliberate practice*, which Gladwell posits can lead to becoming an expert in anything, given enough time to develop the skills. You have probably experienced this as *muscle memory*, or *intuition*. You intuitively do things that you know how to do, and answers in the spaces where you are an expert just jump into your head. This is great when the models are right, but it is not so good when they are not.

What happens when your models are incorrect? Things can get a bit strange, but how might this manifest? Consider what would happen if the location of the keys on your computer keyboard were changed. How fast could you type? QWERTY keyboards are still in use today because millions of people have developed muscle memory for them. This can be related to Kahneman’s System 1, a system of autopilot that is built in humans through repetition, something called “cognitive muscle memory” when it is about you and your area of expertise.

Kahneman describes System 1 and System 2 in the following way: System 1 is intuitive and emotional, and it makes decisions quickly, usually without even thinking about it. System 2 is slower and more deliberate, and it takes an engaged brain. System 1, as you may suspect, is highly related to the mental models that have already discussed. As you’ll learn in the

next section, System 1 is also ripe for cognitive biases, commonly described as intuition but also known as prejudices or preconceived notions. Sometimes System 1 causes actions that happen without thinking, and other times System 2 is aware enough to stop System 1 from doing something that is influenced by some unconscious bias. Sometimes System 2 whiffs completely on stopping System 1 from using an unconsciously biased decision or statement (for example, my “I don’t like this design” flub). If you have a conscience, your perfect 20/20 hindsight usually reminds you of these instances when they are major.

Kahneman discusses how this happens, how to train System 1 to recognize certain patterns, and when to take appropriate actions without having to engage a higher system of thought. Examples of this System 1 at work are an athlete reacting to a ball or you driving home to a place where you have lived for a long time. Did you stop at that stop sign? Did you look for oncoming traffic when you took that left turn? You do not even remember thinking about those things, but here you are, safely at your destination.

If you have mental models, System 1 uses these models to do the “lookups” that provide the quick-and-dirty answers to your instinctive thoughts in your area of expertise, and it recalls them instantly, if necessary. System 2 takes more time, effort, and energy, and you must put your mind into it. As you will see in Chapter 6, in System 2 you can remain aware of your own thoughts and guide them toward metaphoric thinking and new perspectives.

Intuition

If you have good mental models, people often think that you have great *intuition* for finding things in your space. Go ahead, take the pat on the back and the credit for great intuition, because you have earned it. You have painstakingly developed your talents through years of effort and experience. In his book *Talent Is Overrated: What Really Separates World-Class Performers from Everybody Else*, Geoff Colvin says that a master level of talent is developed through deliberate and structured practice; this is reminiscent of Duhigg and Gladwell. As mentioned earlier, Gladwell says it takes 10,000 hours of deliberate practice with the necessary skills to be an expert at your craft. You might also say that it takes 10,000 hours to develop your mental models in the areas where you heavily engage in your own career. Remember that deliberate practice is not the same as simple time-in-job experience. Colvin calls out a difference between practice and experience. For the areas where you have a lot of practice, you have a mental model to call upon as needed to excel at your job. For areas where you are “associated” but not engaged, you have experience but may not have a mental model to draw upon.

How do you strengthen your mental models into intuition? Obviously, you need the years of active engagement, but what is happening during those years to strengthen the models? Mental models are strengthened using lots of what-if questions, lots of active brain engagement, and many hours of hands-on troubleshooting and fire drills. This means not just reading about it but actually doing it. For those in networking, the what-if questions are a constant part of designing, deploying, and troubleshooting the networks that you run every day. Want to be great at data science? Define and build your own use cases.

So where do mental models work against us? Recall the CRT questions from earlier in the chapter. Mental models work against you when they provide an answer too quickly, and your thinking brain (System 2) does not stop them. In such a case, perhaps some known bias has influenced you. This chapter explores many ways to validate what is coming from your intuition and how cognitive biases can influence your thinking. The key point of the next section is to be able to turn off the autopilot and actively engage and think—and write down—any new biases that you would like to learn more about. To force this slow-down and engagement, the following section explores cognitive bias and how it manifests in you and your stakeholders, in an effort to force you into System 2 thinking.

Opening Your Mind to Cognitive Bias

What is meant by *cognitive bias*? Let's look at a few more real-world examples that show how cognitive bias has come up in my life.

My wife and I were riding in the car on a nice fall trip to the Outer Banks beaches of North Carolina. As we travelled through the small towns of eastern North Carolina, getting closer and closer to the Atlantic, she was driving, and I was in the passenger seat, trying to get some Cisco work done so I could disconnect from work when we got to the beach. A few hours into the trip, we entered an area where the speed limit dropped from 65 down to 45 miles per hour. At this point, she was talking on the phone to our son, and when I noticed the speed change, I pointed to the speed limit sign to let her know to slow down a bit to avoid a speeding ticket. A few minutes later the call ended, and my wife said that our son had gotten a ticket.

So what are you thinking right now? What was I thinking? I was thinking that my son had gotten a speeding ticket, because my entire situation placed the speeding ticket context into my mind, and I consumed the information “son got a ticket” in that context. Was I right in thinking that? Obviously not, or this would be a boring story to use here. So what really happened?

At North Carolina State University, where my son was attending engineering school, getting student tickets to football games happens by lottery for the students. My son had just found out that he got tickets to a big game in the recent lottery—not a speeding ticket.

Can you see where my brain filled in the necessary parts of a story that pointed to my son getting a speeding ticket? The context had biased my thinking. Also add the “priming effect” and “anchoring bias” as possibilities here. (All this is discussed later in this chapter.)

My second bias story is about a retired man, Donnie, from my wife's family who invited me to go golfing with him at Lake Gaston in northeastern North Carolina many years ago. I was a young network engineer for Cisco at the time, and I was very happy and excited to see the lake, the lake property, and the lush green golf course. Making conversation while we were golfing, I asked Donnie what he did for a living before he retired to his life of leisure, fishing and golfing at his lake property. Donnie informed me that he was a retired engineer.

Donnie was about 20 years older than I, and I asked him what type of engineer he was before he retired. Perhaps a telecom engineer, I suggested. Maybe he worked on the old phone systems or designed transmission lines? Those were the only systems that I knew of that had been around for the 20 years prior to that time.

So what was Donnie's answer? "No, John," Donnie said. "I drove a train!"

Based on my assumptions and my bias, I went down some storyline and path in my own head, long before getting any details about what kind of engineer Donnie was in his working years. This could have led to an awkward situation if I had been making any judgments about train drivers versus network engineers. Thankfully, we were friendly enough that he could stop me before I started talking shop and making him feel uncomfortable by getting into telecom engineering details.

What bias was this? Depending on how you want to tell the story to yourself, you could assign many names. A few names for this type of bias may be recency bias (I knew engineers who had just retired), context bias (I am an engineer), availability bias (I made a whole narrative in my head based on my available definition of an engineer), or mirroring bias (I assumed that *engineer* in Donnie's vocabulary was the same as in mine). My brain grasped the most recent and available information to give me context to what I just heard and then it wrote a story. That story was wrong. My missing System 2 filter did not stop the "Were you a telecom engineer?" question.

These are a couple of my own examples of how easy it is to experience cognitive bias. It is possible that you can recall some of your own because they are usually memorable. You will encounter many different biases in yourself and in your stakeholders. Whether you are trying to expand your mind to come up with creative analytics solution opportunities in your areas of SME or proposing to deploy your newly developed analytics solution, these biases are present. For each of the biases explored in this section, some very common ways in which you may see them manifest in yourself or your stakeholders are identified. While you are reading them, you may also recognize other instances from your own world about you and your stakeholders that are not identified. It is important to understand how you are viewing things, as well as how your stakeholders may be viewing the same things. Sometimes these views are the same, but on occasion they are wildly different. Being able to take their perspective is an important innovation technique that allows you to see things that you may not have seen before.

Changing Perspective, Using Bias for Good

Why is there a whole section of this book on bias? Because you need to understand where and how you and your stakeholders are experiencing biases, such as functional fixedness, where you see the items in your System 1, your mental models, as working only one way. With these biases, you are trapped inside the box that you actually want to think outside. Many, many biases are at play in yourself and in those for whom you are developing solutions.

Your bias can make you a better data scientist and a better SME, or it can get you in trouble and trap you in that box of thinking. Cognitive bias can be thought of as a prejudice in your mind about the world around you. This prejudice influences how you perceive things. When it comes to data and analysis, this can be dangerous, and you must try to avoid it by proving your impressions. When you use bias to expand your mind for the sake of creativity, bias can provide some interesting opportunities to see things from new perspectives. Exploring bias in yourself and others is an interesting trigger for expanding the mind for innovative thinking.

If seeing things from a new perspective allows you to be innovative, then you need to figure out how to take this new perspective. Bias represents the unconscious perspectives you have right now—perspective from your mental models of how things are, how stuff works, and how things are going to play out. If you call these unintentional thoughts to the surface, are they unintentional any longer? Now they are real and palpable, and you can dissect them.

As discussed earlier in this chapter, it is important to identify your current context (mental models) and perspectives on your area of domain expertise, which drive any job-related biases that you have and, in turn, influence your approach to analytics problems in your area of expertise. Analytics definitions are widely available, and understanding your own perspective is important in helping you to understand why you gravitate to specific parts of certain solutions. As you go through this section, keep three points top of mind:

- Understanding your *own* biases is important in order to be most effective at using them or losing them.
- Understanding your *stakeholder* bias can mean the difference between success and failure in your analytics projects.
- Understanding bias in *others* can bring a completely new perspective that you may not have considered.

The next few pages explain each of the areas of bias and provide some relevant examples to prepare you to broaden your thought process as you dig into the solutions in later chapters. You will find mention of bias in statistics and mathematics. The general definition there is the same: some prejudice that is pulling things in some direction. The bias discussed here is cognitive, or brain-related bias, which is more about insights, intuitions, insinuations, or general impressions that people have about what the data or models are going to tell them. There are many known biases, and in the following sections I cluster selected biases together into some major categories to present a cohesive storyline for you.

Your Bias and Your Solutions

What do you do about biases? When you have your first findings, expand your thinking by reviewing possible bias and review your own assumptions as well as those of your stakeholders against these findings. Because you are the expert in your domain, you can recognize whether you need to gather more data or gather more proof to validate your findings. Nothing counters bias like hard data, great analytics, and cool graphics.

In some cases, especially while reading this book, some bias is welcome. This book provides industry use cases for analytics, which will bring you to a certain frame of mind, creating something of a new context bias. Your bias from your perspective will certainly be different from those of others reading this same book. You will probably apply your context bias to the use cases to determine how they best fit your own environment. Some biases are okay—and even useful when applied to innovation and exploration. So let’s get started reviewing biases.

How You Think: Anchoring, Focalism, Narrative Fallacy, Framing, and Priming

This first category of biases, which could be called *tunnel vision*, is about your brain using something as a “true value,” whether you recognize it or not. It may be an anchor or focalism bias that lives in the brain, an imprint learned from experiences, or something put there using mental framing and priming. All of these lead to you having a rapid recall of some value, some comparison value that your brain fixates on. You then mentally connect the dots and sometimes write narrative fallacies that take you off the true path.

A bias that is very common for engineers is anchoring bias. *Anchoring* is the tendency to rely too heavily, or “anchor,” on one trait or piece of information when making decisions. It might be numbers or values that were recently provided or numbers recalled from your own mental models. Kahneman calls this the *anchoring effect*, or *preconceived notions* that come from System 1. Anchors can change your perception of an entire situation. Say that you just bought a used car for \$10,000. If your perceived value, your anchor for that car, was \$15,000, you got a great deal in your mind. What if you check the true data and find that the book value on that car is \$20,000? You still perceive that you got a fantastic deal—an even better deal than you thought. However, if you find that the book value is only \$9,000, you probably feel like you overpaid, and the car now seems less valuable. That book value is your new anchor. You paid \$10,000, and that should be the value, but your perception of the car value and your deal value is dependent on the book value, which is your anchor. See how easily the anchor changes?

Now consider your anchors in networking. You cannot look up these anchors, but they are in your mental models from your years of experience. Anchoring in this context is the tendency to mentally predict some value or quantity without thinking. For technical folks, this can be extremely valuable, and you need to recognize it when it happens. If the anchor value is incorrect, however, this can result in a failure of your thinking brain from stopping your perceiving brain.

In my early days as a young engineer, I knew exactly how many routes were in a customer’s network routing tables. Further, because I was heavily involved in the design of these systems, I knew how many neighbors each of the major routers should have in the network. When troubleshooting, my mental model had these anchor points ingrained. When something did not match, it got raised to my System 2 awareness to dig in a little further. (I also remember random and odd phone numbers from years ago, so I have to take the good with the bad in my system of remembering numbers.)

Now let's consider a network operations example of anchoring. Say that you have to make a statement to your management about having had five network outages this month. Which of the following statements sounds better?

- “Last month we had 2 major outages on the network, and this month we had 5 major outages”.
- “Last month we had 10 major outages, and this month we had 5 major outages.”

The second one sounds better, even though the two options are reporting the same number of outages for this month. The stakeholder interest is in the current month's number, not the past. If you use past values as anchors for judgment, then the perception of current value changes. It is thus possible to set an anchor—some value to use by which to compare the given number.

In the book *Predictably Irrational*, behavioral economist Dan Ariely describes the anchoring effect as “the fallacy of supply and demand.” Ariely challenges the standard of how economic supply and demand determine pricing. Instead, he posits that your anchor value and perceived value to you relative to that anchor value determines what you are willing to pay. Often vendors supply you that value, as in the case of the manufacturer's suggested retail price (MSRP) on a vehicle. As long as you get under MSRP, you feel you got a good buy. Who came up with MSRP as a comparison? The manufacturers are setting the anchor that you use for comparison. The fox is in the henhouse.

Assuming that you can avoid having anchors placed into your head and that you can rely on what you know and can prove, where can your anchors from mental models fail you? If you are a network engineer who must often analyze things for your customers, these anchors that are part of your bias system can be very valuable. You *intuitively* seem to know quite a bit about the environment, and any numbers pulled from systems within the environment get immediately compared to your mental models, and your human neural network does immediate analysis. Where can this go wrong?

If you look at other networks and keep your *old* anchors in place, you could hit trouble if you sense that your anchors are correct when they are not. I knew how many routes were in the tables of customers where I helped to design the network, and from that I built my own mental model anchor values of how many routes I expected to see in routing tables in networks of similar size. However, when I went from a customer that allowed tens of thousands of routes to a customer that had excellent filtering and summarization in place, I felt that something was missing every time I viewed a routing table that had only hundreds of entries. My mental models screamed out that somebody was surely getting black hole routed somewhere. Now my new mental models have a branch on the “routing table size” area with “filtered” and “not filtered” branches.

What did I just mean by “black hole routed”? Black hole routing, when it is unexpected, is one of the worst conditions that can happen in computer networks. It means that some network device, somewhere in the world, is pulling in the network traffic and routing it into a “black hole,” meaning that it is dropped and lost forever. I was going down yet another bias rat hole when I considered that black hole routing was the issue at my new

client's site. Kahneman describes this as *narrative fallacy*, which is again a preconceived notion, where you use your own perceptions and mental models to apply plausible and probable reasons to what can happen with things as they are. Narrative fallacy is the tendency to assign a familiar story to what you see; in the example with my new customer, missing routes in a network typically meant black hole routing to me. Your brain unconsciously builds narratives from the information you have by mapping it to mental models that may be familiar to you; you may not even realize it is happening.

When something from your area of expertise does not map easily to your mental model, it stands out—just like the way those routes stood out as strange to me, and my brain wanted to assign a quick “why” to the situation. In my old customer networks, when there was no route and no default, the traffic got silently dropped; it was black hole routed. My brain easily built the narrative that having a number of routes that is too small surely indicates black hole routing somewhere in the network.

Where does this become problematic? If you see something that is incorrect, your brain builds a quick narrative based on the first information that was known. If you do not flag it, you make decisions from there, and those decisions are based on bad information. In the case of the two networks I first mentioned in this section, if my second customer network had had way too many routes when I first encountered it because the filtering was broken somewhere, I would not have intuitively seen it. My mental model would have led me to believe that a large number of routes in the environment was quite normal, just as with my previous customer's network.

The lesson here? Make sure you base your anchors on real values, or real base-rate statistics, and not on preconceived notions from experiences or anchors that were set from other sources. From an innovation perspective, what can you do here? For now, it is only important that you recognize that this happens. Challenge your own assumptions to find out if you are right with real data.

Another bias-related issue is called the *framing effect*. Say that you are the one reporting the monthly operational case data from the previous section. By bringing up the data from the previous month of outages, you set up a frame of reference and force a natural human comparison, where people compare the new numbers with the anchor that you have conveniently provided for them. Going from only a few outages to 5 is a big jump! Going from 10 outages to 5 is a big drop! This is further affected by the priming effect, which involves using all the right words to prime the brain for receiving the information. Consider these two sentences:

- We had two outages this week.
- We had two business-impacting outages this week

There is not very much difference here in terms of reporting the same two outages, but one of these statements primes the mind to think that the outages were bad. Add the anchors from the previous story, and the combination of priming with anchors allows your biased stakeholders to build quite a story in their brains.

How do you break out of the anchoring effect? How do you make your analytics solutions more interesting for your stakeholders if you are concerned that they will compare to existing anchors? Ariely describes what Starbucks did. Starbucks was well aware that consumers compared coffee prices to existing anchor prices. How did that change? Starbucks changed the frame of reference and made it not about coffee but about the experience. Starbucks even changed the names of the sizes, which created further separation from the existing anchor of what a “large cup of coffee” should cost. Now when you add the framing effect here, you make the Starbucks visit about coffee house ambiance rather than about a cup of coffee. Couple that with the changes to the naming, and you have removed all ability for people to compare to their anchors. (Biased or not, I do like Starbucks coffee.)

In your newly developed analytics-based solution, would you rather have a 90% success rate or a 10% failure rate? Which one comes to mind first? If you read carefully, you see that they mean the same thing, but the positive words sound better, so you should use these mechanisms when providing analysis to your stakeholders. Most people choose the framing 90% success rate because it sets up a positive-sounding frame. The word *success* initiates a positive priming effect.

How Others Think: Mirroring

Now that we’ve talked about framing and priming, let’s move our bias discussion from how to perceive information to the perception of how others perceive information. One of the most important biases to consider here is called *mirror-image bias*, or *mirroring*.

Mirroring bias is powerful, and when used in the wrong way, it can influence major decisions that impact lives. Philip Mudd discusses a notable case of mirroring bias in his book *Head Game*. Mudd recalls a situation in which the CIA was trying to predict whether another country would take nuclear testing action. The analysts generally said no. The prediction turned out to be incorrect, and the foreign entity did engage in nuclear testing action. Somebody had to explain to the president of the United States why the prediction was incorrect. The root cause was actually determined to be bias in the system of analysis.

Even *after* the testing action was taken, the analysts determined that, given the same data, they would probably make the “no action” prediction again. Some other factor was at play here. What was discovered? Mirroring bias. The analysts assumed that the foreign entity thought just as they did and would therefore take the same action they would, given the same data about the current conditions.

As an engineer, a place where you commonly see mirroring bias is where you are presenting the results of your analytics findings, and you believe the person hearing them is just as excited about receiving them as you are about giving them. You happily throw up your charts and explain the numbers—but then notice that everybody in the room is now buried in their phones. Consider that your audience, your stakeholders, or anyone else who will be using what you create may not think like you. The same things that excite you may not excite them.

Mirroring bias is also evident in one-on-one interactions. In the networking world, it often manifests in engineers explaining the tiny details about an incident on a network to someone in management. Surely that manager is fascinated and interested in the details of the Layer 2 switching and Layer 3 routing states that led to the outage and wants to know the exact root cause—right? The yawn and glassy eyes tell a different story, just like the heads in phones during the meeting.

As people glaze over during your stories of Layer 2 spanning-tree states and routing neighbor relationships, they may be trying to relate parts of what you are saying to things in their mental models, or things they have heard recently. They draw on their own areas of expertise to try to make sense of what you are sharing. This brings up a whole new level of biases—biases related to expertise in you and others.

What Just Happened? Availability, Recency, Correlation, Clustering, and Illusion of Truth

Common biases around expertise are heavily related to the mental models and System 1 covered earlier in this chapter. Availability bias has your management presentation attendees filling in any gaps in your stories from their areas of expertise. The area of expertise they draw from is often related to recency, frequency, and context factors.

People write their narrative stories with the availability bias. Your brain often performs in a last-in, first-out (LIFO) way. This means that when you are making assumptions about what might have caused some result that you are seeing from your data, your brain pulls up the most recent reason you have heard and quickly offers it up as the reason for what you now see. This can happen for you and for your stakeholders, so a double bias is possible.

Let's look at an example. At the time of this writing, terrorism is prevalent in the news. If you hear of a plane crash, or a bombing, recency bias may lead you to immediately think that an explosion or a plane crash is terrorism related. If you gather data about all explosions and all major crashes, though, you will find that terrorism is not the most likely cause of such catastrophes. Kahneman notes that this tendency involves not relying on known good, base-rate statistics about what commonly happens, even though these base-rate statistics are readily available. Valid statistics show that far fewer than 10% of plane crashes are related to terrorism. Explosion and bombing statistics also show that terrorism is not a top cause. However, you may reach for terrorism as an answer if it is the most recent explanation you have heard. Availability bias created by mainstream media reporting many terrorism cases brings terrorism to mind first for most people when they hear of a crash or an explosion.

Let's bring this back into IT and networking. In your environment, if you have had an outage and there is another outage in the same area within a reasonable amount of time, your users assume that the cause of this outage is the same as the last one because IT did not fix it properly. So not only do you have to deal with your own availability bias, you have to deal with bias in the stakeholders and consumers of the solutions that you are building. *Availability* refers to something that is top of mind and is the first available answer in the LIFO mechanism that is your brain.

Humans are always looking for cause–effect relationships and are always spotting patterns, whether they exist or not. So be careful with the analytics mantra that “correlation is not causation” when your users see patterns. If you are going to work with data science, learn, rinse, and repeat “*Correlation is not causation!*” Sometimes there is no narrative or pattern, even if it appears that there is. Consider this along with the narrative bias covered previously—the tendency to try to make stories that make sense of your data, make sense of your situation. Your stakeholders take what is available and recent in their heads, combine it with what you are showing them, and attempt to construct a narrative from it. You therefore need to have the data, analytics, tools, processes, and presentations to address this up front, as part of any solutions you develop. If you do not, cognitive ease kicks in, and stakeholders will make up their own narrative and find comfortable reasons to support a story around a pattern they believe they see.

Let’s go a bit deeper into correlation and causation. An interesting case commonly referenced in the literature is the correlation of an increase in ice cream sales with an increase in drowning deaths. You find statistics that show when ice cream sales increase, drowning deaths increase at an alarmingly high rate. These numbers rise and fall together and are therefore correlated when examined side by side. Does this mean that eating ice cream causes people to drown? Obviously not. If you dig into the details, what you probably recognize here is that both of these activities increase as the temperature rises in summer; therefore, at the same time the number of accidental drowning deaths rises because it is warm enough to swim, so does the number of people enjoying ice cream. There is indeed correlation, but neither one causes the other; there is no cause–effect relationship.

This ice cream story is a prime example of a correlation bias that you will experience in yourself and your stakeholders. If you bring analytics data, and stakeholders correlate it to something readily available in their heads due to recency, frequency, or simple availability, they may assign causation. You can use questioning techniques to expand their thinking and break such connections.

Correlation bias is common. When events happen in your environment, people who are aware of those events naturally associate them with events that seem to occur at the same time. If this happens more than a few times, people make the connection that these events are somehow related, and you are now dealing with something called the *availability cascade*. Always seek to prove causation when you find correlation of events conditions, or situations. If you do not, your biased stakeholders might find them for you and raise them at just the wrong time or make incorrect assumptions about your findings.

Another common bias, clustering bias, further exacerbates false causations. *Clustering bias* involves overestimating the importance of small patterns that appear as runs, streaks, or clusters in samples of data. For example, if two things happen at the same time a few times, stakeholders associate and cluster them as a common event, even if they are entirely unrelated.

Left unchecked, these biases can grow even more over time, eventually turning into an *illusion of truth effect*. This effect is like a snowball effect, in that people are more likely to believe things they previously heard, even if they cannot consciously remember having

heard them. People will believe a familiar statement over an unfamiliar one, and if they are hearing about something in the IT environment that has negative connotation for you, it can grow worse as the hallway conversation takes it on. The legend will grow.

The illusion of truth effect is a self-reinforcing process in which a collective belief gains more and more plausibility through its increasing repetition (or “repeat something long enough, and it will become true”). As new outages happen, the statistics about how bad the environment might be are getting bigger in people’s heads every time they hear it. A common psychology phrase used here is “The emotional tail wags the rational dog.” People are influenced by specific issues recently in the news, and they are increasingly influenced as more reports are shared. If you have two or three issues in a short time in your environment, you may hear some describing it as a “meltdown.”

Your stakeholders hear of one issue and build some narrative, which you may or may not be able to influence with your tools and data. If more of the same type of outages occur, whether they are related to the previous one or not, your stakeholders *will* relate the outages. After three or more outages in the same general space, the availability cascade is hard to stop, and people are looking to replace people, processes, tools, or all of the above. Illusion of truth goes all the way back to the availability bias, as it is the tendency to overestimate the likelihood of events with greater availability in memory, which can be influenced by how recent the memories are or how unusual or emotionally charged they are. Illusion of truth causes untrue conditions or situations to seem like real possibilities. Your stakeholders can actually believe that the sky is truly falling after the support team experiences a rough patch.

This area of bias related to expertise is a very interesting area to innovate. Your data and analytics can show the real truth and the real statistics and can break cycles of bias that are affecting your environment. However, you need to be somewhat savvy about how you go about it. There are real people involved, and some of them are undoubtedly in positions of authority. This area also faces particular biases, including authority bias and the HIPPO impact.

Enter the Boss: HIPPO and Authority Bias

Assume that three unrelated outages in the same part of the network have occurred, and you didn’t get in front of the issue. What can you do now? Your biggest stakeholder is sliding down the availability cascade, thinking that there is some major issue here that is going to require some “big-boy decision making.” You assure him that the outages are not related, and you are analyzing the root cause to find out the reasons. However, management is now involved, and they want action that is contradicting what you want to do. Management also has opinions on what is happening, and your stakeholder believes them, even though your analytics are showing that your assessment is supported by solid data and analysis. Why do they not believe what is right in front of them?

Enter the highest paid persons’ opinion (HIPPO) impact and authority bias. *Authority bias* is the tendency to attribute greater accuracy to the opinion of an authority figure and to believe that opinion over others (including your own at times). As you build out solutions

and find the real reasons in your environments, you may confirm the opinions and impressions of highly paid people in your company—but sometimes you will contradict them. Stakeholders and other folks in your solution environment may support these biases, and you need solid evidence if you wish to disprove them. Sometimes people just “go with” the HIPPO opinion, even if they think the data is telling them something different. This can get political and messy. Tread carefully. Disagreeing with the HIPPO can be dangerous.

On the bright side, authority figures and HIPPOs can be a great source of inspiration as they often know what is hot in the industry and in management circles, and they can share this information with you so that you can target your innovative solutions more effectively. From an innovation perspective, this is pure gold as you can stop guessing and get real data about where to develop solutions with high impact.

What You Know: Confirmation, Expectation, Ambiguity, Context, and Frequency Illusion

Assuming that you do not have an authority issue, you may be ready to start showing off some cool analytics findings and awesome insights. Based on some combination of your brilliance, your experience, your expertise, and your excellent technical prowess, you come up with some solid things to share, backed by real data. What a perfect situation—until you start getting questions from your stakeholders about the areas that you did *not* consider. They may have data that contradicts your findings. How can that happen? For outages, perhaps you have some inkling of what happened, some expectation. You have also gone out and found data to support that expectation. You have mental models, and you recognize that you have an advantage over many because you are the SME, and you know what data supports your findings.

You know of some areas where things commonly break down, and you have some idea of how to build cool analytics solution with the data to show others what you already know, maybe with a cool new visualization or something. You go build that.

From an innovation perspective, your specialty areas are the first areas you should check out. These are the hypotheses that you developed, and you naturally want to find data that makes you right. All engineers want to find data that makes them right. Here is where you must be careful of confirmation bias or expectation bias. Because you have some preconceived notion of what you expect to see, some number strongly anchored in your brain, you are biased to find data and analytics to support your preconceived notion. Even simple correlations without proven causations suffice for a brain looking to make connections.

“Aha!” you say. “The cause of these outages is a bug in the software. Here is the evidence of such a bug.” This evidence may be a published notification from Cisco that the software running in the suspect devices is susceptible to this bug if memory utilization hits 99% on a device. You provide data showing that traffic patterns spiked on each of these outage days, causing the routers to hit that 99% memory threshold, in turn causing the network devices to crash. You have found what you expected to find, confirmed these findings with data, and gone back to your day job. What’s wrong with this picture?

As an expert in your IT domain, you often want to dive into use cases where you have developed a personal hypothesis about the cause of an adverse event or situation (“It’s a bug!”). When used properly, data and analytics can confirm your hypothesis and prove that you positively identified the root cause. However, remember that correlation is not causation. If you want to be a true analyst, you must perform the due diligence to truly prove or confirm your findings. Other common statements made in the analytics world include “You can interrogate the data long enough so that it tells you anything that you want to know” and “If you torture the data long enough, it will confess.” In terms of confirmation or expectation bias, if you truly want to put on blinders and find data to confirm what you think is true, you can often find it. Take the extra steps to perform any necessary validation in these cases because these are areas ripe for people to challenge your findings.

So back to the bug story. After you find the bug, you spend the next days, weeks, and months scheduling the required changes to upgrade the suspect devices so they don’t experience this bug again. You lead it all. There are many folks involved, lots of late nights and weekends, and then you finally complete the upgrades. Problem solved.

Except it is not. Within a week of your final upgrade, there are more device crashes. Recency, frequency, availability cascades...all of it is in play now. Your stakeholders are clear in telling you that you did not solve the problem. What has happened?

You used your skills and experience to confirm what you expected, and you looked no further. For a complete analysis, you needed to take alternate perspectives as well and try to prove your analysis incomplete or even wrong. This is simply following the scientific process: Prove the null hypothesis. Do not fall for confirmation bias—the tendency to search for, interpret, focus on, and remember information in a way that confirms your preconceptions. Did you cover all the bases, or were you subject to expectation bias? Say that you assumed that you found what you were looking for and got confirmation. Did you get real confirmation that it was the real root cause?

Yes, you found a bug, but you did not find the root cause of the outages. Confirmation bias stopped your analysis when you found what you wanted to find. High memory utilization on any electronic component is problematic. Have you ever experienced an extremely slow smartphone, tablet, or computer? If you turn such a device off and turn it back on, it works great again because memory gets cleared. Imagine this issue with a network device responsible for moving millions of bits of data per second. Full memory conditions can wreak all kinds of havoc, and the device may be programmed to reboot itself when it reaches such conditions, in order to recover from a low memory condition. Maybe the bug details documentation was stating this. The root cause is still out there. What causes the memory to go to 99%? Is it excessive traffic hitting the memory due to configuration? Was there a loop in the network causing traffic race conditions that pushed up the memory? The real root cause is related to what caused the 99% memory condition in the first place.

Much as confirmation bias and expectation bias have you dig into data to prove what you already know, ambiguity bias has you avoid doing analysis in areas where you don’t think there is enough information. Ambiguity in this sense means avoiding options for which

missing information makes the probability seem unknown. In the bug case discussed here, perhaps you do not have traffic statistics for the right part of the network, and you think you do not have the data to prove that there was a spike in traffic caused by a loop in that area, so you do not even entertain that as a possible part of the root cause. Start at the question you want answered. Ask your SME peers a few open-ended questions or go down the *why* chain. (You will learn about this in Chapter 6.)

Another angle for this is the experimenter's bias, which involves believing, certifying, and presenting data that agrees with your expectations for the outcome of your analysis and disbelieving, ignoring, or downgrading the interest in data that appears to conflict with your expectations. Scientifically, this is not testing hypotheses, not doing direct testing, and ignoring possible alternative hypotheses. For example, perhaps what you identified as the root cause was only a side effect and not the true cause. In this case, you may have seen from your network management systems that there was 99% memory utilization on these devices that crashed, and you immediately built the narrative, connected the dots from device to bug, and solved the problem!

Maybe in those same charts you saw a significant increase in memory utilization across these and some of the other devices. Some of those other devices went from 10% to 60% memory utilization during the same period, and the increased traffic showed across all the devices for which you have traffic statistics. As soon as you saw the “redline” 99% memory utilization, another bias hit you: Context bias kicked in as you were searching for the solution to the problem, and you therefore began looking for some standout value, blip on the radar, or bump in the night. And you found it. Context bias convinces you that you have surely found the root cause because it is exactly what you were looking to find. You were in the mode, or the context of looking for some know bad values.

I've referenced *context bias* more than a few times, but let's now pause to look at it more directly. A common industry example used for context bias is the case of grocery shopping while you are hungry. Shopping on an empty stomach causes you to choose items differently from when you go shopping after you have eaten. If you are hungry, you choose less healthy, quicker-to-prepare foods. As an SME in your own area of expertise, you know things about your data that other people do not know. This puts you in a different context than the general analyst. You can use this to your advantage and make sure it does not bias your findings. However, you need to be careful not to let your own context interfere with what you are finding, as in the 99% memory example.

Maybe your whole world is routing—and routers, and networks that have routers, and routing protocols. However, analysis that provides much-improved convergence times for WAN Layer 3 failover events is probably not going to excite a data center manager. In your context, the data you have found is pretty cool. In the data center manager's context? It's simply not cool. That person does not even have a context for it. So keep in mind that context bias can cut both ways.

Context bias can be set with priming, creating associations to things that you knew in the past or have recently heard. For example, if we talk about bread, milk, chicken, potatoes, and other food items, and I ask you to fill in the blank of the word *so_p*, what do you say? Studies show that you would likely say *soup*. Now, if we discuss dirty hands, grimy

faces, and washing your hands and then I ask you to fill in the blank in *so_p*, you would probably say *soap*. If you have outages in routers that cause impacts to stakeholders, they are likely to say that “problematic routers” are to blame. If your organization falls prey to the scenario covered in this section and have problematic routers more than a few times, the new context may become “incompetent router support staff.”

This leads to another bias, called *frequency illusion*, in which the frequency of an event appears to increase when you are paying attention to it. Before you started driving the car you now have, how many of them did you see on the road before you bought yours? How many do you see now? Now you have engaged your brain to recognize the car that you now drive, it sees and processes them all. You saw them before but did not process them. Back in the network example, maybe you have regular change controls and upgrades, and small network disruptions are normal as you go about standard maintenance activities. After two outages, however, you are getting increased trouble tickets and complaints from stakeholders and network users. Nothing has changed for you; perhaps a few minutes of downtime for change windows in some areas of the network is normal. But other people are now noticing every little outage and complaining about it. You know the situation has not changed, but frequency illusion in your users is at play now, and what you know may not matter to those people.

What You Don’t Know: Base Rates, Small Numbers, Group Attribution, and Survivorship

After talking about what you know, in true innovator fashion, let’s now consider the alternative perspective: what you do not know. As an analyst and an innovator, you always need to consider the other side—the backside, the under, the over, the null hypothesis, and every other perspective you can take. If you fail to take these perspectives, you end up with an incomplete picture of the problem. Therefore, understanding the foundational environment, or simple base-rate statistics, is important.

In the memory example, you discovered devices at 99% memory and devices at 60% memory. Your attention and focus went to the 99% items highlighted red in your tools. Why didn’t you look at the 60% items? This is an example of *base-rate neglect*. If you looked at the base rate, perhaps you would see that the 99% devices, which crashed, typically run at 65% memory utilization, so there was roughly a 50%+ increase in memory utilization, and the devices crashed. If you looked at the devices showing 60%, you would see that they typically run at 10%, which represents a 600% increase in utilization caused by the true event. However, because these devices did not crash, bias led you to focus on the other devices.

This example may also be related to the “law of small numbers,” where the characteristics of the entire population may be assumed by looking at just a few examples. Engineers are great at using intuition to agree with findings from small samples that may not be statistically significant. The thought here may be: “These devices experienced 99% memory utilization, and therefore all devices that hit 99% memory utilization will crash.”

You can get false models in your head by relying on intuition and small samples and relevant experience rather than real statistics and numbers. This gets worse if you are making decisions on insufficient data and incorrect assumptions, such as spending time and resources to upgrade entire networks based on a symptom rather than based on a root cause. Kahneman describes this phenomenon as “What You See Is All There Is” (WYSIATI) and cites numerous examples of it. People base their perception about an overall situation on the small set of data they have. Couple this with an incorrect or incomplete mental model, and you are subject to making choices and decisions based on incomplete information, or incorrect assumptions about the overall environment that are based on just a small set of observations. After a few major outages, your stakeholders will think the entire network is problematic.

This effect can snowball into identifying an entire environment or part of the network as suspect—such as “all devices with this software will crash and cause outage.” This may be the case even if you used redundant design most places and if this failure and clearing of memory in the routers is normal, and your design handles it very gracefully. There is no outage to upgrade in this case because of your great design, but because the issue is the same type that caused some other error, group attribution error may arise.

Group attribution error is the biased belief that the characteristics of an individual observation is representative of the entire group as a whole. Group attribution error is commonly related to people and groups such as races or genders, but this error can also apply to observations in IT networking. In the earlier 99% example, because these routers caused outage in one place in the network, stakeholders may think the sky is falling, and those devices will cause outages everywhere else as well.

As in an example earlier in this chapter, when examining servers, routers, switches, controllers, or other networking components in their own environment, network engineers often create new instances of mental models. When they look at other environments, they may build anchors and be primed by the values they see in the few devices they examine from the new environment. For example, they may have seen that 99% memory causes crash, which causes outage. So you design the environment to fail around crashes, and 99% memory causes a crash, but there is no outage. This environment does not behave the same as the entire group because the design is better. However, stakeholders want you to work nights and weekends to get everything upgraded—even though that will not fix the problem.

Take this group concept a step further and say that you have a group of routers that you initially do not know about, but you receive event notifications for major outages, and you can go look at them at that time. This is a group for which you have no data, a group that you do not analyze. This group may be the failure cases, and not the survivors. Concentrating on the people or things that “survived” some process and inadvertently overlooking those that did not because of their lack of visibility is called *survivorship bias*.

An interesting story related to survivorship bias is provided in the book *How Not to Be Wrong*, in which author Jordan Ellenberg describes the story of Abraham Wald and his study of bullet holes in World War II planes. During World War II, the government

employed a group of mathematicians to find ways to keep American planes in the air. The idea was to reduce the number of planes that did not return from missions by fortifying the planes against bullets that could bring them down.

Military officers gathered and studied the bullet holes in the aircraft that returned from missions. One early thought was that the planes should have more armor where they were hit the most. This included the fuselage, the fuel system, and the rest of the plane body. They first thought that they did not need to put more armor on the engines because they had the smallest number of bullet holes per square foot in the engines. Wald, a leading mathematician, disagreed with that assessment. Working with the Statistics Research Group in Manhattan, he asked them a question: “Where were the missing bullet holes?”

What was the most likely location? The missing bullet holes from the engines were on the missing planes. The planes that were shot down. The most vulnerable place was not where all the bullet holes were on the returning planes. The most vulnerable place was where the bullet holes were on the planes that did not return.

Restricting your measurements to a final sample and excluding part of the sample that did not survive creates survivorship bias. So how is the story of bullets and World War II important to you and your analytics solutions today? Consider that there has been a large shift to “cloud native” development. In cloud-native environments, as solution components begin to operate poorly, it is very common to just kill the bad one and spin up new instance of some service.

Consider the “bad ones” here in light of Wald’s analysis of planes. If you only analyze the “living” components of the data center, you are only analyzing the “servers that came back.” Consider the earlier example, in which you only examined the “bad ones” that had 99% memory utilization. Had you examined all routers from the suspect area, you would have seen the pattern of looping traffic across *all* routers in that area and realized that the crash was a side effect and not the root cause.

Assume now that you find the network loop, and you need to explain it at a much higher level now due to the visibility that the situation has gained. In this case, your expertise has related bias. What can happen when you try to explain the technical details from your technical perspective?

Your Skills and Expertise: Curse of Knowledge, Group Bias, and Dunning-Kruger

As an expert in your domain, you will often run into situations where you find it extremely difficult to think about problems from the perspective of people who are not experts. This is a common issue and a typical perspective for engineers that spend a lot of time in the trenches. This “curse of knowledge” allows you to excel in your own space but can be a challenge when getting stakeholders to buy in to your solutions, such as to understand the reasons for outage. Perhaps you would like to explain why crashes are okay in the highly resilient part of the network but have trouble articulating, in a nontechnical way, how the failover will happen. Further, when you show data and analytics proving that the failover works, it becomes completely confusing to the executives in the room.

Combining the curse of knowledge with in-group bias, some engineers have a preference for talking to other engineers and don't really care to learn how to explain their solutions in better and broader terms. This can be a major deterrent for innovation because it may mean missing valuable perspectives from members not in the technical experts group.

In-group bias is thinking that people you associate with yourself are smarter, better, and faster than people who are not in your group. A similar bias, *out-group bias*, is related to social inequality, where you see people outside your groups as less favorable than people within your groups. As part of taking different perspectives, how can you put yourself into groups that you perceive as out-groups in your stakeholder community and see things from their perspective?

In-group bias also involves group-think challenges. If your stakeholders are in the group, then great: Things might go rather easily for areas where you all think alike. However, you will miss opportunities for innovation if you do not take new perspectives from the out-groups. Interestingly, sometimes those new perspectives come from the inexperienced members in the group who are reading the recent blogs, hearing the latest news, and trying to understand your area of expertise. They “don't know what they don't know” and may reach a level of confidence such that they are very comfortable participating in the technical meetings and offering up opinions on what needs to be analyzed and how it should be done. This moves us into yet another area of bias, called the *Dunning-Kruger effect*.

The Dunning-Kruger effect happens when unskilled individuals overestimate their abilities while skilled experts underestimate theirs. As you deal with stakeholders, you may have plenty of young and new “data scientists” who see relationships that are not there, correlations without causations, and general patterns of occurrences that do not mean anything. You will also experience many domain SMEs with no data science expertise identifying all of the cool stuff “you could do” with analytics and data science. This might have been a young, talkative junior engineer taking all the airtime in the management meetings, when others knew the situation much, much better. That new guy was just dropping buzzwords and didn't not know the ins and outs of that technology, so he just talked freely. Ah, the good old days before you know about caveats....

Yes, the Dunning-Kruger effect happens a lot in the SME space, and this is where you can possibly gain some new perspective. Consider Occam's razor or the law of parsimony for analytics models. Sometimes the simplest models have the most impact. Sometimes the simplest ideas are the best. Even when you find yourself surrounded by people who do not fully grasp the technology or the science, you may find that they offer a new and interesting perspective that you have not considered—perspective that can guide you toward innovative ideas.

Many of the pundits in the news today provide glaring examples of the Dunning-Kruger effect. Many of these folks are happy to be interviewed, excited about the fame, and ready to be the “expert consultant” on just about any topic. However, real data and results trump pundits. As Kahneman puts it, “People who spend their time, and earn their living, studying a particular topic produce poorer predictions than dart-throwing monkeys who would distribute their choices evenly over the options.” Hindsight is not

foresight, and experience about the past does not give predictive superpowers to anyone. However, it can create challenges for you when trying to sell your new innovative models and systems to other areas of your company.

We Don't Need a New System: IKEA, Not Invented Here, Pro-Innovation, Endowment, Status Quo, Sunk Cost, Zero Price, and Empathy

Say that you build a cool new analytics-based regression analysis model for checking, trending, and predicting memory. Your new system takes live data from telemetry feeds and applies full statistical anomaly detection with full-time series awareness. You are confident that this will allow the company to preempt any future outages like the most recent ones. You are ready to bring it online and replace the old system of simple standard reporting because the old system has no predictive capabilities, no automation, and only rudimentary notification capabilities.

As you present this, your team sits on one side of the room. These people want to see change and innovation for the particular solution area. These people love the innovation, but as deeply engaged stakeholders, they may fail to identify any limitations and weaknesses of their new solution. For each of them, and you, because it is your baby, your creation, it must be cool. Earlier in this chapter, I shared a story of my mental model conflicting with a new design that a customer had been working on for quite some time. You and your team here and my customer and the Cisco team there are clear cases of *pro-innovation bias*, where you get so enamored with the innovation that you do not realize that telemetry data may not yet be available for all devices, and telemetry is the only data pipeline that you designed. You missed a spot. A big spot.

When you have built something and you are presenting it and you will own it in the future, you can also fall prey to the *endowment effect*, in which people who “own” something assign much more value to it than do people who do not own it. Have you ever tried to sell something? You clearly know that your house, car, or baseball card collection has a very high value, and you are selling it at what you think is a great price, yet people are not beating down your door as you thought they would when you listed it for sale. If you have invested your resources into something and it is your baby, you generally value it more highly than do people who have no investment in the solution. Unbeknownst to you, at the very same time the same effect could be happening with the folks in the room who own the solution you are proposing to replace.

Perhaps someone made some recent updates to a system that you want to replace. Even for partial solutions or incremental changes, people place a disproportionately high value on the work they have brought to a solution. Maybe the innovations are from outside vendors, other teams, or other places in the company. Just as with assembly of furniture from IKEA, regardless of the quality of the end result, the people involved have some bias toward making it work. Because they spent the time and labor, they feel there is intrinsic value, regardless of whether the solution solves a problem or meets a need. This is aptly named the *IKEA effect*. People love furniture that they assembled with their own hands. People love tools and systems that they brought online in companies.

If you build things that are going to replace, improve, or upgrade existing systems, you should be prepared to deal with the IKEA effect in stakeholders, peers, coworkers, or friends who created these systems. Who owns the existing solutions at your company? Assuming that you can improve upon them, should you try to improve them in place or replace them completely?

That most recent upgrade to that legacy system invokes yet another challenge. If time, money, and resources were spent to get the existing solution going, replacement or disruption can also hit the sunk cost fallacy. If you have had any formal business training or have taken an economics class, you know that a *sunk cost* is money already spent on something, and you cannot recover that money. When evaluating the value of a solution that they are proposing, people often include the original cost of the existing solution in any analysis. But that money is gone; it is sunk cost. Any evaluation of solutions should start with the value and cost from this point moving forward, and sunk costs should not be part of the equation. But they will be brought up, thanks to the sunk cost fallacy.

On the big company front, this can also manifest as the *not-invented-here syndrome*. People choose to favor things invented by their own company or even their own internal teams. To them, it obviously makes sense to “eat your own dessert” and use your own products as much as possible. Where this bias becomes a problem is when the not-invented-here syndrome causes intra-company competition and departmental thrashing because departments are competing over budgets to be spent on development and improvement of solutions. *Thrashing* in this context means constantly switching gears and causing extra work to try to shoehorn something into a solution just because the group responsible for building the solution invented it. With intra-company not-invented-here syndrome, the invention, initiative, solution, or innovation is often associated with a single D-level manager or C-level executive, and success of the individual may be tied directly to success of the invention. When you are developing solutions that you will turn into systems, try to recognize this at play.

This type of bias has another name: *status-quo bias*. People who want to defend and bolster the existing system exhibit this bias. They want to extend the life of any current tools, processes, and systems. “If it ain’t broke, why fix it?” is a common argument here, usually countered with “We need to be disruptive” from the other extreme. Add in the sunk cost fallacy numbers, and you will find yourself needing to show some really impressive analytics to get this one replaced. Many people do not like change; they like things to stay relatively the same, so they provide strong system justification to keep the existing, “old” solution in place rather than adopt your new solution.

Say that you get buy-in from stakeholders to replace an old system, or you are going to build something brand new. You have access to a very expensive analytics package that was showing you incredible results, but it is going to cost \$1000 per seat for anyone who wants to use it. Your stakeholders have heard that there are open source packages that do “most of the same stuff.” If you are working in analytics, you are going to have to deal with this one. Stakeholders hear about and often choose what is free rather than what you wanted if what you wanted has some cost associated with it.

You can buy some incredibly powerful software packages to do analytics. For each one of these, you can find 10 open source packages that do almost everything the expensive packages do. Now you may spend weeks making the free solution work for you, or you may be able to turn it around in a few hours, but the *zero price effect* comes into play anywhere there is an open source alternative available. The effect is even worse if the open source software is popular and was just presented at some show, some conference, or some meetup attended by your stakeholders.

What does this mean for you as an analyst? If there is a cloud option, or a close Excel tool, or something that is near what you are proposing, be prepared to try it out to see if it meets the need. If it does not, you at least have the justification you need to choose the package that you wanted, and you have the reasoning to justify the cost of the package. You need to have a prepared build-versus-buy analysis.

Getting new analytics solutions in place can be challenging, sometimes involving technical and financial challenges and sometimes involving political challenges. With political challenges, the advice I offer is to stay true to yourself and your values. Seek to understand why people make choices and support the direction they go. The tendency to underestimate the influence or strength of feelings, in either oneself or others, is often called an *empathy gap*. A empathy gap can result in unpleasant conversations after you are perceived to have called someone's baby ugly, stepped on toes, or showed up other engineers in meetings. Simply put, the main concern here is that if people are angry, they are more passionate, and if they are more passionate against you rather than for you, you may not be able to get your innovation accepted.

Many times, I have seen my innovations bubble up 3 to 5 years after I first work on them, as part of some other solution from some other team. They must have found my old work, or come to a similar conclusion long after I did. On one hand, that stinks, but on the other hand, I am here to better my company, and it is still internal, so I justify in my head that it is okay, and I feed the monster called *hindsight bias*.

I Knew It Would Happen: Hindsight, Halo Effect, and Outcome Bias

Hindsight bias and the similar outcome bias both give credit for decisions and innovations that just “happened” to work out, regardless of the up-front information the decision was based on. For example, people tend to recognize startup founders as geniuses, but in many stories you read about them, you may find that they just happened to luck into the right stuff at the right time. For these founders of successful startups, the “genius” moniker is sometimes well deserved, but sometimes it is just hindsight bias. When I see my old innovative ideas bubbling back up in other parts of the company or in related tools, I silently feed another “attaboy” to my hindsight monster. I may have been right, but conditions for adoption of my ideas at the earlier time were not.

What if you had funded some of the well-known startup founders in the early days of their ventures? Would you have spent your retirement money on an idea with no known history? Once a company or analytics solution is labeled as innovative, people tend to recognize that anything coming from the same people must be innovative because a *halo*

effect exists in their minds. However, before these people delivered successful outcomes that biased your hindsight to see them as innovative geniuses, who would have invested in their farfetched solutions?

Interestingly, this bias can be a great thing for you if you figure out how to set up innovative experimenting and “failing fast” such that you can try a lot of things in a short period of time. If you get a few quick wins under your belt, the halo effect works in your favor. If something is successful, then the hindsight bias may kick in. Sometimes called the “I-knew-it-all-along” effect, hindsight bias is the tendency to see past events as being predictable at the time those events happened. Kahneman also describes hindsight and outcome bias as “bias to look at the situation now and make a judgment about the decisions made to arrive at this situation or place.”

When looking at the inverse of this bias, I particularly like Kahneman’s quote in this area: “Actions that seemed prudent in foresight can look irresponsibly negligent in hindsight.” I’d put it like this: “It seemed like a good idea at the time.” These results bring unjust rewards to “risk takers” or those who simply “got lucky”. If you try enough solutions through your innovative experimentation apparatus, perhaps you will get lucky and have a book written about you. Have you read stories and books about successful people or companies? You probably have. Such books sell because their subjects are successful, and people seek to learn how they got that way. There are also some books about why people or companies have failed. In both of these cases, hindsight bias is surely at play. If you were in the same situations as those people or companies when they made their fateful decisions, would you have made the same decisions without the benefit of the hindsight that you have now?

Summary

In this chapter, you have learned about cognitive biases. You have learned how they manifest in you and your stakeholders. Your understanding of these biases should already be at work, forcing you to examine things more closely, which is useful for innovation and creative thinking (covered in Chapter 6). You can expand your own mental models, challenge your preconceived notions, and understand your peers, stakeholders, and company meetings better. Use the information in Table 5-1 as a quick reference for selected biases at play as you go about your daily job.

Table 5-1 *Bias For and Against You*

Bias	Working Against You	Working for You
Anchoring, focalism	Stakeholders have an anchor that is not in line with yours.	Question and explore their impressions, their anchor values, and compare to yours.
Priming	Inadequate detail and negative connotation exist about where analytics help.	Understand their priming. Prime your context where your solution works best.
Imprinting	Things happen the same as they always have.	Find and present data and insight proving that change is possible.

Bias	Working Against You	Working for You
Narrative fallacy	This involves initial impressions about how things are and connecting unconnected dots.	Find and explain the correlations that are not causations. Find the real causes
Mirroring	You assume people see things as you do and do not correct your course. There is a communications gap.	Seek first to understand and then to be understood.
Availability	People rely on their LIFO first impressions.	Uncover and understand the reasons for their impressions. Learn their top of mind.
Recency, frequency	People have top-of-mind impressions based on recent events.	Find the statistics to prove whether it is or is not the norm.
Correlation is not causation	Co-occurrence of things triggers human pattern recognition and story creation.	Share the ice cream story, find the true causes, show that base-rate information also correlates.
HIPPO, authority	You work on things not critical to the success of your chain of command.	Take some time to learn the key players and learn what is important from the key players.
Confirmation, expectation, congruence	You torture the data until it shows exactly what you want to show.	Take another perspective and try to disprove what you have proven.
Experimenter's bias	You do not test enough to validate what you find. Others call that out.	Test the alternative hypothesis and take the other side. Try to prove yourself wrong.
Belief	As Occam's razor says, the simplest and most plausible answer is probably correct to people.	Research and find facts and real data from the problem domain to disprove beliefs.
Context	Current, top-of-mind conditions influence thinking.	Understand their perspective and context and walk a mile in their shoes.
Frequency illusion	It's crashing all the time now.	Create a new frequency. You are always coming up with cool new solutions.
Base-rate neglect, law of small numbers	Stakeholders observe an anomaly in a vacuum and take that to be the norm.	Find out where the incorrect data originated and bring the full data for analysis.
Survivorship bias	All systems seem to be just fine.	Include systems that were not included in the analysis for a complete picture.

Bias	Working Against You	Working for You
Group attribution error	This type of device has nothing but problems, and we need to replace them all.	Find the real root causes of problems.
WYSIATI (What You See Is All There Is)	Based on what we saw in our area, the network behaves this way when issue X happens.	Explore alternatives to the status quo. Show differences in other areas with real data and analysis.
Curse of knowledge	Key stakeholders tune you out because you do not speak the same language.	Technical storytelling to nontechnical audiences is a skill that will take you far. Learn to use metaphors and analogies.
Similarity	You leave important people out of the conversation and speak to technical in-group peers only.	Use analytics storytelling to cover both technical and nontechnical audiences. Listen.
Dunning-Kruger	Inexperienced people steer the conversation to their small area of knowledge. You lose the room.	Include the inexperienced people, teach them, and make them better. Learn new perspectives from their “left field” comments that resonate.
IKEA, not invented here, endowment	People fight to keep analysis methods and tools that they worked on instead of accepting your new innovations.	People you include in the process will feel a sense of ownership with you.
Pro-innovation	People are more excited about what they are inventing than about what you are inventing.	People you include in the process will feel a sense of ownership with you.
Status quo, sunk cost fallacy	Some people just do not like change and disruption, even if it is positive. They often call up sunk costs to defend their position.	Be prepared with a long list of benefits, uses, and financial impacts of your innovation.
Empathy gap	You are an engineer, and often the technology is more exciting to you than are the people.	Read and learn this section of the chapter again to raise awareness of others’ thinking and reasoning.
Outcome bias, hindsight bias	Many companies have historically spent a lot of money on analytics that did not produce many useful models.	Highlight all the positive outcomes and uses of your innovation, especially long after you begin working on the next one.
Halo effect	Because you are new at data science, you don’t have a halo in this space.	Build and deploy a few useful models for your company, and your halo grows.

Innovative Thinking Techniques

There are many different opinions about innovation in the media. Most ideas are not new but rather have resulted from altering atomic parts from other ideas enough that they fit into new spaces. Think of this process as mixing multiple Lego sets to come up with something even cooler than anything in the individual sets. Sometimes this is as easy as seeing things from a new perspective. Every new perspective that you can take gives you a broader picture of the context in which you can innovate.

It follows that a source of good innovation is being able to view problems and solutions from many perspectives and then choose from the best of those perspectives to come up with new and creative ways to approach your own problems. To do this, you must first know your own space well, and you must also have some ability to break out of your comfort zone (and biases). Breaking out of a “built over a long time” comfort zone can be especially difficult for technical types who learn how to develop deep focus. Deep focus can manifest as tunnel vision when trying to innovate.

Recall from Chapter 5, “Mental Models and Cognitive Bias,” that once you know about something and you see and process it, it will not trip you up again. When it comes to expanding your thinking, knowing about your possible bias allows you to recognize that it has been shaping your thinking. This recognition opens up your thought processes and moves you toward innovative thinking. The goal here is to challenge your SME personality to stop, look, and listen—or at least slow down enough to expand upon the knowledge that is already there. You can expand your knowledge domain by forcing yourself to see things a bit differently and to think like not just an SME but also an innovator.

This chapter explores some common innovation tips and tricks for changing your perspective, gaining new ideas and pathways, and opening up new channels of ideas that you can combine with your mental models. This chapter, which draws on a few favorite techniques I have picked up over the years, discusses proven success factors used by successful innovators. The point is to teach you how to “act like an innovator” by discussing the

common activities employed by successful innovators and looking at how you can use these activities to open up your creative processes. If you are not an innovator yet, try to “fake it until you make it” in this chapter. You will come out the other side thinking more creatively (how much more creatively varies from person to person).

What is the link between innovation and bias? In simplest terms, *bias* is residual energy. For example, if you chew a piece of mint gum right now, everything that you taste in the near future is going to taste like mint until the bias the gum has left on your taste buds is gone. I believe you can use this kind of bias to your advantage. Much like cleansing the palette with sherbet between courses to remove residual flavors, if you bring awareness of bias to the forefront, you can be aware enough to know that taste may change. Then you are able to adjust for the flavor you are about to get. Maybe you want to experiment now with this mint bias. Try the chocolate before the sherbet to see what mint-chocolate flavor tastes like. That is innovation.

Acting Like an Innovator and Mindfulness

Are you now skeptical of what you know? Are you more apt to question things that you just intuitively knew? Are you thoughtfully considering why people in meetings are saying what they are saying and what their perspectives might be, such that they could say that? I hope so. Even if it is just a little bit. If you can expand your mind enough to uncover a single new use case, then you have full ROI (return on investment) for choosing this book to help you innovate.

In their book *The Innovator's DNA: Mastering the Five Skills of Disruptive Innovators*, Dyer, Gregersen, and Christensen describe five skills for discovering innovative ways of thinking: associating, questioning, observing, experimenting, and networking. You will gain a much deeper understanding of these techniques by adding that book to your reading list. This chapter includes discussion of those techniques in combination with other favorites and provides relevant examples for how to use them.

Now that Chapter 5 has helped you get your mind to this open state, let's examine innovation techniques you can practice. “Fake it till you make it” does not generally work well in technology because technology is complex, and there are many concrete facts to understand. However, innovation takes an open mind, and if “acting like an innovator” opens your mind, then “fake it till you make it” is actually working for you. Acting like an innovator is simply a means to an end for you—in this case, working toward 10,000 hours of practicing the skills for finding use cases so that you can be an analytics innovator.

What do you want to change? What habits are stopping you from innovating? Here is a short list to consider as you read this section and Chapter 7, “Analytics Use Cases and the Intuition Behind Them”:

- Recognize your tunnel vision, intuition, hunches, and mental models. Use them for metaphoric thinking. Engage Kahneman's System 2 and challenge the first thought that pops into your head when something new is presented to you.

- Challenge everything you know with *why* questions. Why is it that way? Can it be different? Why does the solution use the current algorithm instead of other options? Why did your System 1 give that impression? What narrative did you just construct about what you just learned?
- Slow down and recognize your framing, your anchoring, and other biases that affect the way you are thinking. Try to supply some new anchors and new framing using techniques described in this chapter. Now what is your new perspective? What “Aha!” moments have you experienced?
- Use triggering questions to challenge yourself. Keep a list handy to run through them as you add knowledge of a new opportunity for innovation. The “five whys” engineering approach, described later in this chapter, is a favorite of many.
- Get outside perspectives by reading everything you can. Printed text, audio, video, and any other format of one-way information dissemination is loosely considered reading. Learn and understand both sides of each area, the pros and the cons, the for and the against. What do the pundits say? What do the noobs say? Who really knows what they are talking about? Who has opinions that prompt you to think differently?
- Get outside perspectives by interactively talking to people. I have talked to literally hundreds of people within Cisco about analytics and asked for their perspectives on analytics. In order to develop a common talking model, I developed the analytics infrastructure model and began to call analytics solutions *overlays* for abstraction purposes. In many of my conversations, although people were talking from different places in the analytics infrastructure model, they were all talking about areas of the same desired use case.
- Relax and give your creative side some time. Take notes to read back later. The most creative ideas happen when you let things simmer for a while. Let the new learning cook with your old knowledge and wisdom. Why do the best ideas come to you in the shower, in the car, or lying in bed at night? New things are cooking. Write them down as soon as you can for later review.
- Finally, practice the techniques you learn here and read the books that are referenced in this chapter and Chapter 5. Read them again. Practice some more. Remember that with 10,000 hours of deliberate practice, you can become an expert at anything. For some it will occur sooner and for others later. However, I doubt that anyone can develop an innovation superpower in just a few hundred hours.

Innovation Tips and Techniques

So how do you get started? Let’s get both technical and abstract. Consider that you and your mental models are the “model” of who you are now and what you know. Given that you have a mathematical or algorithmic “model” of something, how can you change

the output of that model? You change the inputs. This chapter describes techniques for changing your inputs. If you change your inputs, you are capable of producing new and different outputs. You will think differently. Consider this story:

You are flying home after a very long and stressful workweek at a remote location. You are tired and ready to get home to your own bed. You are at the airport, standing in line at the counter to try to change your seat location. At the front of the long line, a woman is taking an excessive amount of time talking to the airline representative. She talks, the representative gets on the phone, she talks some more, then more phone calls for the representative. You are getting annoyed. To make things worse, the woman's two small children begin to get restless and start running around playing. They are very loud, running into some passengers' luggage, and yet the woman is just standing there, waiting on the representative to finish the phone call.

After a few excruciatingly long minutes, one giggling child pushes the other into *your* luggage, knocking it over. You are very angry that this woman is letting her children behave like this without seeming to notice how it is affecting the other people in line. You leave your luggage lying on the floor at your place in line and walk to the front. You demand that the woman do something about her unruly children. Consider your anger, perception, and perspective on the situation right at this point.

She never looks at you while you are telling her how you feel. You get angrier. Then she slowly turns toward you and speaks. "I'm so sorry, sir. Their father has been severely injured in an accident while working abroad. I am arranging to meet his medical flight on arrival here, and we will fly home as a family. I do not know the gate. I have not told the children why we are here."

Is your perception and perspective on this situation still the same?

Metaphoric Thinking and New Perspectives

Being able to change your perspective is a critical success factor for innovation. Whether you do it through reading about something or talking to other people, you need to gain new perspectives to change your own thinking patterns. In innovation, one way to do this is to look at one area of solutions that is very different from your specialty area and apply similar solutions to your own problem space. A common way of understanding an area where you may (or may not) have a mental map is achieved through something called *metaphoric thinking*. As the name implies, metaphoric thinking is the ability to think in metaphors, and it is a very handy part of your toolbox when you explore existing use cases, as discussed in Chapter 7.

So how does metaphoric thinking work? For cases where you may not have mental models, a "push" form of metaphoric thinking is a technique that involves using your existing knowledge and trying to apply it in a different area. From a network SME perspective, this is very similar to trying to think like your stakeholders. Perhaps you are an expert in network routing, and you know that every network data packet needs a destination,

or the packet will be lost because it will get dropped by network routers. How can you think of this in metaphoric terms to explain to someone else?

Let's go back to the driving example as a metaphor for traffic moving on your network and the car as a metaphor for a packet on your network. Imagine that the car is a network packet, and the routing table is the Global Positioning System (GPS) from which the network packet will be getting directions. Perhaps you get into the car, and when you go to engage the GPS, it has no destination for you, and you have no destination by default. You will just sit there. If you were out on the road, the blaring honks and yells from other drivers would probably force you to pull off to the side of the road. In network terms, a packet that has no destination must be removed so that packets that do have destinations can continue to be forwarded. You can actually count the packets that have missing destinations in any device where this happens as a forwarding use-case challenge. (Coincidentally, this is black hole routing.)

Let's go a step further with the traffic example. On some highways you see HOV (high-occupancy vehicle) lanes, and in theme parks you often see "fast pass" lanes. While everyone else is seemingly stuck in place, the cars and people in these lanes are humming along at a comfortable pace. In networking, quality of service (QoS) is used to specify which important traffic should go first on congested links. What defines "important"? At a theme park, you can pay money to buy a fast pass, and on a highway, you can save resources by sharing a vehicle with others to gain access to the HOV lane. In either case, you are more *important* from a traffic perspective because you have a premium value to the organization. Perhaps voice for communication has premium value on a network. In a metaphorical sense, these situations have similar solutions: Certain network traffic is more important, and there are methods to provide preferential treatment.

Thinking in metaphors is something you should aspire to do as an innovator because you want to be able to go both ways here. Can you take the "person in a car that is missing directions" situation and apply it to other areas in data networking? Of course. For routing use cases, this might mean dropping packets. Perhaps in switching use cases, it means packets will flood. If you apply network flooding to a traffic metaphor, this means your driver simply tries to drive on every single road until someone comes out of a building to say that the driver has arrived at the right place. Both the switching solution and its metaphorical counterpart are suboptimal.

Associative Thinking

Associating and metaphorical thinking are closely related. As you just learned, metaphorical thinking involves finding metaphors in other domains that are generally close to your problem domain. For devices that experience some crash or outage, a certain set of conditions lead up to that outage. Surely, these devices showed some predisposition to crashing that you should have seen. In a metaphorical sense, how do doctors recognize that people will "crash"? Perhaps you can think like a doctor who finds conditions in a person that indicate the person is predisposed to some negative health event. (Put this idea in your mental basket for the chapters on use cases later in this book.)

Associating is the practice of connecting dots between seemingly unrelated areas. Routers can crash because of a memory leak, which leads to resource exhaustion. What can make people crash? Have you ever dealt with a hungry toddler? If you have, you know that very young people with resource exhaustion do crash.

Association in this case involves using resemblance and causality. Can you find some situation in some other area that resembles your problem? If the problem is router crashing, what caused that problem? Resource exhaustion. Is there something similar to that in the people crashing case? Sure. Food provides energy for a human resource. How do you prevent crashes for toddlers? Do not let the resources get too low: Feed the toddler. (Although it might be handy, there is no software upgrade for a toddler.) Prevention involves guessing when the child (router) will run low on energy resources (router memory) and will need to resupply by eating (recovering memory). You can predict blood sugar with simple trends learned from the child's recent past. You can predict memory utilization from a router's recent past.

Six Thinking Hats

Metaphoric and associative thinking are just a couple of the many possible ways to change your mode of thinking. Another option is to use a lateral thinking method, such as Edward de Bono's "six thinking hats" method. The goal of six thinking hats is to challenge your brain to take many different perspectives on something in order to force yourself to think differently. This section helps you understand the six hats thinking approach so you can add it to your creative toolbox.

A summary perception of de Bono's six colored hats is as follows:

- **Hat 1**—A white hat is the information seeker, seeking data about the situation.
- **Hat 2**—A yellow hat is the optimist, seeking the best possible outcome.
- **Hat 3**—A black hat is the pessimist, looking for what could go wrong.
- **Hat 4**—A red hat is the empath, who goes with intuition about what could happen.
- **Hat 5**—A green hat is the creative, coming up with new alternatives.
- **Hat 6**—A blue hat is the enforcer, making sure that every other hat is heard.

To take the six hats thought process to your own space, imagine that different stakeholders who will benefit from your analytics solutions each wear one of these six different hats, describing their initial perspective. Can you put yourself in the shoes of these people to see what they would want from a solution? Can you broaden your thinking while wearing their hat in order to fully understand the biases they have, based on situation or position?

If you were to transition from the intended form of multiple hats thinking by adding positional nametags, who would be wearing the various hats, and what nametags would they be wearing? As a starting point, say that you are wearing a nametag and a hat.

Instead of using de Bono's colors, use some metaphoric thinking and choose new perspectives. Who is wearing the other nametags? Some suggestions:

- **Nametag 1**—This is you, with your current perspective.
- **Nametag 2**—This is your primary stakeholder. Is somebody footing the bill? How does what you want to build impact that person in a positive way? Is there a downside?
- **Nametag 3**—This represents your primary users. Who is affected by anything that you put into place? What are the positive benefits? What might change if everything worked out just as you wanted it to?
- **Nametag 4**—This is your boss. This person supported your efforts to work on this new and creative solution and provided some level of guidance along the way. How can you ensure that your boss is recognized for his or her efforts?
- **Nametag 5**—This is your competition. What could you build for your company that would scare the competition? How can you make this tag very afraid?
- **Nametag 6**—This is your uninformed colleague, your child, or your spouse. How would you think about and explain this to someone who has absolutely no interest? What is so cool about your new analytics insight?

With a combination of 6 hats and 6 nametags, you can now mentally browse 36 possible perspectives on the given situation. Keep a notepad nearby and continue to write down the ideas that come to mind for later review. You can expand on this technique as necessary to examine all sides, and you may end up with many more than 36 perspectives.

Crowdsourcing Innovation

Crowdsourcing is getting new ideas from a large pool of people by using the wisdom and experience of the crowd. Crowdsourcing is used heavily in Cisco Services, where the engineers are exposed to a wide variety of situations, conditions, and perspectives. Many of these perspectives from customer-facing engineers are unknown to those on the incubation and R&D teams. The crowd knows some of the unknown unknowns, and crowdsourcing can help make them known unknowns. Analytics can help make them known knowns.

The engineers are the *internal* crowd, the internal network of people. Just as internal IT networks can take advantage of public clouds, crowdsourcing makes public crowds available for you to find ideas. (See what I did there with metaphoric thinking?) In today's software world, thanks to GitHub, slide shares, stack overflows, and other code and advice repositories, finding people who have already solved your problem, or one very similar to it, is easier than ever before. If you are able to think metaphorically, then this becomes even easier. When you're dealing with analytics, you can check out some public competitions (for example, see <https://www.kaggle.com/>) to see how things have been done, and then you can use the same algorithms and methodologies for your solution.

Internal to your own organization, start bringing up analytics in hallway conversations. If you want to get new perspectives from external crowdsourcing, go find a meetup or a conference. Maybe it is the start of a new trend, or perhaps it's just a fad, but the number of technology conferences available today is astounding. Nothing is riper for gaining new perspectives than a large crowd of individuals assembled in one place for a common tool or technology. I always leave a show, a conference, or a meetup with a short list of interesting things that I want to try when I get back to my own lab.

I have spent many hours walking conference show floors, asking vendors what they are building, why they are building it, and what analytics they are most proud of in the product they are building. In some cases, I have been impressed, and in others, not so much. When I say “not so much,” I am not judging but looking at the analytics path the individual is taking in terms of whether I have already explored that avenue. Sometimes other people get no further than my own exploration, and I realize the area may be too saturated for use cases. My barrier to entry is high because so much low-hanging fruit is already available. Why build a copy if you can just leverage something that's readily available? When something is already available, it makes sense to buy and use that product to provide input to your higher-level models rather than spend your time building the same thing again. Many companies face this “build versus buy” conundrum over and over again.

Networking

Crowdsourcing involves networking with people. The biggest benefit of networking is not telling people about your ideas but hearing their ideas and gaining new perspectives. You already have your perspective. You can learn someone else's by practicing active listening. After reading about the use cases in the next chapter, challenge yourself to research them further and make them the topic of conversation with peers. You will have your own biased view of what is cool in a use case, but your peers may have completely different perspectives that you may have not considered.

Networking is one of the easiest ways to “think outside the box” because having simple conversations with others pulls you to different modes of thinking. Attend some idea networking conferences in your space—and perhaps some outside your space. Get new perspectives by getting out of your silo and into others, where you can listen to how people have addressed issues that are close to what you often see in your own industry. Be sure to expand the diversity of your network by attending conferences and meetups or having simple conversations that are *not* in your core comfort areas. Make time to network with others and your stakeholders. Create a community of interest and work with people who have different backgrounds. Diversity is powerful.

Watch for instances of outliers everywhere. Stakeholders will most likely bring you outliers because nobody seeks to understand the common areas. If you know the true numbers, things regress to the mean (unless a new mean was established due to some change). Was there a change? What was it?

Questions for Expanding Perspective After Networking

After a show or any extended interaction, do not forget the hats and nametags. You may have just found a new one. The following questions are useful for determining whether you truly understand what you have heard; if you want to explore something later, you must understand it when you are getting the initial interaction:

- Did the new perspective give you an idea? How would your manager view this? Assuming that it all worked perfectly, what does it do for your company?
- How would you explain this to your spouse if your spouse does not work in IT? How can you create a metaphor that your spouse would understand? Spouses and longtime friends are great sounding boards. Nobody gives you truer feedback.
- How would you explain it to your children? Do you understand the innovation, idea, or perspective enough to create a metaphor that anyone can understand?
- For solutions that include people or manual processes, how can you replace these people and processes with devices, services, or components from your areas of expertise. Recall the example of a doctor diagnosing people, which you can apply to diagnosing routers. Does it still work?
- For solutions that look at clustering, rating, ranking, sorting, and prioritizing segments of people and things, do the same rules apply to your space? Can you find suitable replacements?

More About Questioning

Questioning has long been a great way to increase innovation. One obvious use of questioning as an innovative technique is to understand all aspects of solutions in other spaces that you are exploring. This means questioning every part in detail until you fully understand both the actual case and any metaphors that you can map to your own space. Let's continue with the simple metaphor used so far. Presume that, much as you can identify a sick person by examining a set of conditions, you can identify a network device that is sick by examining a set of parameters. Great. Now let's look at an example involving questioning an existing solution that you are reviewing:

- What are the parameters of humans that can indicate that the human is predisposed to a certain condition?
- Are there any parameters that clearly indicate “not exposed at all”? What is a “healthy” device?
- Are there any parameters that are just noise and have no predictive value at all? How can you avoid these imposters (such as shoe size having predictive value for illness)?
- How do you know that a full set of the parameters has been reached? Is it possible to reach a full set in this environment? Are you seeing everything that you need to see? Are you missing some bullet holes?

- Is it possible that the example you are reviewing is an outlier and you should not base all your assumptions on it? Are you seeing all there is?
- Is there a known root cause for the condition? For the device crash?
- If you had perfect data, what would it look like?
- Assuming that you had perfect data, what would you expect to find? Can you avoid expectation bias and also prove that there are no alternative answers that are plausible to your stakeholders?
- How would the world change if your analytics solution worked perfectly? Would it have value? Would this be an analytics Rube Goldberg?
- What is next? Assuming that you had a perfect analytics solution to get the last data point, how could you use that later? Could this be a data point in a new, larger ensemble analysis of many factors?
- Can you make it work some other way? What caused it to work the way it is working right now? Can you apply different reasoning to the problem? Can you use different algorithms?
- Are you subject to Kahneman's "availability heuristic" for any of your questions about the innovation? Are you answering any of the questions in this important area based on connecting mental dots from past occurrences that allow you to make nice neat mental connections and assignments, or do you know for sure? Do you have some bad assumptions?
- Are you adding more and more examples as "availability cascades" to reinforce any bad assumptions? Can you collect alternative examples as well to make sure your models will provide a full view? What is the base rate?
- Why develop the solution this way? What other ways could have worked? Did you try other methods that did not work?
- Where could you challenge the status quo? Where could you do things entirely differently?
- What constraints exist for this innovation? Where does the logic break down? Does that logic breakdown affect what you want to do?
- What additional constraints could you impose to make it fit your space? What constraints could you remove to make it better?
- What did you assume? How can you validate assumptions to apply them in your space?
- What is the state of the art? Are you looking at the "old way" of solving this problem? Are there newer methods now?
- Is there information about the code, algorithms, methods, and procedures that were used, so that you could readily adapt them to your solution?

Pay particular attention to the Rube Goldberg question. Are you taking on this problem because of an availability cascade? Is management interest in this problem due to a recent set of events? Will that interest still be there in a month? If you spend your valuable time building a detailed analysis, a model, and a full deployment of a tool, will the problem still exist when you get finished? Will the hot spot, the flare-up, have flamed out by the time you are ready to present something? Recall the halo bias, where you have built up some credibility in the eyes of stakeholders by providing useful solutions in the past. Do not shrink your earned halo by building solutions that consume a lot of time and provide low value to the organization. Your time is valuable.

CARESS Technique

You generally get great results by talking to people and using active listening techniques to gain new perspectives on problems and possible solutions. One common listening technique is CARESS, which stands for the following:

- **Concentrate**—Concentrate on the speaker and tune out anything else that could take your attention from what the speaker is saying.
- **Acknowledge**—Acknowledge that you are listening through verbal and nonverbal mechanisms to keep the information flowing.
- **Research and respond**—Research the speaker’s meaning by asking questions and respond with probing questions.
- **Emotional control**—Listen again. Practice emotional control throughout by just listening and understanding the speaker. Do not make internal judgments or spend time thinking up a response while someone else is still speaking. Jot down notes to capture key points for later responses so they do not consume your mental resources.
- **Structure**—Structure the big picture of the solution in outline form, mentally or on paper, such that you can drill down on areas that you do not understand when you respond.
- **Sense**—Sense the nonverbal communication of the speaker to determine which areas may be particularly interesting to that person so you can understand his or her point of reference.

Five Whys

“Five whys” is a great questioning technique for innovation. This popular technique is common in engineering contexts for getting to the root of problems. Alternatively, it is valuable for drilling into the details of any use case that you find. Going back to the network example with the crashed router due to a memory leak, the diagram in Figure 6-1 shows an example of a line of questioning using the five whys.

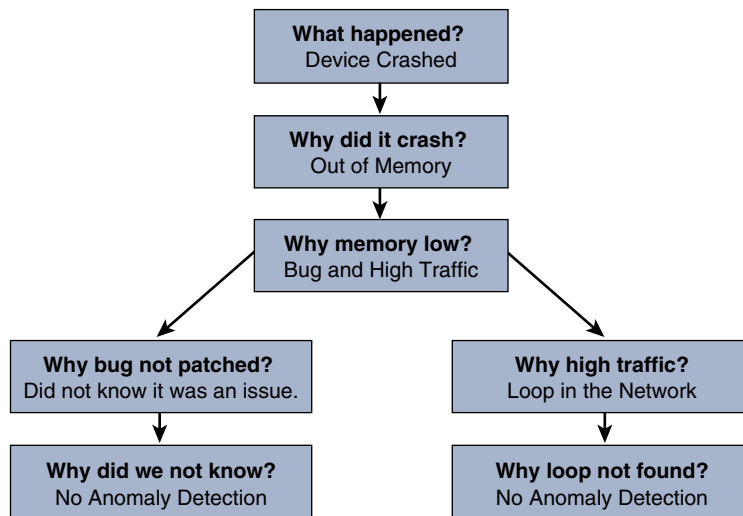


Figure 6-1 *Five Whys Question Example*

With five simple “why” questions, you can uncover two areas that lead you to an analytics option for detecting the router memory problem. Each question should go successively deeper, as illustrated in the technique going down the left path in the figure:

1. Question: What happened?
Answer: A router crashed.
2. Question: Why did it crash?
Answer: Investigation shows that it ran out of memory.
3. Question: Why did it run out of memory?
Answer: Investigation shows there is a memory leak bug published.
4. Question: Why did we not apply the known patch?
Answer: Did not know we were affected.
5. Question: Why did we not see this?
Answer: We do not have memory anomaly detection deployed.

Observation

Earlier in this chapter, in the section “Metaphoric Thinking and New Perspectives,” I challenged you to gain new perspectives through thinking and meeting people. That section covers how to uncover ideas, gain new perspectives, apply questions, and associate similar solutions to your space. What next? Now you watch (sometimes this is “virtual

watching”) to see how the solution operates. Observe things to see what works and what does not work—in your space and in others’ spaces. Observe the entire process, end to end. Do intense observation into the component parts of tasks to get something done. This observation is important when you get to the use cases portion of this book, which goes into detail about popular use cases in industry today. Research and observe how interesting solutions work. Recall that *observed* and *seen* are not the same thing, although they may seem synonymous. Make sure that you are understanding how the solutions work in detail.

Observing is also a fantastic way to strengthen and grow your mental models. “Wow, I have never seen that type of device used for that type of purpose.” Click: A new Lego just snapped onto your model for that device. Now you can go back to questioning mode to add more Legos about how the solution works. Observing is interesting when you can see Kahneman’s WYSIATI (What You See Is All There Is) and law of small numbers in action. People sometimes build an entire tool, system, or model on a very small sample or “perfect demo” version. When you see this happening, it should lead you to a more useful model of identifying, quantifying, qualifying, and modeling the behavior of the entire population.

Inverse Thinking

Another prime area for innovation is using questioning for inverse thinking. *Inverse thinking* is asking “What’s *not* there?” For example, if you are counting hardware MAC addresses on data center edge switches, what about switches that are not showing any MAC addresses? Sometimes “BottomN” is just as interesting as “TopN.”

Consider the case of a healthy network that has millions of syslog messages arriving at a syslog server. TopN shows some interesting findings but is usually the common noise. In the case of syslog, rare messages are generally more interesting than common TopN. Going a step further in the inverse direction, if a device sends a well-known number of messages every day, and then you do not receive any messages from that device for a day, what happened? Thinking this way is a sort of “inverse anomaly detection.”

If your organization is like most other organizations, you have expert systems. There are often targets for those expert systems to apply expertise, such as a configuration item in a network. Here again the “inverse” is a new perspective. If you looked at all your configuration lines within the company, how many would you find are *not* addressed by your expert systems? What configuration lines do *not* have your expert opinion? Should they? As you consider your mental models for what is, don’t forget to employ inverse thinking and also ask “What is not?” or “What is missing?” as other possible areas for finding insight and use cases for your environment.

Orthodoxies are defined as things that are just known to be true. People do not question them, and they use this knowledge in everyday decisions and as foundations for current biases. Inverse thinking can challenge current assumptions. Yes, maybe something “has always been done that way (status quo bias),” but you might determine that

there is a better way. Often attributed to Henry Ford, but actually of unknown origin is the statement, “If I had asked people what they wanted, they would have said faster horses.” Sometimes stakeholders just do not know that there is a better way. Can you find insights that challenge the status quo? Where are “things different” now? Can you develop game-changing solutions to capitalize on newly available technologies, as Henry Ford did with the automobile?

Developing Analytics for Your Company

Put down this book for a bit when you are ready to innovate. Why? After you have read the techniques here, as well as the use cases, you need some time to let these things simmer in your head. This is the process of defocusing. Step away for a while. Try to think up things by not thinking about things. You know that some of the best ideas of your career have happened in the strangest places; this is where defocusing comes in. Go take a shower, take a walk, exercise, run, or find some downtime during your vacation. Read the data and let your brain have some room to work.

Defocusing, Breaking Anchors, and Unpriming

If you enter a space that’s new to you, you will have a “newbie mindset” there. Can you develop this same mindset in your space? Active listening during your conversations with friends and family members who are patient enough to listen to your technobabble helps tremendously in this effort. This is very much akin to answering the question “If you could do it all over again from the beginning, how would you do it now?”

Take targeted reflection time—perhaps while walking, doing yardwork, or tackling projects around the house. With any physical task that you can do on autopilot, your thinking brain will be occupied with something else. Often ideas for innovations come to me while doing home repairs, making a batch of homebrew, or using my smoker. All of these are things that I enjoy that are very slow moving and provide chunks of time when I must watch and wait for steps of the process.

Defocusing can help you avoid “mental thrashing.” Do not be caught thrashing mentally by looking at too many things and switching context between them. *Computer thrashing* occurs when the computer is constantly switching between processes and threads, and each time it switches, it may have to add and remove things from some shared memory space. This is obviously very inefficient. So what are you doing when you try to “slow path” everything at once? Each thing you bring forward needs the attention of your own brain and the memory space for you to load the context, the situation, and what you know so far about it. If you have too many things in the slow path, you may end up being very ineffective.

Breaking anchors and unpriming is about recognizing your biases and preconceived notions and being able to work with them or work around them, if necessary. Innovation is only one area where this skill is beneficial. This is a skill that can make the world a better place.

Experimenting

Compute is cheap, and you know how to get data. Try stuff. Fail fast. Build prototypes. You may be able to use parts of others solutions to compose solutions of your own. You can use “Lego parts” analytics components to assemble new solutions.

Seek emerging trends to see if you can apply them in your space. If they are hot in some other space, how will they affect your space? Will they have any impacts? If you catch an *availability cascade*—a growing mental or popularity hot spot in your area of expertise—what experiments can you run through to produce some cool results?

As discussed in Chapter 5, the law of small numbers, the base rate fallacy, expectation bias, and many other biases that produce anchors in you or your stakeholders may just be incorrect. How can you avoid these traps? One interesting area of analytics is outlier analysis. If you are observing an outlier, why is it an outlier?

As you gain new knowledge about ways to innovate, here are some additional factors that will matter to stakeholders. For any possible use cases that grab your attention, apply the following lenses to see if anything resonates:

- Can you enable something new and useful?
- Can you create a unique value chain?
- Can you disrupt something that already exists in a positive way?
- Can you differentiate something from you or your company from your competitors?
- Can you create or highlight some new competitive advantage?
- Can you enable new revenue streams for your company?
- Can you monetize your innovation, or is it just good to know?
- Can you increase productivity?
- Can you increase organization effectiveness or efficiency?
- Can you optimize operations?
- Can you lower operational expenditures in a measurable way?
- Can you lower capital expenditures in a measurable way?
- Can you simplify how you do things or make something run better?
- Can you increase business agility?
- Can you provide faster time to market for something? (This includes simply “faster time to knowing” for network events and conditions.)
- Can you lower risk in a measurable way?

- Can you increase engagement of stakeholders, customers, or important people inside your own company?
- Can you increase engagement of customers or important people outside your company?
- What can you infer from what you know now? What follows?

Lean Thinking

You have seen the “fail fast” phrase a few times in the book. In his book *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, Eric Ries provides guidance on how an idea can rapidly move through phases, such that you can learn quickly whether it is a feasible idea. You can “fail fast” if it is not. Ries says, “We must learn what customers really want, not what they say they want or what we think they should want.” Apply this to your space but simply change *customers* to *stakeholders*. Use your experience and learning from the other techniques to develop hypotheses about what your stakeholders really need. Do not build them faster horses.

Experimenting (and not falling prey to experimenter’s bias) allows you to uncover the unknown unknowns and show your stakeholders insights they have not already seen. Using your experience and SME skills, determine if these insights are relevant. Using testing and validation, you can find the value in the solution that provides what your stakeholder wanted as well as what you perceived they needed.

The most important nugget from Ries is his advice to “pivot or persevere.” Pivoting, as the name implies, is changing direction; persevering is maintaining course. In discussing your progress with your stakeholders and users, use active listening techniques to gauge whether you are meeting their needs—not just the stated needs but also the additional needs that you hypothesized would be very interesting to them. Observe reactions and feedback to determine whether you have hit the mark and, if so, what parts hit the mark. Pivot your efforts to the hotspots, persevere where you are meeting needs, and stop wasting time on the areas that are not interesting to your stakeholders.

Lean Startup also provides practical advice that correlates to building versus deploying models. You need to expand your “small batch” test models that show promise with larger implementations on larger sets of data. You may need to pivot again as you apply more data in case your small batch was not truly representative of the larger environment. Remember that a model is a generalization of “what is” that you can use to predict “what will be.” If your “what is” is not true, your “what will be” may turn out to be wrong.

Another lesson from *Lean Startup* is that you should align your efforts to some bigger-picture vision of what you want to do. Innovations are built on innovations, and each of your smaller discoveries will have outputs that should contribute to the story you want to tell. Perhaps your router memory solution is just one of hundreds of such models that you build in your environment, all of which contribute to the “network health” indicator that you provide as a final solution to upper management.

Cognitive Trickery

Recall these questions from Chapter 5:

1. If a bat and ball cost \$1.10, and the bat costs \$1 more than the ball, how much does the ball cost?
2. In a lake, there is a patch of lily pads. Every day, the patch doubles in size. If it takes 48 days for the patch to cover the entire lake, how long does it take for the patch to cover half of the lake?
3. If it takes 5 machines 5 minutes to make 5 widgets, how long does it take 100 machines to make 100 widgets?

What happens when you read these questions now?

You have a different perspective on these questions than you had before you read them in Chapter 5. You have learned to stop, look, and think before providing an answer. Your System 2 should now engage in these questions and others like them. Even though you now know the answer, you still think about it. You mentally run through the math again to truly understand the question and its answer. You can create your own tricks that similarly cause you to stop and think.

Quick Innovation Wins

As you start to go down the analytics innovation path, you can find quick wins by programmatically applying what you already know to your environment as simple algorithms. When you turn your current expertise from your existing expert systems into algorithms, you can apply each one programmatically and then focus on the next thing. Share your algorithms with other systems in your company to improve them. Moving forward, these algorithms can underpin machine reasoning systems, and the outcomes of these algorithms can together determine the state of a systems to be used in higher-order models. Every bit of knowledge that you automate creates a new second-level data point for you.

Again consider the router memory example here. You could have a few possible scenarios for automating your knowledge into larger solutions:

- When router memory reaches 99% on this type of router, the router crashes. Implemented models in this space would be analyzing current memory conditions to determine whether and when 99% is predicted.
- When router memory reaches 99% on this other type of router, the router does not crash, but traffic is degraded, and some other value, such as traffic drops on interfaces, increases. Correlate memory utilization with high and increased drops in yet another model.

- If you are doing traffic path modeling, determine the associated traffic paths for certain applications in your environment, using models that generate traffic graphs based on the traffic parameters.
- Use all three of these models together to proactively get notification when applications are impacted by a current condition in the environment. Since your lower-level knowledge is now automated, you have time to build to this level.
- If you have the data from the business, determine the impact on customers of application performance degradation and proactively notify them. If you have full-service assurance, use automation to move customers to a better environment before they even notice the degradation.

Knowing what you have to work with for analytics is high value and provides statistics that you can roll up to management. You now have the foundational data for what you want to build. So, for quick wins that benefit you later, you can do the following:

- Build data pipelines to provide the data to a centralized location.
- Document the data pipelines so you can reuse the data or the process of getting the data.
- Identify missing data sources so you can build new pipelines or find suitable proxies.
- Visualize and dashboard the data so that others can take advantage of it.
- Use the data in your new models for higher-order analysis.
- Develop your own data types from your SME knowledge to enrich the existing data.
- Continuously write down new idea possibilities as you build these systems.
- Identify and make available spaces where you can work (for example, your laptop, servers, virtual machines, the cloud) so you can try, fail fast, and succeed.
- Find the outliers or TopN and BottomN to identify relevant places to start using outlier analysis.
- Start using some of the common analytics tools and packages to get familiar with them. Recall that you must be engaged in order to learn. No amount of just reading about it substitutes for hands-on experience.

Summary

Why have we gone through all the biases in Chapter 5 and innovation in this chapter? Understanding both biases and innovation gives you the tools you need to find use cases. Much as the Cognitive Reflection Test questions forced you to break out of a comfortable answer and think about what you were answering, the use cases in Chapter 7 provide an opportunity for you to do some examining with your innovation lenses. You will gain some new ideas.

You have also learned some useful techniques for creative and metaphoric thinking. In this chapter you have learned techniques that allow you to gain new perspectives and increase your breadth to develop solutions. You have learned questioning techniques that allow you to increase your knowledge and awareness even further. You now have an idea of where and how to get started for some quick wins. Chapter 7 goes through some industry use cases of analytics and the intuition behind them. Keep an open mind and take notes as ideas come to you so that you can later review them. If you already have your own ways of enhancing your creative thinking, now is the time to engage them as well. You only read something for the first time one time, and you may find some fresh ideas in the next chapter if you use all of your innovation tools as you get this first exposure.

This page intentionally left blank

Analytics Use Cases and the Intuition Behind Them

Are you ready to innovate? This chapter reviews use-case ideas from many different facets of industry, including networking and IT. The next few chapters expose you to use-case ideas and the algorithms that support the underlying solutions. Now that you understand that you can change your biases and perspectives by using creative thinking techniques, you can use the triggering ideas in this chapter to get creative.

This chapter will hopefully help you gain inspiration from existing solutions in order to create analytics use cases in your own area of expertise. You can use your own mental models combined with knowledge of how things have worked for others to come up with creative, provable hypotheses about what is happening in your world. When you add your understanding of the available networking data, you can arrive at new and complete analytics solutions that provide compelling use cases.

Does this method work? Pinterest.com has millions of daily visitors, and the entire premise behind the site is to share ideas and gain inspiration from the ideas of others. People use Pinterest for inspiration and then add their own flavor to what they have learned to build something new. You can do the same.

One of the first books I read when starting my analytics journey was *Taming the Big Data Tidal Wave* by Bill Franks. The book offers some interesting insights about how to build an analytics innovation center in an organization. Mr. Franks is now chief analytics officer for The International Institute for Analytics (IIA). In a blog post titled *The Post-Algorithmic Era Has Arrived*, Franks writes that in the past, the most valuable analytics professionals were successful based on their knowledge of tools and algorithms. Their primary role was to use their ability and mental models to identify which algorithms worked best for given situations or scenarios.

That is no longer the only way. Today, software and algorithms are freely available in open source software packages, and computing and storage are generally inexpensive. Building a big data infrastructure is not the end game—just an enabling factor. Franks states, “The post-algorithmic era will be defined by analytics professionals who focus on innovative

uses of algorithms to solve a wider range of problems as opposed to the historical focus on coding and manually testing algorithms.” Franks’s first book was about defining big data infrastructure and innovation centers, but then he pivoted to a new perspective. Franks moved to the thinking that analytics expertise is related to understanding the *gist* of the problem and identifying the right *types* of candidate algorithms that might solve the problem. Then you just run them through black-box automated testing machines, using your chosen algorithms, to see if they have produced desirable results. You can build or buy your own black-box testing environments for your ideas. Many of these black boxes perform deep learning, which can provide a shortcut from raw data to a final solution in the proper context.

I thoroughly agree with Franks’s assessment, and it is a big reason that I do not spend much time on the central engines of the analytics infrastructure model presented in Chapter 2, “Approaches for Analytics and Data Science.” The analytics infrastructure model is useful in defining the necessary components for operationalizing a fully baked analytics solution that includes big data infrastructure. However, many of the components that you need for the engine and algorithm application are now open source, commoditized, and readily available. As Franks calls out, you still need to perform the due diligence of setting up the data and the problem, and you need to apply algorithms that make technical sense for the problem you are trying to solve. You already understand your data and problems. You are now learning an increasing number of options for applying the algorithms.

Any analysis of how analytics is used in industry is not complete without the excellent perspective and research provided by Eric Siegel in his book *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die* (which provided a strong inspiration for using the simple bulleted style in this chapter). As much as I appreciated Franks’s book for helping get started with big data and analytics, I appreciated Siegel’s book for helping me compare my requirements to what other people are actually doing with analytics. Siegel helped me appreciate the value of seeing how others are creating use cases in industries that were previously unknown to me. Reading the use cases in his book provided new perspectives that I had not considered and inspired me to create use cases that Cisco Services uses in supporting customers.

Competing on Analytics: The New Science of Winning, by Thomas Davenport and Jeanne Harris, shaped my early opinion of what is required to build analytics solutions and use cases that provide competitive advantage for a company. In business, there is little value in creating solutions that do not create some kind of competitive advantage or tangible improvement for your company.

I also gained inspiration from Simon Sinek’s book *Start with Why: How Great Leaders Inspire Everyone to Take Action*. Why do you build models? Why do you use this data science stuff in your job? Why should you spend your time learning data science use cases and algorithms? The answer is simple: Analytics models produce insight, and you must tie that insight to some form of business value. If you can find that insight, you can improve the business. Here are some of the activities you will do:

- **Use machine learning and prepared data sets to build models of how things work in your world**—A model is a generalization of *what is*. You build models to represent the current state of something of interest. Your perspective from inside your own company uniquely qualifies you to build these models.
- **Use models to predict future states**—This involves moving from the descriptive analytics to predictive analytics. If you have inside knowledge of *what is*, then you have an inside track for predicting *what will be*.
- **Use models to infer factors that lead to specific outcomes**—You often examine model details (model interpretation) to determine what a model is telling you about how things actually manifest. Sometimes, such as with neural networks, this may not be easy or possible. In most cases, some level of interpretation is possible.
- **Use machine learning methods, such as unsupervised learning, to find interesting groupings**—Models are valuable for understanding your data from different perspectives. Understanding how things actually work *now* is crucial for predicting how they will work in the future.
- **Use machine learning with known states (sometimes called *supervised learning*) to find interesting groups that behave in certain ways**—If things remain status quo, you have uncovered the base rate, or the way things are. You can immediately use these models for generalized predictions. If something happened 95% of the time in the past, you may be able to assume that it has a 95% probability of happening in the future if conditions do not change.
- **Use all of these mechanisms to build input channels for models that require estimates of current and future states**—Advanced analytics solutions are usually several levels abstracted from raw data. The inputs to some models are outputs from previous models.
- **Use many models on the same problem**—Ensemble methods of modeling are very popular and useful as they provide different perspectives on solutions, much as you can choose better use cases by reviewing multiple perspectives.

Models do not need to be complex. Identifying good ways to meet needs at critical times is sometimes a big win and often happens with simple models. However, many systems are combinations of multiple models, ensembles, and analytics techniques that come together in a system of analysis.

Most of the analytics in the following sections are atomic use cases and ideas that produce useful insights in one way or another. Many of them are not business relevant alone but are components that can be used in larger campaigns. Truly groundbreaking business-relevant solutions are combinations of *many* atomic components. Domain experts, marketing specialists, and workflow experts assemble these components into a process that fits a particular need. For example, it may be possible to combine location analytics with buying patterns from particular clusters of customers for targeted advertising. In this same instance, supply chain predictive analytics and logistics can determine that you have what customers want, where they want it, when they want to buy it. Sold.

Analytics Definitions

Before diving into the use cases and ideas, some definitions are in order to align your perspectives:

Note These are *my* definitions so that you understand my perceptions and my bias as I write this book. You can find many other definitions on the Internet. Explore the use cases in this book according to any bias that you perceive I may have that differs from your own thinking. Expanding your perspective will help you maximize your effectiveness in getting new ideas.

- **Use case**—A use case is simply some challenge solved by combining data and data science in a way that solves a business or technical problem for you or your company. The data, the data engine, the algorithms, and the analytics solution are all parts of use cases.
- **Analytics solutions**—Sometimes I interchange the terms *analytics solutions* and *use cases*. In general, a use case solves a problem or produces a desired outcome. An analytics solution is the underlying pipeline from the analytics infrastructure model. This is the assembly of components required to achieve the use case. I differentiate these terms because I believe you can use many analytics solutions to solve different use cases, across different industries, by tweaking a few things and applying data from new domains.
- **Data mining**—Data mining is the process of collecting interesting data. The key word here is *interesting* because you may be looking for specific patterns or types of data. Once you build a model that works, you will use data mining to find all data that matches the input parameters that you chose to use for your models. Data mining differs from *machine learning* in that it means just gathering, creating, or producing data—not actively learning from it. Data mining often precedes machine learning in an analytics solution, however.
- **Hard data**—Hard data are values that are collected or mathematically derived from collected data. Simple counters are an example. Mean, median, mode, and standard deviations are derivations of hard data. Your hair color, height, and shoe size are all hard data.
- **Soft data**—Soft data may be values assigned by humans, it is typically subjective, and it may involve data values that differ from solution to solution. For example, the same network device can be of critical importance in one network, and another customer may use the same kind of device for a less critical function. Similarly, what constitutes a healthy component in a network may differ across organizations.
- **Machine learning**—Machine learning involves using computer power and instances of data to characterize how things work. You use machine learning to build models. You use *data mining* to gather data and *machine learning* to characterize it—in supervised or unsupervised ways.

- **Supervised machine learning**—Supervised machine learning involves using cases of past events to build a model to characterize how a set of inputs map to the output(s) of interest. *Supervised* indicates that some outcome variables are available and used. You call these outcome variables *labels*. Using the router memory example from earlier chapters, a simple labeled case might be that a specific router type with memory >99% will crash. In this case, Crash=Yes is the output variable, or label. Another labeled case might be a different type of router with memory >99% that did not crash. In this situation, Crash=No is the outcome variable, or label. Supervised learning should involve training, test, and validation, and you most commonly use it for building classification models.
- **Unsupervised machine learning**—Unsupervised machine learning generally involves clustering and segmentation. With unsupervised learning, you have the set of input parameters but *do not have a label* for each set of input parameters. You are just looking for interesting patterns in the input space. You generally have no output space and may or may not be looking for it. Using the router memory example again, you might gather all routers and cluster them into memory utilization buckets of 10%. Using your SME skills, you may recognize that routers in the memory cluster “memory >90%” crash more than others, and you can then build a supervised case from that data. Unsupervised learning does not require a train/test split of the data.

How to Use the Information from This Chapter

Before getting started on reviewing the use cases and ideas, the following sections provide a few words of advice to prime your thinking as you go forward.

Priming and Framing Effects

Recall the priming and framing effects, in which the data that you hear in a story takes your mind to a certain place. By reading through the cases here, you will prime your brain in a different direction for each use case. Then you can try to apply this case in a situation where you want to gain more insights. This familiarity can help you frame up your own problems. The goal here is to keep an open mind but also to go down some availability cascades, follow the illusion-of-truth *what-if* paths, and think about the general idea behind the solution. Then you can determine if the idea or style of the current solution fits something that you want to try. Every attempt you try is an instance of deliberate practice. This will make you better at finding use cases in the long term.

Analytics Rube Goldberg Machines

As you open your mind to solutions, make sure that the solutions are useful and relevant to your world. Recall that with a Rube Goldberg machine, you use an excessive amount of activity to accomplish a very simple task, such as turning on a light. If you don't plan your analytics well, you could end up with a very complex and expensive solution that delivers nothing more than some simple rollups of data. Management would not want you

to spend years of time, money, and resources on a data warehouse, only to end up with just a big file share. You can use the data mined to build use cases and increase the value immediately. Just acquiring, rolling up, and storing data may or may not be an enabler for the future. If the benefit is not there, pivot your attention somewhere else. Find ideas in this chapter that are game changers for you and your company. Alternatively, avoid spending excessive time on things that do not move the needle unless you envision them as necessary components of larger systems or your own learning process.

You will hear of the “law of parsimony” in analytics; it basically says that the simplest explanation is usually the best one. Sometimes there are very simple answers to problems, and fancy analytics and algorithms are not needed.

Popular Analytics Use Cases

The purpose of this section is not to get into the details of the underlying analytics solutions. Instead, the goal is to provide you with a broad array of use-case possibilities that you can build.

Keep an open mind, and if any possibility of mapping these use cases pops into your head, write it down before you forget it or replace it with other ideas as you continue to read. When you write something down, be sure to include some reasons you think it might work for your scenario. Think about any associations to your mental models and bias from Chapter 5, “Mental Models and Cognitive Bias,” to explore each interesting use case in your mind. Use the innovation techniques discussed in Chapter 6, “Innovative Thinking Techniques,” to fully explore your idea in writing. As an analytics innovator, it is your job to look at these use cases and determine how to retrofit them to your problems. If you need to stop reading and put some thought into a use case, please do so. Stopping and writing may invoke your System 2. The purpose of this chapter is to generate useful ideas. Write down where you are, change your perspective, write that down, and compare the two (or more) later. In Chapter 8, “Analytics Algorithms and the Intuition Behind Them,” you’ll explore candidate algorithms and techniques that can help you to assemble the use case from ideas you gain here.

There are three general themes of use cases in this section:

- Machine learning and statistics use cases
- Common IT analytics use cases
- Broadly applicable use cases

Under each of these themes are detailed lists of ideas related to various categories. My bias as a network SME weights some areas heavier in networking because that is what I know. Use those as easy mappings to your own networking use cases. I have tried to find relevant use cases from surrounding industries as well, but I cannot list them all as analytics is pervasive across all industries. Some sections are filled with industry use cases, some are filled with simple ideas, and others are from successful solutions used every day by Cisco Services.

There are many overlapping uses of analytics in this chapter. Many use cases do not fall squarely into one category, but some categorization is necessary to allow you to come back to a specific area later when you determine you want to build a solution in that space. I suggest that you read multiple sections and encourage you to do Internet searches to find the latest research and ideas on the topic. Analytics use cases and algorithms are evolving daily, and you should always review the state of the art as you plan and build your own use cases.

Machine Learning and Statistics Use Cases

This section provides a collection of machine learning technologies and techniques, as well as details about many ways to use these techniques. Many of these are atomic uses, which become part of larger overall systems. For example, you might use some method to cluster some things in your environment and then classify that cluster as a specific type of importance, determine some work to do to that cluster, and visualize your findings all as part of an “activity prioritization” or “recommender” system. You will use the classic machine learning techniques from this section over and over again.

Anomalies and Outliers

Anomaly detection is also called *outlier*, or *novelty, detection*. When something is outside the range of normal or expected values, it is called an *anomaly*. Sometimes anomalies are expected in random processes, but other times they are indicators that something is happening that shouldn't be happening in the normal course of operations. Whether the issue is about your security, your location, your behavior, your activities, or data from your networks, there are anomaly detection use cases.

The following are some examples of anomaly detection use cases:

- You can use machine learning to classify, cluster, or segment populations that may have different inherent behaviors to determine what is anomalous. This can be time series anomalies or contextual anomalies, where the definition of *anomaly* changes with time or circumstance.
- You can easily show anomalies in data that you visualize as points far from cluster centers or far from any other clusters.
- Collective anomalies are groups of data observations that together form an anomaly, such as a transaction that does not fit a definition of a normal transaction.
- For supervised learning anomaly detection, there are a few options. Sometimes you are not as interested in learning from the data sets as you are in learning about the misclassification cases of your models. If you built a good supervised model on known good data only, the misclassifications are anomalies because there is something that makes your “known good” model misclassify them. This method, sometimes called semi-supervised learning, is a common whitelisting method.

- In an alternative case, both known good and known bad cases may be used to train the supervised models, and you might use traditional classification to predict the most probable classification. You might do this, for example, where you have historical data such as fraud versus no fraud, spam versus non-spam, or intrusion versus no intrusion.
- You can often identify numeric anomalies by using statistical methods to learn the normal ranges of values. *Point anomalies* are data points that are significantly different from points gathered in the same context.
- If you are calling out anomalies based on known thresholds, then you are using expert systems and doing matching. These are still anomalies, but you don't need to use data science algorithms. You may have first found an anomaly in your algorithmic models and then programmed it into your expert systems for matching.
- Anomaly detection with Internet of Things (IoT) sensor data is one of the easiest use cases of machine data produced by sensors. Statistical anomaly detection is a good start here.
- Some major categories of anomaly detection include simple numeric and categorical outliers, anomalous patterns of transactions or behaviors, and anomalous rate of change over time.

Many find outlier analysis to be one of the most intuitive areas to start in analytics. With outlier analysis, you can go back to your investigative and troubleshooting roots in networking to find *why* something is different from other things. In business, outliers may be new customer segments, new markets, or new opportunities, and you might want to understand more about why something is an outlier. The following are some examples of outlier analysis use cases:

- Outliers by definition are anomalies. Your challenge is determining if they are interesting enough to dig into. Some processes may be inherently messy and might always have a wide range of outputs.
- Recall the Sesame Street analytics from Chapter 1, “Getting Started with Analytics.” Outlier analysis involves digging into the details about why something is not like the others. If you need to show it, build the Sesame Street visualizations.
- Is this truly an outlier, supported by analysis and data? Recall the law of small numbers and make sure that you have feel for the base rate or normal range of data that you are looking at.
- Are you viewing an outlier or something from a different population? A single cat in the middle of a group of dogs would appear to be an outlier if you are only looking at dog traits.
- Perhaps 99% memory utilization on a router is rare and an outlier. Perhaps some other network device maximizes performance by *always* consuming as much memory as possible.

- If you are seeing a rare instance, what makes it rare? Use five whys analysis. Maybe there is a good reason for this outlier, and it is not as interesting as it originally seemed.
- In networking, traffic microbursts and utilization hotspots will show as outliers with the wrong models, and you may need to change the underlying models to time series.
- In failure analysis, both short-lived and long-lived outliers are of interest. Seek to understand the reasons behind both.
- Sometimes outliers are desirable. If your business has customers, and you model the profit of all your customers using a bell curve distribution, which ones are on the high end? Why are they there? What are they finding to be high value that others are not?
- Outliers may indicate the start of a new trend. If you had modeled how people consumed movies in the 1980s and 1990s, watching movies online may have seemed like an outlier. Maybe you can find outliers that allow you to start the next big trend.
- You can examine outliers in healthcare to see why some people live longer or shorter lives. Why are most people susceptible to some condition but some are not? Why do some network devices work well for a purpose but some do not?
- Retail and food industries use outlier analysis to look at locations that do well compared to locations that do not. Identifying the profile of a successful location helps identify the best growth opportunities in the future.

This chapter could list many more use cases of outliers and anomalies. Look around you right now and find something that seems out of place to you. Keep in mind that outliers may be objective and based on statistical measures, or they may be subjective and based on experiences. Regardless of the definition that you use, identifying and investigating differences from the *common* in your environment helps you learn data mining and will surely result in finding some actionable areas of improvement.

Anomaly detection and outlier analysis algorithms are numerous, and application depends on your needs.

Benchmarking

Benchmarking involves comparison against some metric, which you derive as a preferred goal or base upon some known standard. A benchmark may be a subjective and company-specific metric you desire to attain. Benchmarks may be industrywide. Given a single benchmark or benchmark requirement, you can innovate in many areas. The following are examples of benchmarking use cases:

- The first and most obvious use is comparison, with the addition of a soft value of *compliance to benchmark* for your analysis. Exceeding a benchmark may be good or bad, or it may be not important. Adding the soft value helps you identify the criticality of benchmarks.

- Rank items based on their comparison to a benchmark. Perhaps your car does really well in the 0–60 benchmark category, and your *drive to work* overlay on the world moves at a much faster pace than others' *drive to work* overlays. In this case, there are commuters who rank above and below you.
- Use application benchmarking to set a normal response time that provides a metric to determine whether an application is performing well or is degraded.
- Benchmark application performance based on group-based asset tracking. Use the information you gather to identify network hotspots. What you have learned about anomaly detection can help here.
- Use performance benchmarking to compare throughput and bandwidth in network devices. Correlate with the application benchmarks discussed and determine if network bandwidth is causing application degradation.
- Define your networking data KPIs relative to industry or vertical benchmarks that you strive to reach. For example, you may calculate uptime in your environment and strive to reach some number of nines following 99% (for example, 99.99912% uptime, or “three nines”).
- Establish dynamic statistical benchmarks by calculating common and normal values for a given data point and then comparing everyone to the expected value. This value is often the mean or median in the absence of an industry-standard benchmark. This means using the wisdom of the crowd or normal distribution to establish benchmarks.
- Published performance and capacity numbers from any of your vendors are numbers that you can use as benchmarks. Alternatively, you can set benchmarks at some lower number, such as 80% of advertised capacity. When your Internet connection is constantly averaging over 80%, is this affecting the ability to do business? Is it time to upgrade the speed?
- Performance benchmarks can be subjective. Use configuration, device type, and other data points found in clustering and correlation analysis to identify devices that are performing suboptimally.
- Combine correlated benchmark activity. For example, a low data plane performance benchmark correlated with a high control plane benchmark may indicate that there is some type of churn in the environment.
- For any numerical value that you collect or derive, there is a preferred benchmark. You just need to find it and determine the importance.
- Measure compliance in your environment with benchmarking and clustering. If you have components that are compliant, benchmark other similar components using clustering algorithms.
- Examine consistency of configurations through clustering. Identify which benchmark to check by using classification algorithms.

- Depending on the metrics, historical behavior and trend analysis are useful for determining when values trend toward noncompliance.
- National unemployment rates provide a benchmark for unemployment in cities when evaluating them for livability.
- Magazine rankings of best places to live benchmark cities and small towns. You may use these to judge how much your own place to live has to offer.
- Magazine and newspaper rankings of best employers have been setting the benchmarks for job perks and company culture for years.
- Compliance and consistency to some set of standards is common in networking. This may be Health Insurance Portability and Accountability Act (HIPAA) compliance for healthcare or Payment Card Industry (PCI) compliance for banks. The basic theory is the same: You can define compliance loosely as a set of metrics that must meet or exceed a set of thresholds.
- If you know your benchmarks, you can often just establish the metrics (which may also be KPIs) and provide reporting.

How you arrive at the numbers for benchmarking is up to you. This is where your expertise, your bias, your understanding of your company biases, and your creativity are important. Make up your own benchmarks relative to your company needs. If they support the vision, mission, or strategy of the company, then they are good benchmarks that can drive positive behaviors.

Classification

The idea behind classification is to use a model to examine a group of inputs and provide a best guess of a related output. Classification is a typical use case of supervised machine learning, where an algorithm or analytics model separates or segments the data instances into groups, based on a previously trained classification model. Can you classify a cat versus a dog? A baseball versus a football? You train a classifier to process inputs, and then you can classify new instances when you see them. You will use classification a lot. Some key points:

- Classification is a foundational component of analytics and underpins many other types of analysis. Proper classification makes your models work well. Improper classification does the opposite.
- If you have observations with labeled inputs, use machine learning to develop a classification model that classifies previously unseen instances to some known class from your model training. There are *many* algorithms available for this common purpose.
- Use selected groups of hard and soft data from your environment to build input maps of your assets and assign known labels to these inputs. Then use the maps to train a model that identifies classes of previously unknown components as they come online. The choice of labels is entirely subjective.

- Once items are classified, apply appropriate policies based on your model output, such as policies for intent-based networking.
- Cisco Services uses many different classifier methods to assess the risk of customer devices hitting some known event, such as a bug that can cause a network device crash.
- If you are trying to predict the 99% memory impact in a router (as in the earlier example), you need to identify and collect instances of the many types of routers that ran at 99% to train a model, and then you can use that model to classify whether your type of router would crash into “yes” and “no” classes.

Some interesting classification use cases in industry include the following:

- Classification of potential customers or users into levels of desirability for the business. Customers that are more desirable would then get more attention, discounts, ads, or special promotions.
- Insurance companies use classification to determine rates for customers based on risk parameters.
- Use simple classifications of desirability by developing and evaluating a model of pros and cons used as input features.
- Machines can classify images from photos and videos based on pixel patterns as cats and dogs, numbers, letters, or any other object. This is a key input system to AI solutions that interact with the world around them.
- The medical industry uses historical cases of biological markers and known diseases for classification and prediction of possible conditions.
- Potential epidemics and disease growth is classified and shared in healthcare, providing physicians with current statistics that aid in diagnosis of each individual person.
- Retail stores use loyalty cards and point systems to classify customers according to their *loyalty*, or the amount of business they conduct. A store that classifies someone as a top customer—like a casino whale—can offer that person preferred services.

Classification is widely discussed in the analytics literature and also covered in Chapter 8. Spend some time examining multiple classification methods in your model building because doing so builds your analytics skills in a very heavily used area of analytics.

Clustering

Classification involves using labeled cases and supervised learning. Clustering is a form of *unsupervised* learning, where you use machine learning techniques to cluster together groups of items that share common attributes. You don’t have *labels* for unsupervised clustering. The determination of how things get clustered depends on the clustering algorithms, data engineering, feature engineering, and distance metrics used. Popular

clustering algorithms are available for both numeric and categorical features. Common clustering use cases include the following:

- Use clustering as a method of data reduction. In data science terms, the “curse of dimensionality” is a growing issue with the increasing availability of data. *Curse of dimensionality* means that there are just too many predictors with too many values to make reasonable sense of the data. The obvious remedy to this situation is to reduce the number of predictors by removing ones that do not add a lot of value. Do this by clustering the predictors and using the cluster representation in place of the individual values in your models.
- Aggregate or group transactions. For example, if you rename 10 events in the environment as a single incident or new event, you have quickly reduced the amount of data that you need to analyze.
- A simple link that goes down on a network device may produce a link down message from both sides of that link. This may also produce protocol down messages from both sides of that link. If configured to do so, the upper-layer protocol reconvergence around that failed link may also produce events. This is all one cluster.
- Clustering is valuable when looking at cause-and-effect relationships as you can correlate the timing of clustered events with the timing of other clustered events.
- In the case of IT analytics, clusters of similar devices are used in conjunction with anomaly detection to determine behavior and configuration that is outside the norm.
- You can use clustering as a basis for a recommender system, to identify clusters of purchasers and clusters of items that they may purchase. Clustering groups of users, items, and transactions is very common.
- Clustering of users and behaviors is common in *many* industries to determine which users perform certain actions in order to detect anomalies.
- Genome and genetics research groups cluster individuals and geographies predisposed to some condition to determine the factors related to that condition.
- In supervised learning cases, once you classify items, you generally move to clustering them and assign a persona, such as a user persona, to the entire cluster.
- Use clustering to see if your classification models are providing the classifications that you want and expect.
- Further cluster within clusters by using a different set of clustering criteria to develop subclusters. Further cluster servers into Windows and Linux. Further cluster users into power users and new users.
- Associate user personas with groups of user preferences to build a simple recommender system. Maybe your power users prefer Linux and your sales teams prefer Windows.

- Associate groups of devices to groups of attributes that those devices should have. Then build an optimization system for your environment similar to recommender systems used by Amazon and Netflix.
- The IoT takes persona creation to a completely new level. The level of detail available today has made it possible to create very granular clusters that fit a very granular profile for targeted marketing scenarios.
- Choose feature-engineering techniques and added soft data to influence how you want to cluster your observations of interest.
- Use reputation scoring for clustering. Algorithms are used to roll up individual features or groups of features. Clusters of items that score the same (for example, “consumers with great credit” or “network devices with great reliability”) are classified the same for higher-level analysis.
- *Customer segmentation* involves dividing a large group of potential customers into groups. You can identify these groups by characteristics that are meaningful for your product or service.
- A business may identify a target customer segment that it wants to acquire by using clustering and classification. Related to this, the business probably has a few customer segments that it doesn’t want (such as new drivers for a car insurance business).
- Insurance companies use segmentation via clustering to show a worse price for customers that they want to push to their competitors. They can choose to accept such customers who are willing to pay a higher price that covers the increased risk of taking them on, according to the models.
- A cluster of customers or people is often called a *cohort*, and a cohort can be a given label such as “highly active” or “high value.”
- Banks and other financial institutions cluster customers into segments based on financials, behavior, sentiment, and other factors.

Like classification, clustering is widely covered in the literature and in Chapter 8. You can find use cases across all industries, using many different types of clustering algorithms. As an SME in your space, seek to match your available data points to the type of algorithm that best results in clusters that are meaningful and useful to you. Visualization of clustering is very common and useful, and your algorithms and dimensionality reduction techniques need to create something that shows the clusters in a human-consumable format. Like classification, clustering is a key pillar that you should seek to learn more about as you become more proficient with data science and analytics.

Correlation

Correlation is simply *co-relation*, or the appearance of a mutual relationship. Recall from Chapter 6 that eating ice cream does not cause you to drown, but occurrences of these two activities rise and fall together. For any cases of correlation, you must have time awareness in order for all sources to have valid correlation. Correlating data from January

through March with data from July through September does not make sense unless you expect something done in January through March to have a return on investment in two quarters.

Correlation is very intuitive to your pattern-seeking brain, so the use cases may not always be causal in nature, but even then, you may find them interesting. Note that using correlations is often a higher level over using individual data points. Correlations are generally not individual values but instead trends of those individual values. When two values move in the same direction over the same period of time, these numerical values are indeed correlated. That is simple math. Whether there is causation in either of these values toward the other must be investigated.

Correlation can be positive or negative. For example the number of outdoor ice skating injuries would decrease as ice cream eating increases. Both positive and negative correlation can be quantified and used to develop solutions.

Correlation is especially useful in IT networking. Because IT environments are very complex, correlation between multiple sources is a powerful tool to determine cause and effect of problems in the environment. Coupling this with anomaly detection as well as awareness of the changes in the environment further adds quality to the determination of cause and effect. The following are examples of correlation use cases:

- Most IT departments use some form of correlation across the abstraction layers of infrastructure for troubleshooting and diagnostic analytics. Recall that you may have a cloud application on cloud infrastructure on servers in your data center. You need to correlate a lot of layers when troubleshooting.
- Visual values may be arranged in stack charts over time or in a *swim lanes* configuration to allow humans to see correlated patterns.
- Event correlation from different environments within the right time window shows cause-and-effect relationships.
- A burst in event log production from components in an area of the IT environment can be expected if it is correlated with a schedule change event in that environment.
- A burst can be identified as problematic if there was no expected change in this environment.
- Correlation is valuable in looking at the data plane and control plane in terms of maximizing the performance in the environment. Changes in data plane traffic flow patterns are often correlated with control plane activity.
- As is done in Information Technology Infrastructure Library (ITIL) practices, you can group events, incidents, problems, or other sets of data and correlate groups to groups. Perhaps you can correlate an entire group “high web traffic” with “ongoing marketing campaign.”
- Groups could be transactions (ordered groups). You could correlate transactions with other transactions, other clusters or groups, or events.

- Groups map to other purposes, such as a group of IT plus IoT data that allows you to know where a person is standing at a given time. Correlate that with other groups and other events at the same location, and you will know with some probability what they are doing there.
- Correlate time spent to work activities in an environment. Which activities can you shorten to save time?
- Correlate incidents to compliance percentages. Do more incidents happen on non-compliant components? Does a higher percentage of noncompliance correlate with more incidents?
- You can correlate application results with application traffic load or session opens with session activity. Inverse correlations could be DoS/DDoS attacks crippling the application.
- Wearable health devices and mobile phone applications enable correlation of location, activities, heart rate, workout schedules, weather, and much more.
- If you are tracking your resource intake in the form of calories, you can correlate weight and health numbers such as cholesterol to the physical activity levels.
- Look at configurations or functions performed in the environment and correlate devices that perform those functions well versus devices or components that do not perform them well. This provides insight into the best platform for the best purpose in the IT environment.

For any value that you track something over time, you can correlate with something else over time. Just be sure to do the following:

- Standardize the scales across the two numbers. A number that scales from 1 to 10 with a number that scales from 1 to 1 million is going to make the 1 to 10 scale look like a flat line, and the visual correlation will not be obvious.
- Standardize the timeframes based on the windows of analysis desired.
- You may need to transform the data in some way to find correlations, such as applying log functions or adjusting for other known factors.
- When correlations are done on non-linear data, you may have to make your data appear to be linear through some transformation of the values.

There are many instances of interesting correlations in the literature. Some are completely unrelated yet very interesting. For your own environment, you need to find correlations that have causations that you can do something about. There are algorithms and methods for measuring the degree of correlation. Correlation in predictors used in analytics models sometimes lowers the effectiveness of the models, and you will often evaluate correlation when building analytics models.

Data Visualization

Data visualization is a no-brainer in analytics. Placing data into a graph or a pie or bubble chart allows for easy human examination of that data. Industry experts such as Stephen Few, Edward Tufte, and Nathan Yau have published impressive literature in this area. Many packages, such as Tableau, are available for data visualization by non-experts in the domain. You can use web libraries such as JavaScript D3 to create graphics that your stakeholders can use to interact with the data. They can put on their innovator hats and take many different perspectives in a very short amount of time.

Here are some popular visualizations, categorized by the type of presentation layer that you would use:

Note Many of these visualizations have multiple purposes in industry, so search for them online to find images of interesting and creative uses of each type. There are many variations, options, and names for similar visualizations that may not be listed here.

- Single-value visualization
 - A big number presented as a single value
 - Ordered list of single values and labels
 - Gauge that shows a range of possible values
 - Bullet graph to show boundaries to the value
 - Color on a scale to show meaning (green, yellow, red)
 - Line graph or trend line with a time component
 - Box plot to examine statistical measures
 - Histogram
- Comparing two dimensions
 - Bar chart (horizontal) and column chart (vertical)
 - Scatterplot or simple bubble chart
 - Line chart with both values on the same normalized scale
 - Area chart
 - Choropleth or cartogram for geolocation data
 - 2×2 box Cartesian
- Comparing three or more dimensions
 - Bubble chart with size or color component
 - Proportional symbol maps, where a bubble does not have to be a bubble image

- Pie chart
- Radar chart
- Overlay of dots or bubbles on images or maps
- Timeline or time series line or area map
- Venn diagram
- Area chart
- Comparing more than three dimensions
 - Many lines on a line graph
 - Slices on a pie chart
 - Parallel coordinates graph
 - Radar chart
 - Bubble chart with size and color
 - Heat map
 - Map with proportional dots or bubbles
 - Contour map
 - Sankey diagram
 - Venn diagram
- Visualizing transactions
 - Flowchart
 - Sankey diagram
 - Parallel coordinates graph
 - Infographic
 - Layer chart

Note The University of St. Gallen in Switzerland provides one of my favorite sites for reviewing possible visualizations: http://www.visual-literacy.org/periodic_table/periodic_table.html.

Data visualization using interactive graphics is very important for building engaging applications and workflows to highlight use cases. This small section barely scratches the surface of the possibilities for data visualization. As you develop your own ideas for use cases, spend some time looking at image searches of the visualizations you might use.

The right visualization can enhance the power of a very small insight many times over. You will enjoy liberal use of visualization for your own personal use as you explore data and build solutions.

When it comes time to create visualizations that you will share with others, ensure that those visualizations do not require your expert knowledge of the data for others to understand what you are showing. Remember that many people seeing your visualization will not have the background and context that you have, and you need to provide it for them. The insights you want to show could actually be masked by confusing and complex visualizations.

Natural Language Processing

Natural language processing (NLP) is really about *understanding* and *deriving meaning* from language, semantics included. You use NLP to assist computers in understanding human linguistics. You can use NLP to gain the essence of text for your own purposes. While much NLP is for figuring out semantic meanings, the methods used along the way are extremely valuable for you. Use NLP for cleaning text, ordering text, removing low-value words, and developing document (or any blob of text) representations that you can use in your analytics models.

Common NLP use cases include the following:

- Cisco Services often uses NLP for cleaning question-and-answer text to generate FAQs.
- NLP is used for generating feature data sets from descriptive text to be used as categorical features in algorithms.
- NLP is used to extract sentiment from text, such as Twitter feed analysis about a company or its products.
- NLP enables you to remove noisy text such as common words that add no value to an analysis.
- NLP is not just for text. NLP is language processing, and it is therefore a foundational component for AI systems that need to understand the meaning of human-provided instructions. Interim systems commonly convert speech to text and then extract the meaning from the text. Deep learning systems seek to eliminate the interim steps.
- Automated grading of school and industry certification tests involves using NLP techniques to parse and understand answers provided by test takers.
- Topic modeling is used in a variety of industries to find common sets of topics across unstructured text data.
- Humans use different terms to say the same thing or may simply write things in different ways. Use NLP techniques to clean and deduplicate records.

- Latent semantic analysis on documents and text is common in many industries. Use latent semantic analysis to find latent meanings or themes that associate documents.
- Sentiment analysis with social media feeds, forum feeds, or Q&A can be performed by using NLP techniques to identify the subjects and the words and phrases that represent feelings.
- Topic modeling is useful in industry where clusters of similar words provide insight into the theme of the input text (actual themes, not latent ones, as with latent semantic analysis). Topic modeling techniques extract the essence of comments, questions, and feedback in social media environments.
- Cisco Services used topic modeling to improve training presentations by using the topics of presentation questions from early classes to improve the materials for later classes.
- Much as with market basket, clustering, and grouping analysis, you can extract common topic themes from within or across clusters in order to identify the clusters. You apply topic models on network data to identify the device purpose based on the configured items.
- Topic models provide context to analysis in many industries. They do not need to be part of the predictive path and are sometimes offshoots. If you simply want to cluster routers and switches by type, you can do that. Topic modeling then tells you the purpose of the router or switch.
- Use NLP to generate simple word counts for word clouds.
- NLP can be used on log messages to examine the counts of words over time period N . If you have usable standard deviations, then do some anomaly detection to determine when there are out-of-profile conditions.
- N -grams may be valuable to you. N -grams are groups of words in order, such as bigrams and trigrams.
- Use NLP with web scraping or API data acquisition to extract meaning from unstructured text.
- Most companies use NLP to examine user feedback from all sources. You can, for example, use NLP to examine your trouble tickets.
- The semantic parts of NLP are used for sentiment analysis. The semantic understanding is required in order to recognize sarcasm and similar expressions that may be misunderstood without context.

NLP has many useful facets. As you develop use cases, consider using NLP for full solutions or for simple feature engineering to generate variables for other types of models. For any categorical variable space represented by text, NLP has something to offer.

Statistics and Descriptive Analytics

Statistics and analytics are not distinguished much in this book. In my experience, there is much more precision and rigor in statistical fields, and *close enough* often works well in analytics. This precision and rigor is where statistics can be high value. Recall that descriptive analytics involves a state of *what is* in the environment, and you can use statistics to precisely describe an environment. Rather than sharing a large number of industry- or IT-based statistics use cases, this section focuses on the general knowledge that you can obtain from statistics. Here are some areas where statistics is high value for descriptive analytics solutions:

- Descriptive analytics data can be cleaned, transformed, ranked, sorted, or otherwise munged and be ready for use in next-level analytics models.
- Central tendencies such as the mean, median, mode, or standard deviation provide representative inputs to many different analytics algorithms.
- Using standard deviation is an easy way to define an outlier. In a normal distribution (Gaussian), outliers can be two or three standard deviations from the mean. Extremity analysis involves looking at the top side and bottom side outliers.
- Minimum values, maximum values, quartiles, and percentiles are the basis for many descriptive analytics visualizations to be used instantly to provide context for users.
- Variance is a measure of the spread of data values. You can square the variance to get standard deviations, and you already know that you can use standard deviation for outlier detection.
- You can use *population* variance to calculate the variance of the entire population or *sample* variance to generate an estimate of the population variance.
- Covariance is a measure of how much two variables vary together. You can use correlation techniques instead of covariance by standardizing the covariance units.
- Probability theory from statistics underlies many analytics algorithms. Predictive analytics involves highly probable events based on a set of input variables.
- Sums-of-squares distance measures are foundational to linear approximation methods such as linear regression.
- Panel data (longitudinal) analysis is heavily rooted in statistics. Methods from this space are valuable when you want to examine subjects over time with statistical precision.
- Be sure that your asset-tracking solutions show counts and existence of all your data, such as devices, hardware, software, configurations, policies, and more. Try to be as detailed as an electronic health record so you have data available for any analytics you want to try in the future.
- Top-N and bottom-N reporting is highly valuable to stakeholders. Such reporting can often bring you ideas for use cases.

- For any numerical values, understand the base statistics, such as mean, median, mode, range, quartiles, and percentiles in general.
- Provide comparison statistics in visual formats, such as bar charts, pie charts, or line charts. Depending on your audience, simple lists may suffice.
- If you collect the values over time, correlate changes in various parts of your data and investigate the correlations for causations.
- Present gauge- and counter-based performance statistics over time and apply everything in this section. (Gauges are statistics describing the current time period, and counters are growing aggregates that include past time periods.)
- Create your own KPIs based on existing data or targets that you wish to achieve that have some statistical basis.
- Gain understanding of the common and base rates from things in your environment and build solutions that capture deviations from those rates by using anomaly-detection techniques.
- Document and understand the overall population that is your environment and provide comparison to any stakeholder that only knows his or her own small part of that population. Is that stakeholder the best or the worst?
- Statistics from activity systems, such as ticketing systems, provide interesting data to correlate with what you see in your device statistics. Growing trouble tickets correlated with shrinking inventory of a component is a reverse correlation that shows people are removing it because it is problematic.
- Go a step further and look for correlations of activity from your business value reporting systems to determine if there are factors in the inventory that are influencing the business either positively or negatively.

While there is a lot of focus on analytics algorithms in the literature, don't forget the power of statistics in finding insight. Many analytics algorithms are extensions of foundational statistics. Many others are not. IT has a vast array of data, and the statistics area is rich for finding areas for improvement. Cisco Services uses statistics in conjunction with automation, machine learning, and analytics in all the tools it has recently built for customer-facing consultants.

Time Series Analysis

Many use cases have some component of hourly, daily, weekly, monthly, quarterly, or yearly trends in the data. There may also be long-term trends over an entire set of data. These are all special use cases that require time series-aware algorithms. The following are some common time series use cases:

- Call detail records from help desk and call center activity monitoring and forecasting systems are often analyzed using time series methods.

- Inventory management can be used with supply chain analytics to ensure that inventory of required resources is available when needed.
- Financial market analysis solutions range far and wide, from people trying to buy stock to people trying to predict overall market performance.
- Internet clickstream analysis uses time series analysis to account for seasonal and marketing activity when analyzing usage patterns.
- Budget analysis can be done to ensure that budgets match the business needs in the face of changing requirements for time, such as stocking extra inventory for a holiday season.
- Hotels, conference centers, and other venues use time series analysis to determine the busy hours and the unoccupied times.
- Sales and marketing forecasts must take weekly, yearly, and seasonal trends into account.
- Fraud, intrusion, and anomaly detection systems need time series awareness to understand the normal behavior in the analysis time period.
- IoT sensor data could have a time series component, depending on the role of the IoT component. Warehouse activity is increased when the warehouse is actively operating.
- Global transportation solutions use time series analysis to avoid busy hours that can add time to transportation routes.
- Sentiments and behaviors in social networks can change very rapidly. Modeling the behavior for future prediction or classification requires time-based understanding coupled with context awareness.
- Workload projections and forecasts use time and seasonal components. For example, Cyber Monday holiday sales in the United States show a heavy increase in activity for online retailers.
- System activity logs in IT often change based on the activity levels, which often have a time series component.
- Telemetry data from networks or IoT environments often provides snapshots of the same values at many different time intervals.

If you have a requirement to forecast or predict trends based on hour, day, quarter, or periodic events that change the normal course of operation, you need to use time series methods. Recognize the time series algorithm requirement if you can graph your data and it shows as an oscillating, cyclical view that may or may not trend up or down in amplitude over time. (Some examples of these graphs are shown in Chapter 8.)

Voice, Video, and Image Recognition

Voice, video, and image recognition are hot topics in analytics today. These are based on variants of complex neural networks and are quickly evolving and improving. For your purposes, view these as simple inputs just like any numbers and text. There are lots of algorithms and analytics involved in dissecting, modeling, and classifying in image, voice, and video analytics, but the outcomes are a classified or predicted class or value. Until you have some skills under your belt, if you need voice, video, or image recognition, look to purchase a package or system, or use cloud resources that provide the output you need to use in your models. Building your own consumes a lot of time.

Common IT Analytics Use Cases

Hopefully now that you have read about the classic machine learning use cases, you have some ideas brewing about things you could build. This section shifts the focus to assembling atomic components of those classic machine learning use cases into broader solutions that are applicable in most IT environments. Solutions in this section may contain components from many categories discussed in the previous section.

Activity Prioritization

Activity prioritization is a guiding principle for Cisco Services, and in this section I use many Cisco Services examples. Services engineers have a lot of available data and opportunities to help customers. Almost every analytics use case developed for customers in optimization-based services is guided by two simple questions:

- Does this activity optimize how to spend time (opex)?
- Does this activity optimize how to spend money (capex)?

Cisco views customer recommendations that are made for networks through these two lenses. The most common use case of effective time spend is in condition-based maintenance, or *predictive maintenance*, covered later in this chapter.

Condition-based maintenance involves collecting and analyzing data from assets in order to know the current conditions. Once these current conditions are known and a device is deemed worthy of time spend based on age, place in network, purpose, or function, the following are possible and are quite common:

- Model components may use a data-based representation of everything you know about your network elements, including software, hardware, features, and performance.
- Start with descriptive analytics and top-N reporting. What is your worst? What is your best? Do you have outliers? Are any of these values critical?
- Perform extreme-value analysis by comparing best to worst, top values to bottom values. What is different? What can you infer? Why are the values high or low?

- As with the memory case, build predictive models to predict whether these factors will trend toward a critical threshold either high or low.
- Build predictive models to identify when these factors will reach critical thresholds.
- Deploy these models with a schedule that identifies timelines for maintenance activities that allow for time-saving repairs (scheduled versus emergency/outage, reactive versus proactive).
- Combine some maintenance activities in critical areas. Why touch the environment more than once? Why go through the initial change control process more than once?

Where to spend the money is the second critical question, and it is a natural follow-on to the first part of this process. Assuming that a periodic cost is associated with an asset, when does it become cost-prohibitive or unrealistic to maintain that asset? The following factors are considered in the analysis:

- Use collected and derived data, including support costs and the value of the component, to provide a cost metric. Now you have one number for a value equation.
- A soft value in this calculation could be the importance of this asset to the business, the impact of maintenance or change in the area where the asset is functioning, or the criticality of this area to the business.
- A second hard or soft value may be the current performance and health rating correlated with the business impact. Will increasing performance improve business? Is this a bottleneck?
- Another soft value is the cost and ease of doing work. In maintaining or replacing some assets, you may affect business. You must evaluate whether it is worth “taking the hit” to replace the asset with something more reliable or performant or whether it would be better to leave it in place.
- When an asset appears on the maintenance schedule, if the cost of performing the maintenance is approaching or has surpassed the value of the asset, it may be time to replace it with a like device or new architecture altogether.
- If the cost of maintaining an asset is more than the cost of replacement, what is the cumulative cost of replacing versus maintaining the entire system that this asset resides within?
- The historical maintenance records should also be included in this calculation, but do not fall for the sunk cost fallacy in wanting to keep something in place. If it is taking excessive maintenance time that is detracting from other opportunities, then it may be time to replace it, regardless of the amount of past money sunk into it.
- If you tabulate and sort the value metrics, perhaps you can apply a simple metric such as capex and available budget to the lowest-value assets for replacement.

- Include both the capex cost of the component and the opex to replace the asset that is in service now.
- Present value and future value calculations also come in to play here as you evaluate possible activity alternatives. These calculations get into the territory of MBAs, but MBAs always have real and relevant numbers to use in the calculations. There is value to stepping back and simply evaluating cost of potential activities.

Activity prioritization often involves equations, algorithms, and costs. It does not always involve predicting the future, but values that feed the equations may be *predicted values* from your models. When you know the amount of time your networking staff spends on particular types of devices, you can develop predictive models that estimate how much future time you will spend on maintaining those devices. Make sure the MBAs include your numbers in their models just as you want to use their numbers in yours.

In industry, activity prioritization may take different forms. You may gain some new perspective from a few of these:

- Company activities should align to the stated mission, vision, and strategy for the company. An individual analytics project should support some program that aligns to that vision, mission, and strategy.
- Companies have limited resources; compare activity benefits with both long-term and short-term lenses to determine the most effective use of resources. Sometimes a behind-the-scenes model that enables a multitude of other models is the most effective in the long term.
- Measuring and sharing the positive impact of prioritization provides further runway to develop supportive systems, such as additional analytics solutions.
- Opportunity cost goes with inverse thinking (refer to Chapter 6). By choosing an activity, what are you choosing *not* to do?
- Prioritize activities that support the most profitable parts of the business first.
- Prioritize activities that have global benefits that may not show up on a balance sheet, such as sustainability. You may have to assign some soft or estimated values here.
- Prioritize activities that have a multiplier effect, such as data sharing. This produces exponential versus linear growth of solutions that help the business.
- Activity-based costing is an exercise that adds value to activity prioritization.
- Project management teams have a *critical path* of activities for the important steps that define project timelines and success. There are projects in every industry, and if you decrease the length of the critical path with analytics, you can help.
- Sales teams in any industry use lift-and-gain analysis to understand potential customers that should receive the most attention. Any industry that has a recurring revenue model can use lift-and-gain analysis to proactively address churn. (*Churn* is covered later in this chapter.)

- Reinforcement learning allows artificial intelligence systems to learn from their experiences and make informed choices about the activity that should happen next.
- Many industries use activity prioritization to identify where to send their limited resources (for example, fraud investigators in the insurance industry).

For your world, you are uniquely qualified to understand and quantify the factors needed to develop activity prioritization models. In defining solutions in this space, you can use the following:

- Mathematical equations, statistics, sorted data, spreadsheets, and algorithms of your own
- Unsupervised machine learning methods for clustering, segmenting, or grouping options or devices
- Supervised machine learning to classify and predict how you expect things to behave, with regression analysis to predict future trends in any numerical values

Asset Tracking

Asset tracking is an industry-agnostic problem. You have things out there that you are responsible for, and each one has some cost and some benefit associated with it. Asset tracking involves using technology to understand what is out there and what it is doing for your business. It is a foundational component of most other analytics solutions. If you have a fully operational data collection environment, asset tracking is the first use case of bringing forward valuable data points for analysis. This includes physical, virtual, cloud workloads, people, and things (IoT). Sometimes in IT networking, this goes even deeper, to the level of software process, virtual machine, container, service asset, or microservices level.

These are the important areas of asset tracking:

- You want to know your inventory, and all metadata for the assets, such as software, hardware, features, characteristics, activities, and roles.
- You want to know where an asset is within a solution, business, location, or criticality context.
- You want to know the available capabilities of an asset in terms of management, control, and data plane access. These planes may not be identified for assets outside IT, but the themes remain. You need to learn about it, understand how it interacts with other assets, and track the function it is performing.
- You want to know what an asset is currently doing in the context of a solution. As you learned in Chapter 3, “Understanding Networking Data Sources,” you can slice some assets into multiple assets and perform multiple functions on an asset or within a slice of the asset.

- You want to know the base physical asset, as well as any virtual assets that are part of it. You want to maintain the relationship knowledge of the virtual-to-physical mapping.
- You want to evaluate whether an asset should be where it is, given your current model of the environment.
- You want an automated way to add new assets to your systems. Microservices created by an automated system are an example in which automation is required. If you are doing virtualization, your IT asset base expands on demand, and you may not know about it.
- You can have perfect service assurance on managed devices, but some unmanaged component in the mix can break your models of the environment.
- You want to know the costs and value to the business of the assets so you can use that information in your soft data calculations.
- You can track the geographic location of network devices by installing an IoT sensor on the devices. Alternatively, you can supply the data as new data that you create and add to your data stores if you know the location.
- You do not need to confine asset tracking to buildings that you own or to network and compute devices and services. Today you can tag anything with a sensor (wireless, mobile, BLE, RFID) and use local infrastructure or the cloud to bring the data about the asset back to your systems.
- IoT vehicle sensors are heavily used in transportation and construction industries already. Companies today can know the exact locations of their assets on the planet. If it is instrumented and if the solution warrants it, you can get real-time telemetry from those assets to understand how they are working.
- You can use group-based asset tracking and location analytics to validate that things that should stay together are together. Perhaps in the construction case, there is a set of expensive tools and machinery that is moving from one job location to another. You can use asset tracking with location analytics to ensure that the location of each piece of equipment is within some predefined range.
- You can use asset tracking for migrations. Perhaps you have enabled handheld communication devices in your environment. The system is only partially deployed, and solution A devices do not work with newer solution B infrastructure. Devices and infrastructure related to solution A or B should stay together. Asset tracking for the old and new solutions provides you with real-time migration status.
- You can use group-based methods of asset tracking in asset discovery, and you can use analytics to determine if there is something that is not showing. For example, if each of your vehicles has four wheels, you should have four tire pressure readings for each vehicle.

- You can use group-based asset tracking to identify too much or too little with resources. For example, if each of your building floors has at least one printer, one closet switch, and telephony components, you have a way to infer what is missing. If you have 1000 MAC addresses in your switch tables but only 5 tracked assets on the floor, where are these MAC addresses coming from?
- Asset tracking—at the group or individual level—is performed in healthcare facilities to track the medical devices within the facility. You can have only so many crash carts, and knowing exactly where they are can save lives.
- Asset tracking is very common in data centers, as it is important to understand where a virtual component may reside on physical infrastructure. If you know what assets you have and know where they are, then you can group them and determine whether a problem is related to the underlay network or overlay solution. You can know whether the entire group is experiencing problems or whether a problem is with one individual asset.
- An interesting facet of asset tracking is tracking software assets or service assets. The existence, count, and correlation of services to the users in the environment is important. If some service in the environment is a required component of a login transaction, and that service goes missing, then it can be determined that the entire login service will be unavailable.
- Casinos sometimes track their chips so they can determine trends in real time. Why do they change game dealers just when you were doing so well? Maybe it is just coincidence. My biased self sees a pattern.
- Most establishments with high-value clients, such as casinos, like to know exactly where their high-value clients are at any given time so that they can offer concierge services and preferential treatment.

Asset tracking is a quick win for you. Before you begin building an analytics solution, you really need to understand what you have to work with. What is the population for which you will be providing analysis? Are you able to get the entire population to characterize it, or are you going to be developing a model and analysis on a representative sample, using statistical inference? Visualizing your assets in simple dashboards is also a quick win because the sheer number of assets in a business is sometimes unknown to management, and they will find immediate value in knowing what is out there in their scope of coverage.

Behavior Analytics

Behavior analytics involves identifying behaviors, both normal or abnormal. Behavior analytics includes a set of activities and a time window within which you are tracking those activities. Behavior analytics can be applied to people, machines, software, devices, or anything else that has a pattern of behavior that you can model. The outputs of behavior analytics are useful in most industries. If you know how something has behaved in the

past, and nothing has changed, you can reasonably expect that it will behave the same way in the future. This is true for most components that are not worn or broken, but it is only sometimes true for people. Behavior analysis is commonly related to transaction analysis. The following are some examples of behavior analytics use cases:

- For people behavior, segment users into similar clusters and correlate those clusters with the transactions that those users should be making.
- Store loyalty cards show buying behavior and location, so they can correlate customer behavior with the experience.
- Airline programs show flying behaviors. Device logs can show component behaviors.
- Location analytics can show where you were and where you are now.
- You can use behavior analytics to establish known good patterns of behavior as baselines or KPIs. Many people are creatures of habit.
- Many IT devices perform a very narrow set of functions, which you can whitelist as normal behavior.
- If your users have specific roles in the company, you can whitelist behaviors within your systems for them. What happens when they begin to stray from those behaviors? You may need a new feature or function.
- You can further correlate behaviors with the location from which they should be happening. For example, if user Joe, who is a forklift operator at a remote warehouse, begins to request access to proprietary information from a centralized HR environment, this should appear as anomalous behavior.
- Correlate the user to the data plane packets to do behavior analytics. Breaking apart network traffic in order to understand the purpose of the traffic is generally not hard to do.
- Associate a user with traffic and associate that traffic with some purpose on the network. By association, you can correlate the user to the purpose for using your network.
- You can use machine learning or simple statistical modeling to understand acceptable behavior for users. For example, Joe the forklift operator happens to have a computer on his desk. Joe comes in every morning and logs in to the warehouse, and you can see that he badged into the door based on your time reporting system to determine normal behavior.
- What happens when Joe the forklift operator begins to access sensitive data? Say that Joe's account accesses such data from a location from which he does not work. This happens during a time when you know Joe has logged in and is reading the news with his morning coffee at his warehouse. Your behavior analytics solution picks this up. Your human SME knows Joe cannot be in two places at once. This is anomaly detection using behavior analysis.

- Learn and train normal behaviors and use classification models to determine what is normal and what is not. Ask users for input. This is how learning spam filters work.
- Customer behavior analytics using location analysis from IoT sensors connecting to user phones or devices is valuable in identifying resource usage patterns. You can use this data to improve the customer experience across many industries.
- IoT beacon data can be used to monitor customer browsing and shopping patterns in a store. Retailers can use creative product placement to ensure that the customer passes every sale.
- Did you ever wonder why the items you commonly buy together are on opposite sides of the store? Previous market basket analysis has determined that you will buy something together. The store may separate these items to different parts of the store, placing all the things it wants to market to you in between.
- How would you characterize your driving behavior? As you have surely seen by now, insurance companies are creating telematics sensors to characterize your driving patterns in data and adjust your insurance rates accordingly.
- How do your customers interact with your company? Can you model this for any benefit to yourself and your customers?
- Behavior analytics is huge in cybersecurity. Patterns of behavior on networks uncover hidden command-and-control nodes, active scans, and footprinting activity.
- Low-level service behavior analytics for software can be used to uncover rootkit, malware, and other non-normal behavior in certain types of server systems.
- You can observe whitelisting and blacklisting behavior in order to evaluate security policy. Is the process, server, or environment supposed to be normally open or normally closed?
- Identify attacks such as DDoS attacks, which are very hard to stop. The behavior is easy to identify if you have packet data to characterize the behavior of the client-side connection requests.
- Consider what you learned in Chapter 5 about bias. Frequency and recency of events of any type may create availability cascades in any industry. These are ripe areas for a quick analysis to compare your base rates and the impact of those events on behaviors.
- Use behavior analytics to generate rules, heuristics, and signatures to apply at edge locations to create fewer outliers in your central data collection systems and attain tighter control of critical environments.
- Reinforcement learning systems learn the best behavior for maximizing rewards in many systems.

Association rules and sequential pattern-matching algorithms are very useful for creating transactions or sequences. You can apply anomaly detection algorithms or simple statistical analysis to the sets of transactions. Image recognition technology has come far enough that many behaviors are learned by observation. You can have a lot of fun with behavior analysis. Call it computerized people watching.

Bug and Software Defect Analysis

In IT and networking today, almost everything is built from software or is in some way *software defined*. The inescapable fact is that software has bugs. It has become another interesting case of correlation and causation. The number of software bugs is increasing. The use of software is increasing. Is this correlated? Of course, but what is the causation? Skills gap in quality software development is a good guess. The growth of available skilled software developers is not keeping up with the need. Current software developers are having to do much more in a much shorter time. This is not a good recipe. Using analytics to identify defects and improve software quality has a lot of value in increasing the productivity of software professionals.

Here is an area where you can get creative by using something you have already learned from this section: asset tracking. You can track skills as assets and build a solution for your skills gap. The following are some ideas for improving your own company's skills gap in software development:

- Use asset tracking to understand the current landscape of technologies in your environment.
- Find and offer free training related to the top-N new or growing technologies.
- Set up behavior analytics to track who is using training resources and who is not.
- Set quality benchmarks to see which departments or groups experience the most negative impact from bugs and software issues.
- Track all of this over time to show how the system worked—or did not work.

This list covers the human side of trying to reduce software issues through organizational education. What can you do to identify and find bugs in production? Obviously, you know where you have had bug impact in production. Outside production, companies commonly use testing and simulation to uncover bugs as well. Using anomaly detection techniques, you can monitor the test and production environments in the following ways:

- Monitor resource utilization for each deployment type. What are the boundaries for good operation? Can tracking help you determine that you are staying within those boundaries for any software resource?
- What part of the software rarely gets used? This is a common place where bugs lurk because you don't get much real-world testing.
- What are the boundaries of what the device running the software can do? Does the software gracefully abide by those boundaries?

- Take a page from hardware testing and create and track counters. Create new counters if possible. Set benchmarks.
- When you know a component has a bug, collect data on the current state of the component at the time of the bug. You can then use this to build labeled cases for supervised learning. Be sure to capture this same state from similar systems that do not show the bug so you have both yes and no cases.

Machine learning is great for pattern matching. Use modeling methods that allow for interpretation of the input parameters to determine what inputs contribute most to the appearance of software issues and defects. Do not forget to include the soft values. Soft values in this case might be assessments of the current conditions, state of the environment, usage, or other descriptions about how you use the software. Just as you are trying to take ideas from other industries to develop your own solutions in this section, people and systems sometimes use software for purposes not intended when it was developed.

As you get more into software analysis, soft data becomes more important. You might observe a need for a soft value such as criticality and develop a mechanism to derive it. Further, you may have input variables that are outputs from other analytics models, as in these examples:

- Use data mining to pull data from ticketing systems that are related to the software defect you are analyzing.
- Use the text analytics components of NLP to understand more about what tickets contain.
- If your software is public or widely used, also perform this data mining on social media sites such as forums and blogs.
- If software is your product, use sentiment analysis on blogs and forums to compare your software to that of competitors.
- Extract sentiment about your software and use that information as a soft value. Be careful about sarcasm, which is hard to characterize.
- Perform data mining on the logging and events produced by your software to identify patterns that correlate with the occurrence of defects.
- With any data that you have collected so far, use unsupervised learning techniques to see if there are particular groupings that are more or less associated with the defect you are analyzing.
- Remember again that correlation is not causation. However, it does aid in your understanding of the problem.

In Cisco Services, many groups perform any and all of the efforts just mentioned to ensure that customers can spend their time more effectively gaining benefit from Cisco devices rather than focusing on software defects. If customers experience more than a single bug in a short amount of time, frequency illusion bias can take hold, and any bug thereafter will take valuable customer time and attention away from running the business.

Capacity Planning

Capacity planning is a cross-industry problem. You can generally apply the following questions with any of your resources, regardless of industry, to learn more about the idea behind capacity planning solutions—and you can answer many of these questions with analytics solutions that you build:

- How much capacity do we have?
- How much of that capacity are we using now?
- What is our consumption rate with that capacity?
- What is our shrink or growth rate with that capacity?
- How efficiently are we using this capacity? How can we be more efficient?
- When will we reach some critical threshold where we need to add or remove capacity from some part of the business?
- Can we re-allocate capacity from low-utilization areas to high-utilization areas?
- Is capacity reallocation worth it? Will this create unwanted change and thrashing in the environment?
- When will it converge back to normal capacity? When will it regress to the mean operational state? Or is this a new normal?
- How much time does it take to add capacity? How does this fit with our capacity exhaustion prediction models?
- Are there alternative ways to address our capacity needs? (Are we building a faster horse when there are cars available now?)
- Can we identify a capacity sweet spot that makes effective use of what we need today and allows for growth and periodic activity bursts?

Capacity planning is a common request from Cisco customers. Capacity planning does not include specific algorithms that solve all cases, but it is linked to many other areas discussed in this chapter. Considerations for capacity planning include the following:

- It is an optimization problem, where you want to maximize the effectiveness of your resources. Use optimization algorithms and use cases for this purpose.
- It is a scheduling problem where you want to schedule dynamic resources to eliminate bottlenecks by putting them in the place with the available capacity.
- Capacity in IT workload scheduling includes available memory, the central processing unit (CPU), storage, data transfer performance, bandwidth, address space, and many other factors.
- Understanding your foundational resource capacity (descriptive analytics) is an asset tracking problem. Use ideas from the “Asset Tracking” section, earlier in this chapter, to improve.

- Use predictive models with historical utilization data to determine run rate and the time to reach critical thresholds for your resources. You know this concept already as you do this with paying your bills with your money resource.
- Capacity prediction may have a time series component. Your back-office resources have a weekday pattern of use. Your customer-facing resources may have a weekend pattern of use if you are in retail.
- Determine whether using all your capacity leads to efficient use of resources or clipping of your opportunities. Using all network capacity for overnight backup is great. Using all retail store capacity (inventory) for a big sale results in your having nothing left to sell.
- Sometimes capacity between systems is algorithmically related. Site-to-site bandwidth depends on the applications deployed at each site. Pizza delivery driver capacity may depend on current promotions, day of week, or sports schedules.
- The well-known traveling salesperson problem is about efficient use of the salesperson's time, increasing the person's capacity to sell if he or she optimizes the route. Consider the cost savings that UPS and FedEx realize in this space.
- How much capacity on demand can you generate? Virtualization using x86 is very popular because it involves using software to create and deploy capacity on demand, using a generalized resource. Consider how Amazon and Netflix as content providers do this.

Sometimes capacity planning is entirely related to business planning and expected growth, so there are not always hard numbers. For example, many service providers build capacity well in excess of current and near-term needs in order to support some upcoming push to rapidly acquire new customers. As with many other solutions, with capacity planning there is some art mixed with the data science.

Event Log Analysis

As more and more IT infrastructure moves to software, the value of event logs from that software is increasing. Virtual (software-defined) components do not have blinky green lights to let you know that they are working properly. Event logs from devices are a rich source of information on what is happening. Sometimes you even receive messages from areas where you had no previous analysis set up. Events are usually syslog sourced, but events can be any type of standardized, triggered output from a device—from IT or any other industry. This is a valuable type of telemetry data.

What can you do with events? The following are some pointers from what is done in Cisco Services:

- Event logs are not always negative events, although you commonly use them to look for negative events. Software developers of some components have configured a software capability to send messages. You can often configure such software to send you messages describing normal activity as well as the negative or positive events.

- Receipt of some type of event log is sometimes the first indicator that a new component has connected to the domain. If you are using standardized templates for deployment of new entities, you may see new log messages arrive when the device comes online because your log receiver is part of the standard template.
- Descriptive statistics are often the first step with log analysis. Top-N logs, components, message types, and other factors are collected.
- You can use NLP techniques to parse the log messages into useful content for modeling purposes.
- You can use classifiers with message types to understand what type of device is sending messages. For example, if new device logs appear, and they show routing neighbor relationships forming, then your model can easily classify the device as a router.
- Mine the events for new categories of what is happening in the infrastructure. Routing messages indicate routing. Lots of user connections up and down at 8 a.m. and 5 p.m. usually indicate an end user-connected device. Activity logs from wireless devices may show gathering places.
- Event log messages are usually sent with a time component, which opens up the opportunities for time-based use cases such as trending, time series, and transaction analysis.
- You can use log messages correlated with other known events at the same time to find correlations. Having a common time component often results in finding the cause of the correlations. A simple example from networking is a routing neighbor relationship going down. This is commonly preceded by a connection between the components going down. Recall that if you don't have a route, you might get black hole routed.
- Over time, you can learn normal syslog activity of each individual component, and you can use that information for anomaly detection. This can be transaction, count, severity, technology, or content based.
- You can use sequential pattern mining on sequences of messages. If you are logging routing relationships that are forming, you can treat this just like a shopping activity or a website clickstream analysis and find incomplete transactions to see when routing neighbor relationships did not fully form.
- Cisco Services builds analysis on the syslog *right side*. Standard logs are usually in the format *standard_category-details_about_the_event*. You build full analysis of a system activity by using NLP techniques to extract the data from the details parts of messages.
- You can build word clouds of common activity from a certain set of devices to describe an area visually.

- Identify sets of messages that indicate a condition. Individual sets of messages in a particular timeframe indicate an incident, and incidents can be mapped to larger problems, which may be collections of incidents.
- Service assurance solutions and Cisco Network Early Warning (NEW) take the incident mapping a step further, recognizing the incident by using sequential pattern mining and taking automated action with automated fault management.
- You can think of event logs as Twitter feeds and apply all the same analysis. Logs are messages coming in from many sources with different topics. Use NLP and sentiment analysis to know how the components feel about something in the log message streams.
- Inverse thinking techniques apply. What components are not sending logs? Which components are sending more logs than normal? Fewer logs than normal? Why?
- Apply location analytics to log messages to identify activity in specific areas.
- Output from your log models can trigger autonomous operations. Cisco uses automated fault management to trigger engagement from Cisco support.
- You can use machine learning techniques on log content, log sequences, or counts to cluster and segment. You can then label the output clusters as interesting or not.
- You can use analytics classification techniques with log data. Add labels to historical data about actionable log messages to create classification models that identify these actionable logs in future streams.

I only cover IT log analysis here because I think IT is leading the industry in this space. However, these log analysis principles apply across any industry where you have software sending you status and event messages. For example, most producers of industrial equipment today enable logging on these devices. Your IoT devices may have event logging capabilities. When the components are part of a fully managed service, these event logs may be sent back to the manufacturer or support partner for analysis. If you own the log-producing devices, you generally get access to the log outputs for your own analysis.

Failure Analysis

Failure analysis is a special case of churn models (covered later in this chapter). When will something fail? When will something churn? The major difference is that you often have many latent factors in churn model, such as customer sentiment, or unknown influences, such as a competitor specifically targeting your customer. You can use the same techniques for failure analysis because you have most of the data, but you may be missing some causal factors. Failure analysis is more about understanding why things failed than about predicting that they will fail or churn. Use both failure and churn analysis for determining when things will fail.

Perform failure analysis when you get detailed data about failures with target variables (labels). This is a supervised learning case because you have labels. In addition to predicting the failure and *time to failure*, getting labeled cases of failure data is extremely

valuable for inferring the factors that most likely led to the failure. Compare the failure patterns and models to the non-failure patterns and models. These models naturally roll over to predictive models, where the presence (or absence) of some condition affects the failure time prediction.

Following are some use cases of failure analysis:

- Why do customers (stakeholders) leave? This is churn, and it is also a failure of your business to provide enough value.
- Why did some line of business decide to bypass IT infrastructure and use the cloud? Where did IT fail, and why?
- Why did device, service, application, or package X fail in the environment? What is different for ones that did not fail?
- Engineering failure analysis is common across many industries and has been around for many years. Engineering failure analysis provides valuable thresholds and boundaries that you can use with your predictive assessments, as you did when looking at the limit of router memory (How much is installed?).
- Predictive failure analysis is common in web-scale environments to predict when you will exceed capacity to the point of customer impact (failure). Then you can use scale-up automation activities to preempt the expected failure.
- Design teams use failure analysis from field use of designs as compared to theoretical use of the same designs. Failure analysis can be used to determine factors that shorten the expected life spans of products in the field. High temperatures or missing earth ground are common findings for electronic equipment such as routers and switches.
- Warranty analysis is used with failure analysis to optimize the time period and pricing for warranties. (Based on the number of consumer product failures that I have experienced right after the warranty has run out, I think there has been some incredible work in this area!)
- Many failure analysis activities involve activity simulation on real or computer-modeled systems. This simulation is needed to generate long term MTBF (mean time between failures) ratings for systems.
- *Failure analysis* is commonly synonymous with *root cause analysis (RCA)*. Like RCA in Cisco Services, failure analysis commonly involves gathering all of relevant information and putting it in front of SMEs. After reading this book, you can apply domain knowledge and a little data science.
- You apply the identified causes and the outputs of failure analysis back to historical data as labels when you want to build analytics models for predicting future failures.

Keep in mind that you can view failure analysis from multiple perspectives, using inverse thinking. Taking the alternative view in the case of line of business using cloud instead of IT, the failure analysis or choice to move to the cloud may have been model or algorithm

based. Trying to understand how the choice was made from the other perspective may uncover factors that you have not considered. Often failures are related to factors that you have not measured or cannot measure. You would have recognized the failure if you had been measuring it.

Information Retrieval

You have access to a lot of data, and you often need to search that data in different ways. Perhaps you are just exploring the data to find interesting patterns. You can build information retrieval systems with machine learning to explore your data. Information retrieval simply provides the ability to filter your massive data to a sorted list of the most relevant results, based on some set of query items. You can search mathematical representations of your data much faster than raw data.

Information retrieval is used for many purposes. Here are a few:

- You need information about something. This is the standard online search, where you supply some search terms, and a closest match algorithm returns the most relevant items to your query. Your query does not have to start with text. It can be a device, an image, or anything else.
- Consider that the search items can be anything. You can search for people with your own name by entering your name. You can search for similar pictures by entering an image. You can search for similar devices by entering a device profile.
- In many cases, you need to find nearest neighbors for other algorithms. You can build the search index out of anything and use many different nearest neighbor algorithms to determine *nearness*.
- For supervised cases, you may want to work on a small subset. You can use nearest neighbor search methods to identify a narrow population by choosing only the nearest results from your query to use for model building.
- Cisco uses information retrieval methods on device fingerprints in order to find similar devices that may experience the same types of adverse conditions.
- Information retrieval techniques on two or more lists are used to find nearest neighbors in different groups. If you enter the same search query into two different search engines that were built from entirely different data, the top-N highly similar matches from both lists are often related in some way as well.
- Use filtering with information retrieval. You can filter the search index items before searching or filter the results after searching.
- Use text analytics and NLP techniques to build your indexes. Topic modeling packages such as Gensim can do much of the work for you. (You will build an index in later chapters of this book.)

- Information retrieval can be automated and used as part of other analytics solutions. Sometimes knowing something about the nearest neighbors provides valuable input to some other solution you are building.
- Information extraction systems go a step further than simple information retrieval, using neural networks and artificial techniques to answer questions. Chatbots are built on this premise.
- Combine information retrieval with topic modeling from NLP to get the theme of the results from a given query.

Information retrieval systems have been popular since the early days of the Internet, when search engines first came about. You can find published research on the algorithms that many companies used. If you can turn a search entry into a document representation, then information retrieval becomes a valuable tool for you. Modern information retrieval is trending toward understanding the context of the query and returning relevant results. However, basic information retrieval is still very relevant and useful.

Optimization

Optimization is one of the most common uses of math and algorithms in analytics. What is the easiest, best, or fastest way to accomplish what you need to get done? While mathematical-based optimization functions can be quite complex and beyond what is covered in this book, you can realize many simple optimizations by using common analytics techniques without having to understand the math behind them.

Here are some optimization examples:

- If you cluster similar devices, you can determine whether they are configured the same and which devices are performing most optimally.
- If you go deep into analytics algorithms after reading this book, you may find that the reinforcement and deep learning that you are reading about right now is about optimizing *reward functions* within the algorithms. You can associate these algorithms with everyday phenomena. How many times do you need to touch a hot stove to train your own reward function for taking the action of reaching out and touching it?
- Optimizing the performance of a network or maximizing the effectiveness of its infrastructure is a common use case.
- Self-leveling wireless networks are a common use case. They involve optimization of both the user experience and the upstream bandwidth. There are underlying signal optimization functions as well.
- *Active-active* load balancing with stateless infrastructure is a data center or cloud optimization that allows N+1 redundancy to take the place of the old 50% paradigm, in which half of your redundant infrastructure sits idle.

- Optimal resource utilization in your network devices is a common use case. Learn about the memory, CPU, and other components of your network devices and find a benchmark that provides optimal performance. Being above such thresholds may indicate performance degradation.
- Optimize the use of your brain, skills, and experience by having consistent infrastructure hardware, software, and configuration with known characteristics around which you can build analysis. It's often the outliers that break down at the wrong times because they don't fit the performance and uptime models you have built for the common infrastructure. This type of optimization helps you make good use of your time.
- As items under your control become outdated, consider the time it takes to maintain, troubleshoot, repair, and otherwise keep them up to date. Your time has an associated cost, which you can seek to optimize.
- Move your expert systems to automated algorithms. Optimize the effectiveness of your own learning.
- Scheduling virtual infrastructure placement usually depends on an optimization function that takes into account bandwidth, storage, proximity to user, and available capacity in the cloud.
- Activity optimization happens in call centers when you can analyze and predict what the operators need to know in order to close calls in a shorter time and put relevant and useful data on the operators screen just when they need it. Customer relationship management (CRM) systems do this.
- You can use pricing optimization to maximize revenues by using factors such as supply and demand, location, availability, and competitors' prices to determine the best market price for your product or service. That hotel next to the football stadium is much more expensive around game day.
- Offer customization is a common use case for pricing optimization. If you are going to do the work to optimize the price to the most effective price, you also want to make sure the targeted audience is aware of it.
- Offer customization combines segmentation, recommendations engines, lift and gain, and many other models to identify the best offer, the most important set of users, and the best time and location to make offers.
- Optimization functions are used with recommender engines and segmentation. Can you identify who is most likely to take your offers? Which customers are high value? Which devices are high value? Which devices are high impact?
- Can you use loyalty cards for IT? Can you optimize the performance and experience of the customers who most use your services?
- Perform supply chain optimization by proactively moving items to where they are needed next, based on your predictive models.

- Optimize networks by putting decision systems closest to the users and putting servers closest to the data and bandwidth consumers.
- Graph theory is a popular method for route optimization, product placement, and product groupings.
- Many companies perform pricing optimization to look for segments that are mis-priced by competitors. Identifying these customers or groups becomes more realistic when they have lifetime value calculations and risk models for the segments.
- Hotels use pricing optimization models to predict the optimal price, based on the activities, load, and expected utilization for the time period you are scheduling.
- IoT sensors can be used to examine soil in fields in order to optimize the environment for growth of specific crops.
- Most oil and gas companies today provide some level of per-well data acquisition, such that extraction rate, temperatures, and pressures are measured for every revenue-producing asset. This data is used to optimize production outputs.

Optimization problems are very good for use cases when you can find the right definition of *optimization*. When you have a definition, you can develop your own algorithm or function to track it by combining with standard analytics algorithms.

Predictive Maintenance

Whereas corrective maintenance is reactive, predictive maintenance is proactive. Predicting when something will break decreases support cost because scheduled maintenance can happen before the item breaks down. Predictive maintenance is highly related to failure analysis, as well as churn or survival models. If you understand and predict when something will churn, and if you understand the factors behind churn, you can sometimes predict the timeframe for churn. In such cases, you can predict when something will break and build predictive maintenance schedules. Perhaps output from a recommender system prioritizes maintenance activities.

Understanding current and past operation and failures is crucial in developing predictive maintenance solutions. One way this is enabled is by putting sensors on everything. The good news is that you already have sensors on your network devices. Almost every company responsible for transportation has some level of sensors on the vehicles. When you collect data that is part of your predictive failure models on a regular basis, predictive maintenance is a natural next step. The following are examples of predictive maintenance use cases:

- Predictive maintenance should be intuitive, based on what you have read in this chapter. Recall from the router memory example that the asset has a resource required for successful operation, and trending of that resource toward exhaustion or breakdown can help predict, within a reasonable time window, when it will no longer be effective.

- *Condition-based maintenance* is a term used heavily in the predictive maintenance space. Maybe something did not fully fail but is reaching a suboptimal condition, or maybe it will reach a suboptimal condition in a predictable amount of time.
- Oil pressure or levels in an engine are like available memory in a router: When the oil level or pressure gets low, very bad things happen. Predicting oil pressure is hard. Modeling router memory is much easier.
- Perform probability estimation to show the probability of when something might break, why it might break, or even whether it might break at all, given current conditions.
- Cluster or classify the items most likely to suffer from failures based on the factors that your models indicate are the largest contributors to failures.
- Statistical process control (SPC) is a field of predictive maintenance related to manufacturing environment that provides many useful statistical multivariate methods to use with telemetry data.
- When using high-volume telemetry data from machines or systems, use neural networks for many applications. High-volume sensor data from IoT environments is a great source of data for neural networks that require a lot of training data.
- Delivery companies have systems of sensors on vehicles to capture data points for predictive maintenance. Use SPC methods with this data. Consider that your network is basically a packet delivery company.
- Use event log analysis to collect and analyze machine data output that is textual in nature. Event and telemetry analysis is a very common source for predictive maintenance models.
- Smart meters are very popular today. No longer do humans have to walk the block to gather meter readings. This digitization of meters results in lower energy costs, as well as increased visibility into patterns and trends in the energy usage, house by house. This same technology is used for smart maintenance activities, through models that associate individual readings or sets of readings to known failure cases.
- When you have collected data and cases of past failures, there are many supervised learning classification algorithms available for deriving failure probability predictions that you can use on their own or as guidance to other models and algorithms.
- Cisco Services builds models that predict the probability of device issues, such as critical bugs and crashes. These models can be used with similarity techniques to notify engineers who help customers with similar devices that their customers have devices with a higher-than-normal risk of experiencing the issue.

Predictive maintenance solutions can create a snowball of success for you. When you can tailor maintenance schedules to avoid outages and failures, you free up time and resources to focus on other activities. From a network operations perspective, this is one of the intuitive next steps after you have your basic asset tracking solution in place.

Predicting Trends

If you ask your family members, friends, and coworkers what analytics means to them, one of the very first answers you are likely to get is that analytics is about trends. This is completely understandable because everyone has been through experiences where trends are meaningful. The idea is generally that something trending in a particular way continues to trend that way if nothing changes. If you can model a recent trend, then you can sometimes predict the future of that trend.

Also consider the following points about trends:

- If you have ever had to buy a house or rent an apartment, you understand the simple trend that a one-bedroom, one-bath dwelling is typically less expensive than a two-bedroom, two-bath dwelling. You can gather data and extrapolate the trend line to get a feel for what a three-bedroom, three-bath dwelling is going to cost you.
- In a simple numerical case, a trend is a line drawn through the chart that most closely aligns to the known data points. Predictive capability is obtained by choosing anything on the x- and y-axes of the chart and taking the value of the line at the point where they meet on the chart. This is linear regression.
- Another common trend area is pattern recognition. Pattern recognition can be used to determine whether an event will occur. For example, if you are employed by a company that's open 8 a.m. to 5 p.m. Monday through Friday, you live 30 minutes from the office, and you like to arrive 15 minutes early, you can reasonably predict that on a Tuesday at 7:30 a.m., you will be sitting in traffic. This is your trend. You are always sitting in traffic on Tuesday at 7:30 a.m.
- While the foregoing are simple examples of pattern recognition and trending, things can get much more complex, and contributing factors (commonly called *features*) can number in the hundreds or thousands, hiding the true conditions that lead to the trend you wish to predict.
- Trends are very important for correlation analysis. When two things trend together, there is correlation to be quantified and measured.
- Sometimes trends are not made from fancy analytics. You may just need to extrapolate a single trend from a single value to gain understanding.
- Trends can be large and abstract, as in market shifts, or small and mathematical, as in housing price trends. Some trends may first appear as outliers when a change is in progress.
- Trends are sometimes helpful in recognizing time changes or seasonality in data. Short-term trend changes may show this, while a confounding longer-term trend may also exist. Beware of local minimums and maximums when looking at trends.
- Use time series analysis to determine effects of some action before, during, or after the action was taken. This is common in network migration and upgrade environments.

- Cisco Services uses trending to understand where customers are making changes and where they are not. Trends of customer activity should correlate to the urgency and security of the recommendations made by service consultants to their customers.
- Use trending and correlation together to determine cause-and-effect relationships. Seek to understand the causality behind trends that you correlate in your own environment.
- Trends can be second- or third-level data, such as speed or acceleration. In this case, you are not interested in the individual or cumulative values but the relative change in value for some given time period. This is the case with trending Twitter topics.
- Your smartphone uses location analytics and common patterns of activity to predict where you might need to be next, based on your past trends of activity.

Trending using descriptive analytics is a foundational use case, as stakeholders commonly want to know what has changed and what has not. You can also use trending from normality for rudimentary anomaly detection. If your daily trend of activity on your website is 1000 visitors that open full sessions and start surfing, a day of 10,000 visitors that only half-open sessions may indicate a DDoS attack. You need to have your base trends in place in order to recognize anomalies from them.

Recommender Systems

You see recommender systems on the front pages of Netflix, Amazon, and many other Internet sites. These systems recommend to you additional items that you may like, based on the items you have chosen to date. At a foundational level, recommender systems identify groups that closely match other groups in some aspect of interest. People who watch *this* watch *that* (Netflix). People who bought *this* also bought *that* (Amazon). It's all the same from intuition and innovation perspectives. A group of users is associated to a group of items. Over time, it is possible to learn from the user selections how to improve the classification and formation of the groups and thus how to improve future recommendations. Underneath, recommender systems usually involve some style of collaborative filtering.

Abstracting intuition further, the spirit of collaborative filtering is to learn patterns shared by many different components of a system and recognizing that these are all collaborators to that pattern. You can find sets that have *most* but not *all* of the pattern and determine that you may need to add more components (items, features, configurations) that allow the system to complete the pattern.

Keep in mind the following key points about recommender systems:

- Collaborative filters group users and items based on machine learned device preference, time preferences, and many other dimensions.
- Solutions dealing with people, preferences, and behavior analytics are also called *social filtering* solutions.

- Netflix takes the analytics solution even further, adding things such as completion rates for shows (whether you watched the whole thing) and your binge progression.
- You can map usage patterns to customer segments of similar usage to determine whether you are likely to lose certain customers in order to form customer churn lists.
- You can group high-value customers based on similar features and provide concierge services to these customers.
- In IT, you can group network components based on roles, features, or functions, and you can determine your high-value groups by using machine learning segmentation and clustering. Then you can match high-priority groups of activities to them for your own activity prioritization system.
- Similar features are either explicit or implicit. Companies such as Amazon and Netflix ask you for ratings so that they can associate you with users who have similar interests, based on explicit ratings. You can implicitly learn or infer things about users and add the things you learn as new variables.
- Amazon and Netflix also practice route optimization to deliver a purchase to you from the closest location in order to decrease the cost of delivery. For Amazon, this involves road miles and transportation. For Netflix it is content delivery.
- Netflix called its early recommender system Cinematch. Cinematch clusters movies and then associates clusters of people to them.
- A recommender system can grow a business and is a high-value place to spend your time learning analytics if you can use it in that capacity. (Netflix sponsored a \$1 million Kaggle competition for a new recommender engine.)
- Like Netflix and Amazon, you can also identify which customer segments are most valuable (based on lifetime value or current value, for example) to your business or department. Can you metaphorically apply this information to the infrastructure you manage?
- Use collaborative filtering to find people who will increase performance (your profit) by purchasing suggested offerings. Find groups of networking components that benefit from the same enhancements, upgrades, or configurations.
- Many suggestions will be on target because many people are alike in their buying preferences. This involves looking at the similarity of the purchasers. Look at the similarity of your networking components.
- People will impulse buy if you catch them in context. Lower your time spend by making sure that your networking groups *buy* everything that your collaborative filters recommend for them during the same change window.
- Many things go together, so surely a purchase of item B may improve your purchase of item A alone. This involves looking at the similarity of the item sets.

- You may find that there is a common hierarchy. You can use such a hierarchy to identify the next required item to recommend. Someone is buying a printer and so needs ink. Someone is installing a router and so needs a software version and a configuration. View these as transactions and use *transaction analysis* techniques to identify what is next.
- Sometimes a single item or type of component is the center of a group. If you like a movie featuring Anthony Hopkins, then you may like other movies that he has done. If you are installing a new router in a known Border Gateway Protocol (BGP) area, then the other BGP items in that same area have a set of configuration items that you want on the newly installed router. You can use a recommender system to create a golden configuration template for the area.
- If you liked one movie about aliens, you may like all movies about aliens. If you need BGP on your router, then you might want to browse all BGP and your associated configuration items that are generally close, such as underlying Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS) routing protocols.
- Some recommendations are valid only during a specific time window. For example, you may buy milk and bread on the same trip to the store, but recommending that you also buy eggs a day later is not useful. Dynamic generation of the groups and items may benefit from a time component.
- In the context of your configuration use case, use recommendation engines to look at clusters of devices with similar configurations in order to recommend missing configurations on some of the devices.
- Examine devices with similar performance characteristics to determine if there are performance-enhancing configurations. Learn and apply these configurations on devices in the same group if they do not currently have that configuration.
- Build recommendations engines to look at the set of features configured at the control plane of the device to ensure that the device is supposed to be performing like the other devices within the cluster in which it falls.
- If you know that people like you also choose to do certain things, how do you find people like you? This is part of Cisco Services fingerprinting solutions. If you fingerprint a snapshot of benchmarked KPIs and they are very similar, you can also look at compliance.
- Next-best-offer analysis determines products that you will most likely want to purchase next, given the products you have already purchased. Next-best-action work in Cisco Services predicts actions that you would take next, given the set of actions that you have already taken. Combined with clustering and similarity analysis, multiple next-best-action options are typically offered.
- Capture the choices made by users to enhance the next-best-action options in future models to improve the validity of the choices. Segmentation and clustering algorithms for both user and item improve as you identify common sets.

- Build recommender systems with lift-and-gain analysis. Lift-and-gain models identify the top customers most likely to buy or respond to ads. Can you turn this around to devices instead of people?
- Have custom algorithms to do the sorting, ranking, or voting against clusters to make recommendations. Use machine learning to do the sorting and then assign some lift-and-gain analysis to apply the recommendations.
- Recall the important IT questions: Where do I spend my time? Where do I spend my money? Can you now build a recommender system based on your own algorithms to identify the best action?
- Convert your expert systems to algorithms in order to apply them in recommender systems. Derive algorithms from the recommendations in your expert systems and offer them as recommended actions.

Recommender systems are very important from a process perspective because they aid in making choices about next steps. If you are building a service assurance system, look for recommendations that you can fully automate. The core concept is to recommend items that limit the options that users (or systems) must review. Presenting relevant options saves time and ultimately increases productivity.

Scheduling

Scheduling is a somewhat broad term in the context of use cases. Workload scheduling in networking and IT involves optimally putting things in the places that provide the most benefit. You are scheduled to be at work during your work hours because you are expected to provide benefit at that time. If you have limited space or need, your schedule must be coordinated with those of others so that the role is always filled but at different times by different resources. The idea behind scheduling is to use data and algorithms to define optimal resource utilization.

Following are some considerations for developing scheduling solutions:

- Workload placement and other IT scheduling use cases are sometimes more algorithmic than analytic, but they can have a prediction component. Simple algorithms such as first come, first served (FCFS), round-robin, and queued priority scheduling are commonly used.
- Scheduling and autonomous operations go together well. For example, if you have a set of cloud servers that you buy to run your business every day from 8 a.m. to 5 p.m., would you buy another set of cloud servers to run some data moving that you do every day from 6 p.m. to 8 a.m.? Of course not. You would use the cloud instances to run the business from 8 a.m. to 5 p.m. and then repurpose them to run the 6 p.m. to 8 a.m. job after the daily work is done.
- In cloud and mass virtualization environments, scheduling of the workload into the infrastructure has many requirements that can be optimized algorithmically. For example, does the workload need storage? Where is that storage?

- How close to the storage should you build your workloads? What is the predicted performance for candidate locations? How close to the user should you place these workloads? What is the predicted experience for each of the options?
- How close should you place this workload to other workloads that are part of the same application overlay?
- Do your high-value stakeholders get different treatment than other stakeholders? Do you have different placement policies?
- CPU and memory scheduling within servers are used to maximize the resources for servers that must perform multiple activities, such as virtualization.
- Scheduling your analytics algorithms to run on tens of CPUs rather than thousands of GPUs can dramatically impact operations of your analytics solutions.
- You can use machine learning and supervised learning to build models of historical performance to use as inputs to future schedules.
- Scheduling and placement go together. Placement choices may have a model themselves, coming from recommender systems or next-best-action models.
- You can use clustering or classification to group your scheduling candidates or candidate locations.

Across industries, scheduling comes in many flavors. Using standard algorithms is common because the cost benefit to squeezing the last bit of performance out of your infrastructure may not be worth it. Focus on scheduling solutions for expensive resources to maximize the value of what you build. For scheduling low-end resources such as x86 servers and workloads, it may be less expensive in the long term to just use available schedulers from your vendors. Workload placement is used in this section for illustration purposes because IT and networking folks are familiar with the paradigms. You can extend these paradigms to your own area of expertise to find additional use cases.

Service Assurance

There are many definitions of *service assurance use cases*. Here is mine: *Service assurance use cases* are systems that assure the desired, promised, or expected performance and operation of a system by working across many facets of that system to keep the system within specification, using fully automated methods. Service assurance can apply to full systems or to subsystems. Many subsystem service assurance solutions are combined into higher-level systems that encompass other important aspects of the system, such as customer or user feedback loops.

The boundary definition of a *service* is subjective, and you often get to choose the boundary required to support the need. As the level of virtualization, segmentation, and cloud usage rises, so does the need for service assurance solutions.

Examples of service assurance use cases include the following:

- Network service assurance systems ensure that consistent and engineering-approved configurations are maintained on devices. This often involves fully automated remediation, using zero-touch mechanisms. In this case, *configuration* is the service being assured. This is common in industry compliance scenarios.
- Foundational network assurance systems include configuration, fault, events, performance, bandwidth, quality of service (QoS), and many other operational areas. A service-level agreement (SLA) defines the service level that must be maintained. The assurance systems maintain a SLA defined level of service using analytics and automation. Not meeting SLAs can result in excess costs if there is a guaranteed level involved.
- A network service assurance system can have an application added to become a new system. Critical business applications such as voice and video should have associated service assurance systems. Each individual application defined as an overlay in Chapter 3 can have an assurance system to provide a minimum level of service for that particular application among all the other overlays. Adding the customer feedback loop is a critical success factor here.
- Use network assurance systems to expand policy and intent into configuration and actions at the network layer. You do not need to understand how to implement the policy on many different types of devices; you just need to ensure that the assurance system has a method to deploy the policies for each device type and the system as a whole. The *service* here is a secure network infrastructure. Well-built network service assurance systems provide true self-healing networks.
- Mobile carriers were among the first industries to build service assurance systems, using analytics to collect data for measuring the current performance of the phone experience. They make automated adjustments to components provided to your sessions to ensure that you get the best experience possible.
- A large part of wireless networking service assurance is built into the system already, and you probably don't notice it. If an access point wireless signal fails, the wireless client simply joins another one and continues to support customer needs. The service here is simply a reliable *signal*.
- To continue the wireless example, think of the many redundant systems you have experienced in the past. Things have just worked as expected, regardless of your location, proximity, or activity. How do these systems provide service assurance for you?

Assurance systems rely on many subsystems coming together to support the fully uninterrupted coverage of a particular service. These smaller subsystems are also composed of subsystems. All these systems are common IT management areas that you may recognize, and all of them are supported by analytics when developing service assurance systems.

The following are some examples of assurance systems:

- Quality assurance systems to ensure that each atomic component is doing what it needs to do when it needs to do it
- Quality control (QC) to ensure that the components are working within operating specifications
- Active service quality assessments to ensure that the customer experience is met in a satisfactory way
- Service-level management to identify the KPIs that must be assured by the system
- Fault and event management to analyze the digital exhaust of components
- Performance management to ensure that components are performing according to desired performance specifications
- Active monitoring and data collection to validate policy, intent, and performance
- SLA management to ensure that realistic and attainable SLAs are used
- Service impact analysis, using testing and simulations of stakeholder activity and what-if scenarios
- Full analytics capability to model, collect, or derive existing and newly developed metrics and KPIs
- Ticketing systems management to collect feedback from systems or stakeholders
- Customer experience management systems to measure and ensure stakeholder satisfaction
- Outlier investigations for KPIs, SLAs, or critical metric misses
- Exit interview process, automated or manual, for lost customers or components
- Benchmark comparison for KPIs, SLAs, or metrics to known industry values

Analytics solutions are pervasive throughout service assurance systems. It may take a few, tens, or hundreds of individual analytics solutions to build a fully automated, smart service assurance system. As you identify and build an analytics use case, consider how the use case can be a subsystem or provide components for systems that support services that your company provides.

Transaction Analysis

Transaction analysis involves the examination of a set of events or items, usually over or within a particular time window. Transactions are either ordered or unordered. Transaction analysis applies very heavily in IT environments because many automated processes are actually ordered transactions, and many unordered sets of events occur together, within a specified time window. Ordered transactions are called *sequential patterns*. The idea behind transaction analysis is that there is a set of items, possibly in a defined flow with interim states, that you can capture as observations for analysis.

Here are some common areas of transaction analysis:

- Many companies do clickstream analysis on websites to determine why certain users drop the shopping cart before purchasing. Successful transactions all the way through to shopping cart and full purchase are examined and compared to unsuccessful transactions, where people started to browse but then did not fully check out.
- You can do this same type of analysis on poorly performing applications on the IT infrastructure by looking at each step of an application overlay.
- In stateful protocols, devices are aware of neighbors to which they are connected. These devices perform capabilities exchange and neighbor negotiation to determine how to use their neighbors to most effectively move data plane traffic.
- This act of exchanging capabilities and negotiating with neighbors by definition follows a very standard process. You can use transaction analysis with event logs to determine that everybody has successfully negotiated this connectivity with neighbors, and there is a fully connected IT infrastructure.
- For neighbors who did not complete the protocol transactions, you can infer that you have a problem in the components or the transport.
- Temporal data mining and sequential pattern analysis look for patterns in data that occur in the same order over the same time period, over and over again.
- Event logs often have a pattern, such as a pattern of syslog messages that are leading to a known sequence of events sequence.
- Any simple *trail* of how people traversed your website is a transaction of steps. Do all trails end at the same place? What is that place, and why do people leave after getting to it? Sequential traffic patterns are used to see the point in the site traversal where people decide to exit. If exit is not desired at this point, then some work can be done to keep them browsing past it. (If it is the checkout page, great!)
- Market basket analysis is a form of unordered transaction analysis. The sets are interesting, but the order does not matter. Apriori and FP growth are two common algorithms examined in Chapter 8 that are used to create association rules from transactions.
- Mobile carriers know what product and services you are using, and they use this information for customer churn modeling. They often know the order in which you are using them as well.
- Online purchase and credit card transactions are analyzed for fraud using transaction analysis.
- In healthcare, a basket or transaction is a group of symptoms of a disease or condition.

- An example of market basket analysis on customer transactions is a drug store recognizing that people often buy beer and diapers together.
- An example of linking customer segments or clusters together is the focus of the story of a major retailer sending pregnancy-related coupons to the home of a girl whose parents did not know she was pregnant. The unsupervised analysis of her market baskets matched up with supervised purchases by people known to be pregnant.
- You can zoom out and analyze transactions as groups of transaction; this process is commonly used in financial fraud detection. Uncommon transactions may indicate fraud. Most payment processing systems perform some type of transaction analysis.
- Onboarding or offloading activities in any industry follow standard procedures that you can track as transactions. You can detect anomalies or provide descriptive statistics about migration processes.
- Attribution modeling involves tracking the origins or initiators of transactions.
- Sankey diagrams are useful for ordered transaction analysis because they show interim transactions. Parallel coordinates charts are also useful because they show the flow among possible alternative steps the flows can take.
- In graph analysis, another form of transaction analysis, ordered and unordered relationships are shown in a node-and-connector format.
- You can combine transaction analysis with time series methods to understand the overall transactions relative to time. Perhaps some transactions are normal during working hours but not normal over the weekend. Conversely, IT change transactions may be rare during working hours but common during recognized change windows.
- If you have a lot of data, you can use recurrent neural networks (RNNs) for a wide variety of use cases where sequence and order of inputs matters, such as language translation. A common sentence could be a common ordered transaction.

Transaction analysis solutions are powerful because they expand your use cases to entire sets and sequences rather than just individual data points. They sometimes involve human activity and so may be messy because human activity and choices can be random at times. Temporal data mining solutions and sequential pattern analysis techniques are often required to get the right data for transaction analysis.

Broadly Applicable Use Cases

This section looks at solutions and use cases that are applicable to many industries. Just as the IT use cases build upon the atomic machine learning ideas, you can combine many of those components with your industry knowledge to create very relevant use cases. Just as before, use the examples in this section to generate new ideas. Recall that this chapter is about generating ideas. If you have any ideas lingering from the last section, write them down and explore them fully before shifting gears to go into this section.

Autonomous Operations

The most notable example of autonomous operations today is the self-driving car. However, solutions in this space are not all as complex as a self-driving car. Autonomous vehicles are a very mature case of preemptive analytics. If a use case can learn about something, make a decision to act, and automatically perform that action, then it is autonomous operations.

Common autonomous solutions in industry today include the following:

- Full service assurance in network solutions. Self-healing networks with full service assurance layers are common among mobile carriers and with Cisco. Physical and virtual devices in networks can and do fail, but users are none the wiser because their needs are still being met.
- GM, Ford, and many other auto manufacturers are working on self-driving cars. The idea here is to see a situation and react to it without human intervention, using reinforcement learning to understand the situation and then take appropriate action.
- Wireless devices take advantage of self-optimizing wireless technology to move you from one access point to another. These models are based on many factors that may affect your experience, such as current load and signal strength. Autonomous operations may include leveling of users across wireless access, based on signal analytics. This optimizes the bandwidth utilization of multiple access points around you.
- Content providers optimize your experience by algorithmically moving the content (such as movies and television) closer to you, based on where you are and on what device you access the content. You are unlikely to know that the video source moved closer to you while you were watching it.
- Cloud providers may move assets such as storage and compute closer together in order to consume fewer resources across the internal cloud networks.
- Chatbots autonomously engage customers on a support lines or in Q&A environments. In many cases of common questions, customers leave a site quite satisfied, unaware that they were communicating with a piece of software.
- In smart meeting rooms, the lights go off when you leave the room, and the temperature adjusts when it senses that you are present.
- Medical devices read, analyze, diagnose, and respond with appropriate measures.
- Advertisers provide the right deal for you when you are in the best place to frame or prime you for purchase of their products.
- Cisco uses automated fault management in services to trigger engagement from Cisco support in a fully automated system.

Can you enable autonomous operations? Sure you can. Do you have those annoying support calls with the same subject and the same resolution? You do not need a chatbot to engage the user in conversation. You need automated remediation. Simply auto-correcting

a condition using preemptive analytics is an example of autonomous operations that you can deploy. You can use predictive models to predict when the correctable event will occur. Then you can use data collection to validate that it has occurred, and you can follow up with automation to correct it. *Occurred* in some cases is not an actual failure event; perhaps instead you need to set a “90% threshold” to trigger your auto-remediation activities. If you want to tout your accomplishments from automated systems, notify users that something broke and you fixed it automatically. Now you are making waves and creating a halo effect for yourself.

Business Model Optimization

Business model optimization is one of the major driving forces behind the growth of innovation with analytics. Many cases of business model optimization have resulted in brand-new companies as people have left their existing companies and moved on to start their own. Their cases are interesting. In hindsight, it is easy to see that status quo bias and the sunk cost fallacy may have played roles in the original employers of these founders not changing their existing business models. Hindsight bias may allow you to understand that change may not have been an option for the original company at the time the ideas were first conceived. Here are some interesting examples of business model optimization:

- A major bank and credit card company was created when someone identified a segment of the population that had low credit ratings yet paid their bills. While working for their former employer, the person who started this company used analytics to determine that the credit scoring of a specific segment was incorrect. A base rate had changed. A previously high-risk segment was now much less risky and thus could be offered lower rates. Management at the existing bank did not want to offer these lower rates, so a new credit card company was formed, with analytics at its core. More of these old models were changed to identify more segments to grow the company.
- You can use business model optimizations within your own company to identify and serve new market segments before competitors do. Also take from this that base rates change as your company evolves. Don’t get stuck on old anchors—either in your brain or in your models.
- A major airline was developed through insights that happy employees are productive employees, and consistent infrastructure reduces operating expenses due to drastically lowered support and maintenance costs.
- A furniture maker found success by recognizing that some people did not want to order and wait for furniture. They were okay with putting it together themselves if they could take it home that day in their own vehicle right after purchase.
- A coffee maker determined that it could make money selling a commodity product if it changed the surroundings game to improve the customer experience with purchasing the commodity.

- Many package shippers and transporters realize competitive advantage by using analytics to perform route optimization.
- Constraint analysis is often used to identify the boundary and bottleneck conditions of current business processes. If you remove barriers, you can change the existing business models and improve your company.
- NLP and text analytics are used for data mining of all customer social media interactions for sentiment and product feedback. This feedback data is valuable for identifying constraints.
- Use Monte Carlo simulation methods to simulate changes to an environment to see the impacts of changed constraints. In a talk with Cisco employees, Adam Steltzner, the lead engineer for the Mars Entry, Descent, and Landing (EDL) project team, said that NASA flew to Mars millions of times in simulations before anything left Earth.
- Conjoint analysis can be used to find the optimal product characteristics that are most valued by customers.
- Companies use yield and price analysis in attempts to manipulate supply and demand. When things are hard to get, people may value them more, as you learned in Chapter 5. A competitor may fill the gap if you do not take action.

Any company that wishes to remain in business should be constantly using analytics for business model optimization of its own business processes. Companies of any size benefit from lean principles. Good use of analytics can help you make the decision to pivot or persevere.

Churn and Retention

Retention value is the value of keeping something or keeping something the way it is. This solution is common among insurance industries, mobile carriers, and anywhere else you realize residual income or benefit by keeping customers. In many cases, you can use analytics and algorithms to determine a retention value (lifetime value) to use in your calculations. In some cases, this is very hard to quantify (for example, employee retention in companies). Retention value is a primary input to models that predict churn, or change of state (for example, losing an existing customer).

Churn prediction is a straightforward classification problem. Using supervised learning, you go back in time, look at activity, check to see who remains active after some time, and come up with a model that separates users who remain active from those who do not. With tons of data, what are the best indicators of a user's likelihood to keep opening an app? You can stack rank your output by using lift-and-gain analysis to determine where you want to prevent churn.

Here is how churn and retention are done with analytics:

- Define churn that is relevant in your space. Is this a customer leaving, employee attrition, network event, or a line of business moving services from your IT department to the cloud?

- After you define churn in the proper context, translate it into a target variable to use with analytics.
- Define retention value for the observations of interest. Sometimes when things cost more than they benefit, you want them to churn.
- Insurance companies that show you prices from competitors that are lower than their prices want you to churn and are taking active steps to help you do it. Your lifetime value to their business is below some threshold that they are targeting.
- Use segmentation and classification techniques to divide segments of your observations (customers, components, services) and rank them. This does not have to be actioned but can be a guide for activity prioritization (churn prevention).
- Churn models are heavily used in the mobile carrier space, as mobile carriers seek to keep you onboard to maximize the utilization of the massive networks that they have built to optimize your experience.
- Along those same lines, churn models are valuable in any space where large up-front investment was made to build a resource (mobile carrier, cable TV, telephone networks, your data center) and return on investment is dependent on paid usage of that resource.
- Churn models typically focus on current assets when the cost of onboarding an asset is high relative to the cost of keeping them. (Replace *asset* with *customer* in this statement, and you have the mobile carrier case.)
- You could develop a system to capture labeled cases of churn to train your churn classifiers. How do you define these labeled cases? One example would be to use customers that have been stagnant for four months. You need a churn variable to build labeled cases of *left* and *stayed* and, sometimes, *upgraded*.
- In networking, you can apply the concepts “had trouble ticket” and “did not have trouble ticket.” If you want to prevent churn, you want to prevent trouble tickets. Status quo bias works in your favor here, as it usually takes a compelling event to cause a churn. Don’t be the reason for that event.
- If you have done good feature engineering, and you gather the right hard and soft data for variables, you can examine the input space of the models to determine contributing factors for churn. Examine them for improvement options.
- Some of these input variables may be comparison to benchmarks, KPIs, SLAs, or other relevant metrics.
- Don’t skip the lifetime value calculation of the model subject. In business, a customer can have a lifetime value assigned. Some customers are lucrative, and some actually cost you money. Some devices are troublesome, and some just work.
- Have you ever wondered why you get that “deep discount to stay” only a few times before your provider (phone, TV, or any other paid service) happily helps you leave? If so, you changed your place in the lifetime value calculation.

- You may want to pay extra attention to the top of your ranks. For high-value customers, concierge services, special pricing, and special treatment are used to maintain existing profitable customers.
- Content providers like Netflix use behavior analysis and activity levels (as well as a few others things) to determine whether you are going to leave the service.
- Readmission in healthcare, recidivism in jails, and renewals for services all involve the same analysis theory: identifying who meets the criteria and whether it is worth being proactive to change something.
- Churn use cases have multiple analytics facets. You need a risk model to see the propensity to churn and a decision model to see whether a customer is valuable enough to maintain.
- Mobile carriers used to use retention value to justify giving you free hardware and locking you into a longer-term contract.
- These calculations underpin Randy Bias's pets versus cattle paradigm of cloud infrastructure. Is it easier to spend many hours fixing a cloud instance, or should you use automation to move traffic off, kill it, and start a new instance? Churn, baby, churn.

If you think you have a use case for this area, you may also benefit from reviewing the methods in the following related areas, which are used in many industries:

- Attrition modeling
- Survival analysis
- Failure analysis
- Failure time analysis
- Duration analysis
- Transition analysis
- Lift-and-gain analysis
- Time-to-event analysis
- Reactivation or renewal analysis

Remember that churn simply means that you are predicting that something will change state. Whether you do something about the pending change depends entirely on the value of performing that change. You can use activity prioritization to prevent some churn.

Dropouts and Inverse Thinking

An interesting area of use case development and innovative thinking is considering what you do not know or did not examine. This is sometimes about the items for which you do not have data or awareness. However, if the items are part of your environment or related to your analysis, you must account for them. Many times these may be the causations

behind your correlations. There is real power in extracting these causations. Other times, inverse thinking involves just taking an adversarial approach and examining all perspectives. An entire focus area of analytics, called *adversarial learning*, is dedicated to uncovering weaknesses in analytical models. (Adversarial learning is not covered in this book, but you might want to research it on your own if you work in cybersecurity.)

Here are some areas where you use inverse thinking:

- Dropout analysis is commonly used in survey, website, and transaction analysis. Who dropped out? Where did they drop out? At what step did they drop out? Where did most people drop out?
- In the data flows in your environment, where did traffic drop off? Why?
- What event log messages are missing from your components? Are they missing because nothing is happening, or is there another factor? Did a device drop out?
- What parts of transactions are missing? This type of inverse thinking is heavily used in website clickthrough analysis, where you identify which sections of a website are not being visited. You may find that this point is where people are stopping their shopping and walking away with no purchase from you.
- Are there blind spots in your analysis? Are there latent factors that you need to estimate, imply, proxy, or guess?
- Are any hotspots overshadowing rare events? Are the rare occurrences more important than the common ones? Maybe you should be analyzing the bottom side outliers instead of top-N.
- Recall the law of small numbers. Distribution analysis techniques are often used to understand what the population looks like. Then you can determine whether your analysis truly represents the normal range or whether you are building an entire solution around outliers.
- For anything with a defined protocol, such as a routing protocol handshake, what parts are missing? Simple dashboards with descriptive analytics are very useful here.
- If you are examining usage, what parts of your system are not being used? Why?
- Who uses what? Why do they use that? Should staff be using the new training systems where you show that only 40% of people have logged in? Why are they not using your system?
- What people did not buy a product? Why did they choose something else over your product? Many businesses uncover new customer segments by understanding when a product is missing important features and then adding required functionality to bring in new customer segments.
- Service impact analysis takes advantage of dropout analysis. By looking across patterns in any type of service or system, bottlenecks can be identified using dropout analysis. If you account for traffic along an entire application path by examining second-by-second traffic versus location in the path, where do you have dropout?

- Dropout is a technique used in deep learning to improve the accuracy of models by randomly dropping some inputs in the model.
- A form of dropout is part of ensemble methods such as random forest, where only some predictors are used in weak learning models that come together for a consensus prediction.
- Inverse thinking analysis includes a category called *inverse problem*. This generally involves starting with the result and modeling the reasons for arriving at that result. The goal is to estimate parameters that you cannot measure by successively eliminating factors.
- Inverse analysis is used in materials science, chemistry, and many other industries to examine why something behaved the way it did. You can examine why something in your network behaved the way it did.
- Failure analysis is another form of inverse analysis that is covered previously in this chapter.

As you develop ideas for analysis with your innovative *why* questions, take the inverse view by asking *why not*. Why did the router crash? Why did the similar router not crash? Inverse thinking algorithms and intuition come in many forms. For use cases you choose to develop, be sure to consider the alternative views even if you are only doing due diligence toward fully understanding the problem.

Engagement Models

With engagement models, you can measure or infer engagement of a subject to a topic. The idea is that the subject has a choice of various options that you want them to do. Alternatively, they could choose to do something else that you may not want them to do. If you can understand the level of engagement, you can determine and sometimes predict options for next steps; this is related to *activity prioritization*.

The following are some examples of engagement models related to analytics:

- Online retailers want a website customer to stay engaged with the website—hopefully all the way through to a shopping cart. (Transaction analysis helps here.)
- If a customer did not purchase, how long was the customer at the site? How much did the customer do? The longer the person is there, the more advertisement revenue possibilities you may have. How can you engage customers longer?
- For location analytics, dwell time is often used as engagement. You can identify that a customer is in the place you want him or her to be, such as in your business location.
- How engaged are your employees? Companies can measure employee engagement by using a variety of methods. The thinking is that engaged employees are productive employees.

- Are employees working on the right things? Some companies define engagement in terms of outcomes and results.
- Cisco Services uses high-touch engagement models to ensure that customers maximize the benefit of their network infrastructure through ongoing optimization.
- Customer engagement at conferences is measured using smartphone apps, social media, and location analytics. Engagement is enhanced with artificial intelligence, chatbots, gaming, and other interesting activities. Given a set of alternatives, you need to make the subject want to engage in the alternative that provides the best mutual benefit.
- When you understand your customers and their engagement, you can use propensity modeling for prediction. Given the engagement pattern, what is likely to happen next, based on what you saw before from similar subjects?
- Note how closely propensity modeling relates to transaction analysis, which is useful in all phases of networking. If you know the first n steps in a transaction that you have seen many times before, you can predict step $n+1$ and, sometimes, the outcome of the transaction.
- Service providers use engagement models to identify the most relevant services for customers or the best action to take next for customers in a specific segment. Engaged customers may have found their ROI and might want to purchase more. Disengaged customers are not getting the value of what they have already purchased.

Engagement models are commonly related to people and behaviors, but it is quite possible to replace *people* with *network components* and use some of the same thinking to develop use cases. Use engagement models with activity prioritization to determine actions or recommendations.

Fraud and Intrusion Detection

Fraud detection is valuable in any industry. Fraud detection is related to anomaly detection because you can identify some anomalous activities as fraud. Fraud detection is a tough challenge because not all anomalous activities are fraudulent. Fraudulent activities are performed by people intending to defraud. The same activity sometimes happens as a mistake or new activity that was not seen before. One of the challenges in fraud detection is to identify the variables and interactions of variables that can be classified as fraud. Once this is done, building classification models is straightforward.

Fraud categories are vast, and there many methods are being tried every day to identify fraud. The following are some key points to consider about fraud detection:

- Anyone or anything can perform abnormal activities.
- Fraudulent actors perform many normal transactions.
- Fraud can be seemingly normal transactions performed by seemingly appropriate actors (forgeries).

- Knowing the points above, you can still use pattern detection techniques and anomaly detection mechanisms for fraud detection cases.
- You can use statistical machine learning to establish normal ranges for activities.
- Do you get requests to approve credit card transactions on your mobile phone when you first use your card in a new city? Patterns outside the normal ranges can be flagged as potential fraud.
- You can use unsupervised clustering techniques to group sets of activities and then associate certain groups with higher fraud rates. Then you can develop more detailed models on that subset to work toward finding clear indicators of fraud.
- If someone is gaming the system, you may find activities in your models that are very normal but higher than usual in volume. These can be fraud cases where some bad actor has learned how to color within the lines. DDoS attacks fall in this category as the transactions can seem quite normal to the entity that is being attacked.
- IoT smart meters can be used with other meters used for similar purposes to detect fraud. If your meter does not report enough minimum usage, you must be using an alternative way to get a service.
- Adversarial learning techniques are used to create simulated fraudulent actors in order to improve fraud detection systems.
- Network- and host-based intrusion detection systems use unsupervised learning algorithms to identify normal behavior of traffic to and from network-connected devices. This can be first-level counts, normal conversations, normal conversation length per conversation type, normal or abnormal handshake mechanisms, or time series patterns, among other things.
- Have you ever had to log in again when you use a new device for content or application access? Content providers know the normal patterns of their music and video users. In addition, they know who paid for the content and on which device.
- Companies monitor access to sensitive information resources and generate models of normal and expected behavior. Further, they monitor movement of sensitive data from these systems for validity. Seeing your collection of customer credit card numbers flowing out your Internet connection is an anomaly you want to know about.
- Context is important. Your credit card number showing up in a foreign country transaction is a huge anomaly—unless you have notified the credit card company that you are taking the trip.
- You can use shrink and theft analytics to identify fraud in retail settings.
- It is common in industry to use NLP techniques to find fraud, including similarity of patents, plagiarism in documents, and commonality of software code.
- You can use lift-and-gain and clustering and segmentation techniques to identify high-probability and high-value fraud possibilities.

Fraud and intrusion detection is a particularly hot area of analytics right now. Companies are developing new and unique ways to combat fraudulent actors. Cisco has many products and services in this space, such as Stealthwatch and Encrypted Traffic Analytics, as well as thousands of engineers working daily to improve the state of the art. Other companies also have teams working on safety online. The pace of advancements in this space by these large dedicated teams is an indicator that this is an area to buy versus build your own. You can build on foundational systems from any vendor using the points from this section. Starting from scratch and trying to build your own will leave you exposed and is not recommended. However, you should seek to add your own analytics enhancements to whatever you choose to buy.

Healthcare and Psychology

Applications of analytics and statistical methods in healthcare could fill a small library—and probably do in some medical research facilities. For example, in human genome research, studies showed that certain people have a genetic predisposition to certain diseases. Knowing about this predisposition, a person can be proactive and diligent about avoiding risky behavior. The idea behind this concept was used to build the fingerprint example in the use cases of this book.

Here are a few examples of using analytics and statistics in healthcare and psychology:

- A cancer diagnosis can be made by using anomaly detection with image recognition to identify outliers and unusual data in scans.
- Psychology uses dimensionality reduction factor analysis techniques to identify unknown characteristics that appear to be unknown traits that may not be reflected in the current data collection. This is common in trying to measure intelligence, personality, attitudes and beliefs, and many other soft skills.
- Anomaly detection is used in review of medical claims, prescription usage, and Medicare fraud. It helps determine which cases to identify and call out for further review.
- Drug providers use social media analytics and data mining to predict where they need additional supplies of important products, such as flu vaccines. This is called *diagnostic targeting*.
- Using panel data (also called longitudinal data) and related analysis is very common for examining effects of treatments on individuals and groups. You can examine effects of changes on individuals or groups of devices in your network by using these techniques.
- Certain segments of populations that are especially predisposed to a condition can be identified based on traits (for example, sickle cell traits in humans).
- Activity prioritization and recommender systems are used to suggest next-best actions for healthcare professionals. Individual care management plans specific to individuals are created from these systems.

- Transaction analysis and sequential pattern mining techniques are used to identify sequences of conditions from medical monitoring data that indicate patients are trending toward a known condition.
- Precision medicine is aimed at providing care that is specific to a patient's genetic makeup.
- Preventive health management solutions are used to identify patients who have a current condition with a set of circumstances that may lead to additional illness or disease. (Similarly, when your router reaches 99%, it may be ready to crash.)
- Analytics can be used to determine which patients are at risk for hospital readmission.
- Consider how many monitors and devices are used in healthcare settings to gather data for analysis. As you wish to go deeper with analytics, you need to gather deeper and more granular data using methods such as telemetry.
- Electronic health records are maintained for all patients so that healthcare providers can learn about the patients' histories. (Can you maintain a history of your network components using data?)
- Electronic health records are perfect data summaries to use with many types of analytics algorithms because they eliminate the repeated data collection phase, which can be a challenge.
- Anonymized data is shared with healthcare researchers to draw insights from a larger population. Cisco Services has used globally anonymized data to understand more about device hardware, software, and configuration related to potential issues.
- Evidence-based medicine is common in healthcare for quickly diagnosing conditions. You already do this in your head in IT, and you can turn it into algorithms. The probability of certain conditions changes dynamically as more evidence is gathered.
- Consider the inverse thinking and opportunity cost of predictive analytics in healthcare. Prediction and notification of potential health issues allows for proactivity, which in turn allows healthcare providers more time to address things that cannot be predicted.

These are just a few examples in the wide array of healthcare-related use cases. Due to the high value of possible solutions (making people better, saving lives), healthcare is rich and deep with analytics solutions. Putting on a metaphoric thinking hat in this space related to your own healthcare experiences will surely bring you ideas about ways to heal your sick devices and prevent illness in your healthy ones.

Logistics and Delivery Models

The idea behind logistics and delivery use cases is to minimize expense by optimizing delivery. Models used for these purposes are benefiting greatly from the addition of data-producing sensors, radio frequency identification (RFID), the Global Positioning System

(GPS), scanners, and other facilities that offer near-real-time data. You can associate some of the following use cases to moving data assets in your environment:

- Most major companies use some form of supply chain analytics solutions. Many are detailed on the Internet.
- Manufacturers predict usage and have raw materials arrive at just the right time so they can lower storage costs.
- Transportation companies optimize routing paths to minimize the time or mileage for delivering goods, lowering their cost of doing business.
- Last-mile analytics focuses on the challenges of delivering in urban and other areas that add time to delivery. (Consider your last mile inside your virtualized servers.)
- Many logistics solutions focus on using the *fast path*, such as choosing highways over secondary roads or avoiding left turns. Consider your fast paths in your networks.
- Project management uses the *critical path*—the fastest way to get the project done. There are analysis techniques for improving the critical path.
- Sensitive goods that can be damaged are given higher priority, much as sensitive traffic on your network is given special treatment. When it is expensive to lose a payload, the extra effort is worth it. (Do you have expensive-to-lose payloads?)
- Many companies use Monte Carlo simulation methods to simulate possible alternatives and trade-offs for the best options.
- The traveling salesperson problem mentioned previously in this chapter is a well-known logistics problem that seeks to minimize the distance a salesperson must travel to reach some number of destinations in the shortest time.
- Consider logistics solutions when you look at scheduling workloads in your data center and hybrid cloud environments because determining the best distance (shortest, highest bandwidth, least expensive) is a deployment goal.
- Computer vision, image recognition, and global visibility are used to avoid hazards for delivery. Vision is also used to place an order to fill a store shelf that is showing low inventory.
- Predictive analytics and seasonal forecasting can be used to ensure that a system has enough resources to fill the demand. (You can use these techniques with your virtualized servers.)
- Machine learning algorithms search for patterns in variably priced raw materials and delivery methods to identify the optimal method of procurement.
- Warehouse placement near centers of densely clustered need is common. “Densely clustered” can be a geographical concept, but it could also be a cluster of time to deliver. A city may show as a dense cluster of need, but putting a warehouse in the middle of a city might not be feasible or fast.

From a networking perspective, your job is delivery and/or supply of packets, workloads, security, and policy. Consider how to optimize the delivery of each of these. For example, deploying policy at the edge of the network keeps packets that are eventually dropped off your crowded roads in your cities (data centers). Path optimization techniques can decrease latency and/or maximize bandwidth utilization in your networks.

Reinforcement Learning

Reinforcement learning is a foundational component in artificial intelligence, and use cases and advanced techniques are growing daily. The algorithms are rooted in neural networks, with enhancements added based on the specific use case. Many algorithms and interesting use cases are documented in great detail in academic and industry papers. This type of learning provides benefits in any industry with sufficient data and automation capabilities.

Reinforcement learning can be a misleading name in analytics. It is often thought that reinforcement learning is simply adding more higher-quality observations to existing models. This can improve the accuracy of existing models, but it is not true reinforcement learning; rather, it is adding more observations and generating a better model with additional inputs. True reinforcement learning is using neural networks to learn the best action to take. Reinforcement learning algorithms choose actions by using an inherent reward system that allows them to develop maximum benefit for choosing a class or an action. Then you let them train a very large number of times to learn the most rewarding actions to take. Much as human brains have a dopamine response, reinforcement learning is about learning to maximize the rewards that are obtained through sequences of actions.

The following are some key points about reinforcement learning:

- Reinforcement learning systems are being trained to play games such as backgammon, chess, and go better than any human can play them.
- Reinforcement learning is used for self-driving cars and self-flying planes and helicopters (small ones).
- Reinforcement learning can manage your investment portfolio.
- Reinforcement learning is used to make humanoid robots work.
- Reinforcement learning can control a single manufacturing process or an entire plant.
- Optimal control theory-based systems seek to develop a control law to perform optimally by reducing costs.
- Utility theory from economics seeks to rank possible alternatives in order of preference.
- In psychology, the classical conditioning systems and Pavlov's dog research was about associating stimuli with anticipated rewards.
- Operations research fields in all disciplines seek to reduce cost or time spent toward some final reward.

Reinforcement learning, deep learning, adversarial learning, and many other methods and technologies are being heavily explored across many industries at the time of writing. Often these systems replace a series of atomic machine learning components that you have painstakingly built by hand—if there is enough data available to train them. You will see some form of neural network–rooted artificial intelligence based on reinforcement learning in many industries in the future.

Smart Society

Smart society refers to taking advantage of connected devices to improve the experiences of people. Governing bodies and companies are using data and analytics to improve and optimize the human experience in unexpected ways. Here are some creative solutions in industry that are getting the *smart* label:

- Everyone has a device today. *Smart cities* track concentrations of people by tracking concentrations of phones, and they adjust the presence of safety personnel accordingly.
- Smart cities share *people hotspots* with transportation partners and vendors to ensure that these crowds have access to the common services required in cities. (This sounds like an IT scale-up solution.)
- *Smart energy* solutions work in many areas. Nobody in the room? Time to turn out the lights and turn down the heat. Models show upcoming usage? Start preparing required systems for rapid human response.
- *Smart manufacturing* uses real-time process adjustments to eliminate waste and rework. Computers today can perform SPC in real time, making automated adjustments to optimize the entire manufacturing process.
- *Smart agriculture* involves using sensors in soil and on farm equipment, coupled with research and analytics about the optimum growing environment for the desired crop. Does the crop need water? Soil sensors tell you whether it does.
- *Smart retail* is about optimizing your shopping experience as well as targeted market. If you are standing in front of something for a long time in the store, maybe it's time to send you a coupon.
- *Smart health* is evolving fast as knowledge workers replace traditional factory workers. We are all busy, and we need to optimize our time, but we also need to stay healthy in sedentary jobs. We have wearables that communicate with the cloud. We are not yet in The Matrix, but we are getting there.
- *Smart mobility and transportation* is about fleet management, traffic logistics and improvement, and connected vehicles.
- *Smart travel* makes it easier than ever before to optimize a trip. Have you ever used Waze? If so, you have been an IoT sensor enabling the smart society.
- I do not know of any use cases of combined smart cities and self-driving cars. However, I am really looking forward to seeing these smart technologies converge.

The algorithms and intuitions for the related solutions are broad and wide, but you can gain inspiration by using metaphoric thinking techniques. *Smart* in this case means aiding or making data-driven decisions using analytics. You can use the *smart* label on any of your solutions where you perform autonomous operations based on outputs of analytics solutions that you build. Can you build smart network operations?

Some Final Notes on Use Cases

As you learned in Chapters 5 and 6, experience, bias, and perspective have a lot to do with how you see things. They also have a lot to do with how you name the various classes of analytics solutions. I have used my own perspective to name the use cases in this chapter, and these names may or may not match yours. This section includes some commonly used names that were not given dedicated sections in the chapter.

The Internet of Things is evolving very quickly. I have tried to share use cases within this chapter, but there are not as many today as there will be when the IoT fully catches on. At that point, IoT use cases will grow much faster than anyone can document them. Imagine that everything around you has a sensor in it or on it. What could you do with all that information? A lot.

You can find years of *operations research* analytics. This is about optimizing operations, shortening the time to get jobs done, increasing productivity, and lowering operational cost. All these processes aim to increase profitability or customer experience. I do not use the terminology here, but this is very much in line with questions related to where to spend your time and budgets.

Rules, heuristics, and signatures are common enrichments for deriving some variables used in your models, as standalone models, or as part of a system of models. Every industry seems to have its own taxonomy and methodology. In many *expert systems* deployments today, you apply these to the data in a production environment. *Known attack vectors* and *security signatures* are common terms in the security space. *High memory utilization* might be the name of the simple rule/model you created for your suspect router memory case. From my perspective, these are cases of known good models. When you learn a signature of interest from a known good model, you move it into your system and apply it to the data, and it provides value. You can have thousands of these simple models. These are excellent inputs to next-level models.

Summary

In Chapter 5, you gained new understanding of how others may think and receive the use cases that you create. You also learned how to generate more ideas by taking the perspectives of others. Then you opened your mind beyond that by using creative thinking and innovation techniques from Chapter 6.

In this chapter, you had a chance to employ your new innovation capability as you reviewed a wide variety of possible use cases in order to expand your available pool of ideas. Table 7-1 provides a summary of what you covered in this chapter.

Table 7-1 *Use Case Categories Covered in This Chapter*

Machine Learning and Statistics Use Cases	Common IT Analytics Use Cases	Broadly Applicable Use Cases
Anomalies and outliers	Activity prioritization	Autonomous operations
Benchmarking	Asset tracking	Business model optimization
Classification	Behavior analytics	Churn and retention
Clustering	Bug and software defect analysis	Dropouts and inverse thinking
Correlation	Capacity planning	Engagement models
Data visualization	Event log analysis	Fraud and intrusion detection
Natural language processing	Failure analysis	Healthcare and psychology
Statistics and descriptive analytics	Information retrieval	Logistics and delivery models
Time series analysis	Optimization	Reinforcement learning
	Predictive maintenance	Smart society
	Predicting trends	
	Recommender systems	
	Scheduling	
	Service assurance	
	Transaction analysis	

You should now have an idea of the breadth and depth of analytics use cases that you can develop. You are making a great choice to learn more about analytics.

Chapter 8 moves back down into some details and algorithms. At this point, you should take the time to write down any new things you want to try and also review and refresh anything you wrote down before now. You will gain more ideas in the next chapter, primarily related to algorithms and solutions. This may or may not prime you for additional use-case ideas. In the next chapter, you will begin to refine your ideas by finding algorithms that support the intuition behind the use cases you want to build.

This page intentionally left blank

Analytics Algorithms and the Intuition Behind Them

This chapter reviews common algorithms and their purposes at a high level. As you review them, challenge yourself to understand how they match up with the use cases in Chapter 7, “Analytics Use Cases and the Intuition Behind Them.” By now, you should have some idea about areas where you want to innovate. The purpose of this chapter is to introduce you to candidate algorithms to see if they meet your development goals. You are still innovating, and you therefore need to consider how to validate these algorithms and your data to come together in a unique solution.

The goal here is to provide the intuition behind the algorithms. Your role is to determine if an algorithm fits the use case that you want to try. If it does, you can do further research to determine how to map your data to the algorithm at the lowest levels, using the latest available techniques. Detailed examination of the options, parameters, estimation methods, and operations of the algorithms in this section is beyond the scope of this book, whose goal is to get you started with analytics. You can find entire books and abundant Internet literature on any of the algorithms that you find interesting.

About the Algorithms

It is common to see data science and analytics summed up as having three main areas: classification, clustering, and regression analysis. You may also see machine learning described as supervised, unsupervised, and semi-supervised. There is much more involved to developing analytics solutions, however. You need to use these components as building blocks combined with many other common activities to build full solutions. For example, clustering with data visualization is powerful. Statistics are valuable as model inputs, and cleaning text for feature selection is a necessity. You need to employ many supporting activities to build a complete system that supports a use case. Much of the time, you need to use multiple algorithms with a large supporting cast of other activities—rather like extras in a movie. Remove the extras, and the movie is not the same. Remove the supporting activities in analytics, and your models are not very good either.

This chapter covers many algorithms and the supporting activities that you need to understand to be successful. You will perform many of these supporting activities along with the foundational clustering, classification, regression, and machine learning parts of analytics. Short sections are provided for each of them just to give you a basic awareness of what they do and what they can provide for your solutions. In some cases, there is more detail where it is necessary for the insights to take hold. The following topics are explored in this chapter:

- Understanding data and statistical methods as well as the math needed for analytics solutions
- Unsupervised machine learning techniques for clustering, segmentation, transaction analysis, and dimensionality reduction
- Supervised learning for classification, regression, prediction, and time series analysis
- Text and document cleaning, encoding, topic modeling, information retrieval, and sentiment analysis
- A few other interesting concepts to help you understand how to evaluate and use the algorithms to develop use cases

Algorithms and Assumptions

The most important thing for you to understand about proven algorithms is that the input requirements and assumptions are critical to the successful use of an algorithm. For example, consider this simple algorithm to predict height:

Function (gender, age, weight) = height

Assume that gender is categorical and should be male or female, age ranges from 1 to 90, and weight ranges from 1 to 500 pounds. The values dog or cat would break this algorithm. Using an age of 200 or weight of 0 would break the algorithm as well. Using the model to predict the height of a cat or dog would give incorrect predictions. These are simplified examples of assumptions that you need to learn about the algorithms you are using. Analytics algorithms are subject to these same kinds of requirements. They work within specific boundaries on certain types of data. Many models have sweet spots in terms of the type of data on which they are most effective.

Always write down your assumptions so you can go back and review them after you journey into the algorithm details. Write down and validate exactly how you think you can fit your data to the requirements of the algorithm. Sometimes you can use an algorithm to fit your purpose as is. If you took the gender, age, and weight model and trained it on cats and dogs instead of male and female, then you would find that it is generally accurate for predictions because you used the model for the same kind of data for which you trained it.

For many algorithms, there may be assumptions of normally distributed data as inputs. Further, there may be expectations that variance and standard deviations are normal across the output variables such that you will get normally distributed residual errors from your models. Transformation of variables may be required to make them fit the inputs as required by the algorithms, or it may make the model algorithms work better. For example if you have nonlinear data but would like to use linear models, see if some transformation, such as $1/x$, x^2 , or $\log(x)$, makes your data appear to be linear. Then use the algorithms. Don't forget to convert the values back later for interpretation purposes. You will convert text to number representations to build models, and you will convert them back to display results many, many times as you build use cases.

This section provides selected analytics algorithms used in many of the use cases provided in Chapter 7. Now that you have ideas for use cases, you can use this chapter to select algorithm classes that perform the analyses that you want to try on your data. When you have an idea and an algorithm, you are ready to move to the low-level design phase of digging into the details of your data and the models requirements to make the most effective use of them together.

Additional Background

Here are some definitions that you should carry with you as you go through the algorithms in this chapter:

- **Feature selection**—This refers to deciding which features to use in the models you will be building. There are guided and unguided methods. By contrast, feature engineering involves getting these features ready to be used by models.
- **Feature engineering**—This means massaging the data into a format that works well with the algorithms you want to use.
- **Training, testing, and validating a model**—In any case where you want to characterize or generalize the existing environment in order to predict the future, you need to build the model on a set of *training* data (with output labels) and then apply it on *test* data (also with output labels) during model building. You can build a model to predict perfectly what happens in training data because the models are simply mathematical representations of the training data. During model building, you use test data to optimize the parameters. After optimizing the model parameters, you apply models to previously unseen *validation* data to assess models for effectiveness. When only a limited amount of data is available for analysis, the data may be split three ways into training, testing, and validation data sets.
- **Overfitting**—This means developing a model that perfectly characterizes the training and test data but does not perform well on the validation set or on new data. Finding the right model that best generalizes something without going too far and overfitting to the training data is part art and part science.

- **Interpreting models**—Interpreting models is important. You may also call it *model explainability*. Once you have a model, and it makes a prediction, you want to understand the factors from the input space that are the largest contributors to that prediction. Some algorithms are very easy to explain, and others are not. Consider your requirements when choosing an algorithm. For example, Neural networks are powerful classifiers, but they are very hard to interpret. Random Forest models are easy to interpret.
- **Statistics, plots, and tests**—You will encounter many statistics, plots, and tests that are specific to algorithms as you dig into the details of the algorithms in which you are interested. In this context, *statistic* means some commonly used value, such as an *F statistic*, which is used during the evaluation of differences between the means of two populations. You may use a *q-q plot* to evaluate quantiles of data, or a *Breusch–Pagan test* to produce another statistic that you use to evaluate input data during model building. Data science is filled with these useful little nuggets. Each algorithm and type of analysis may have many statistics or tests available to validate accuracy or effectiveness.

As you find topics in this chapter and perform your outside research, you will read about a type of bias that is different from the cognitive bias that examined in Chapter 5, “Mental Models and Cognitive Bias.” The bias encountered with algorithms is bias in data that can cause model predictions to be incorrect. Assume that the center circles in Figure 8-1 are the true targets for your model building. This simple illustration shows how bias and variance in model inputs can manifest in predictions made by those models.

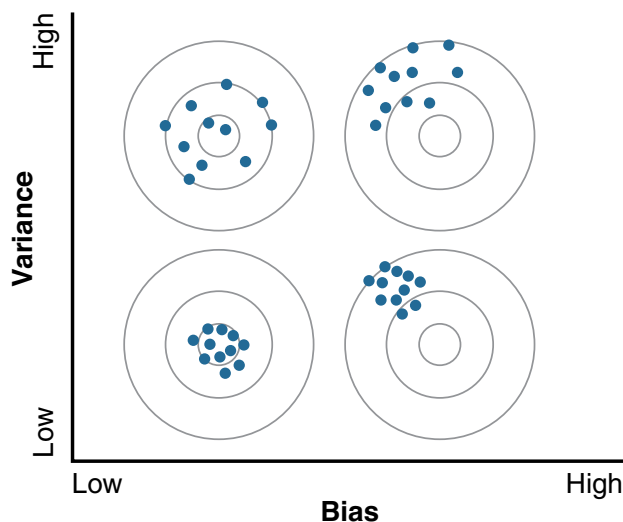


Figure 8-1 *Bias and Variance Comparison*

Interestingly, the purpose of exploring cognitive bias in this book was to make you think a bit outside the box. That concept is the same as being a bit outside the circle in these

diagrams. Using bias for innovation purposes is acceptable. However, bias is not a good thing when building mathematical models to support business decisions in your use cases.

Now that you know about assumptions and have some definitions in your pocket, let's get started looking at what to use for your solutions.

Data and Statistics

In earlier chapters you learned how to collect data. Before we get into algorithms, it is important for you to understand how to explore and represent data in ways that fit the algorithms.

Statistics

When working with numerical data, such as counters, gauges, or counts of components in your environment, you get a lot of quick wins. Just presenting the data in visual formats is a good first step that allows you to engage with your stakeholders to show progress.

The next step is to apply statistics to show some other things you can do with the data that you have gathered. Descriptive analytics that describe the current state is required in order to understand changes from the past states to current state and to predict the trends into the future. Descriptive statistics include a lot of numerical and categorical data points. There is a lot of power in the numbers from descriptive analytics.

You are already aware of the standard measures of central tendency, such as mean, median, and mode. You can go further and examine interquartile ranges by splitting the data into four equal boundaries to find the 25% bottom and top and the 50% middle values. You can quickly visualize statistics by using box-and-whisker plots, as shown in Figure 8-2, where the interquartile ranges and outer edges of the data are defined. Using this method, you can identify rare values on the upper and lower ends. You can define outliers in the distribution by using different measures for upper and lower bounds. I use the $1.5 \times \text{IQR}$ range in this Figure 8-2.

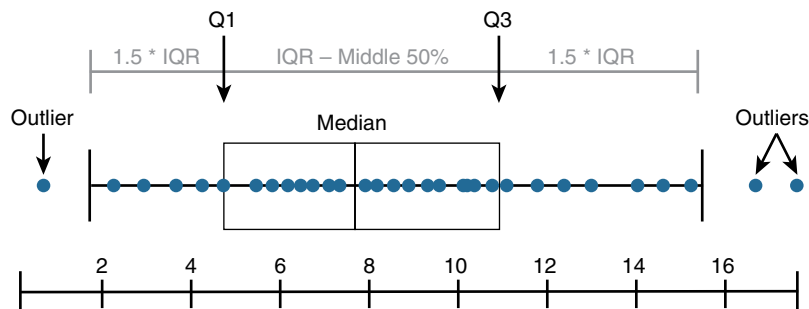


Figure 8-2 Box Plot for Data Examination

You can develop boxplots side by side to compare data. This allows you to take a very quick and intuitive look at all the numerical data values. For example, if you were to plot memory readings from devices over time and the plots looked like the examples in Figure 8-3, what could you glean? You could obviously see a high outlier reading on Device 1 and that Device 4 has a wide range of values.

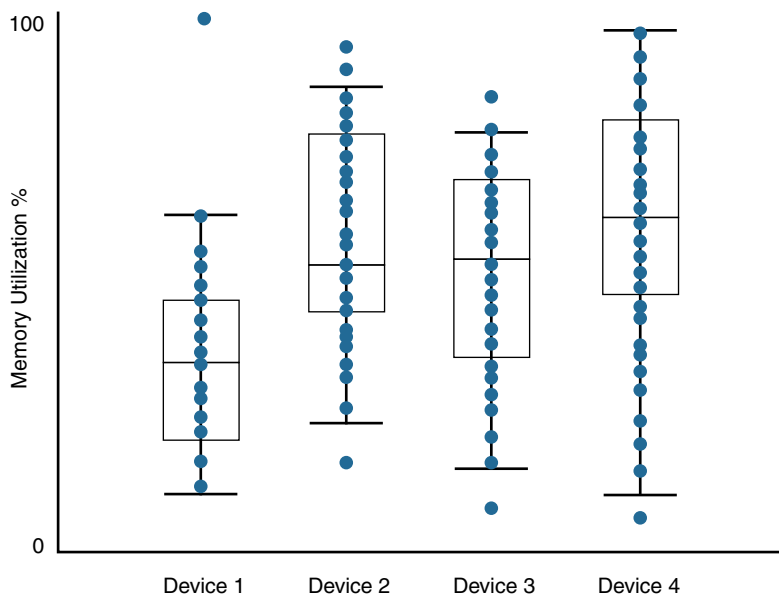


Figure 8-3 *Box Plot for Data Comparison*

You often need to understand the distribution of variables in order to meet assumptions for analytics models. Many algorithms work best with (and some require) a normal distribution of inputs. Using box plots is a very effective way to analyze distributions quickly and in comparison. Some algorithms work best when data is all in the same range of values. You can use transformations to get your data in the proper ranges and box plots to validate the transformations.

Plotting the counts of your discrete numbers allows you to find the distribution. If your numbers are represented as continuous, you can transform or round them to get discrete representations. When things are normally distributed, as shown in Figure 8-4, mean, median, and mode might be the same. Viewing the *count* of values in a distribution is very common. Distributions are not the values themselves but instead the counts of the bins or values stacked up to show concentrations. Perhaps Figure 8-4 is a representation of everybody you know, sorted and counted by height from 4 feet tall to 7 feet tall. There will be many more counts at the common ranges between 5 and 6 feet in the middle of the distribution. Most of the time distributions are not as clean. You will see examples of skewed distributions in Chapter 10, “Developing Real Use Cases: The Power of Statistics.”

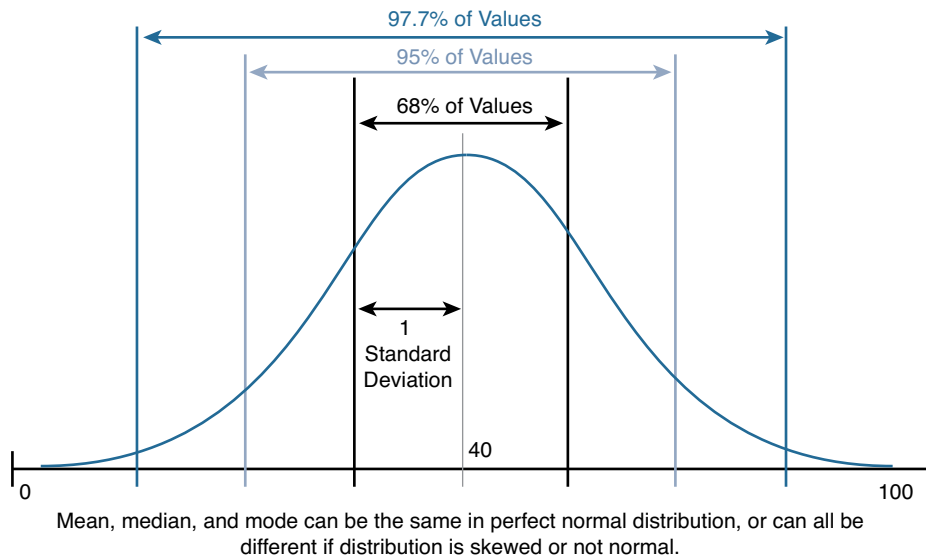


Figure 8-4 *The Normal Distribution and Standard Deviation*

You can calculate standard deviation as a measure of distance from the mean to learn how tightly grouped your values are. You can use standard deviation for anomaly detection. Establishing a normal range over a given time period or time series through statistical anomaly detection provides a baseline, and values outside normal can be raised to a higher-level system. If you defined the boundaries by standard deviations to pick up the outer 0.3% as outliers, you can build anomaly detection systems that identify the outliers as shown in Figure 8-5.

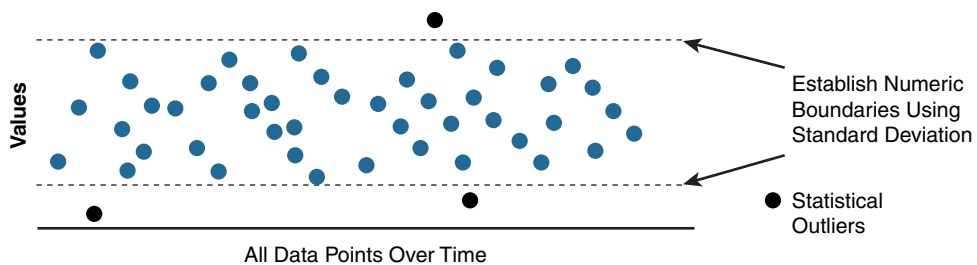


Figure 8-5 *Statistical Outliers*

If you have a well-behaved normal range of numbers with constant variance, statistical anomaly detection is an easy win. You can define confidence intervals to identify the probability that future data from the same population will fall inside or outside the anomaly lines in Figure 8-5.

Correlation

Correlation is simply a relationship between two things, with or without causation. There are varying degrees of correlation, as shown in the simple correlation diagrams in Figure 8-6. Correlations can be perfectly positive or negative relationships, or they can be anywhere in between.

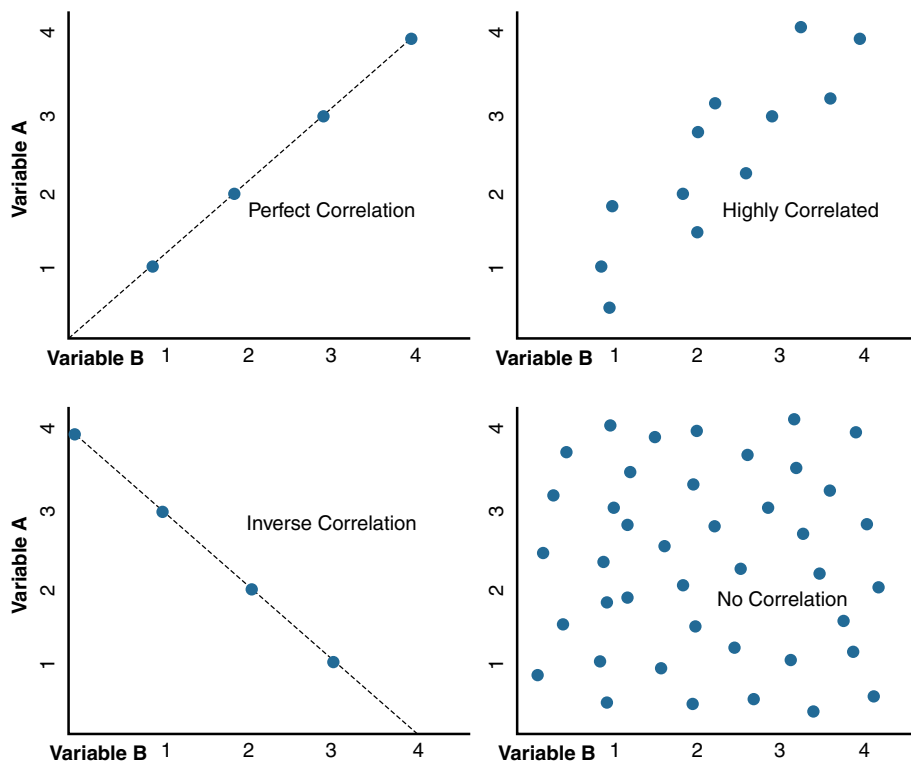


Figure 8-6 *Correlation Explained*

In analytics, you measure correlations between values, but causation must be proven separately. Recall from Chapter 5 that ice cream sales and drowning death numbers can be correlated. But one does not cause the other. Correlation is not just important for finding relationships in trends that you see on a diagram. For model building in analytics, having correlated variables adds complexity and can lower the performance of many types of models. Always check your variables for correlation and determine if your chosen algorithm is robust enough to handle correlation; you may need to remove or combine some variables.

The following are some key points about correlation:

- Correlation can be negative or positive, and it is usually represented by a numerical positive or negative value between 0 and 1.

- Correlation applies to more than just simple numbers. Correlation is the relative change in one variable with respect to another, using many mathematical functions or transformations. The correlation may not always be linear.
- When developing models, you may see correlations expressed as Pearson's correlation coefficient, Spearman's rank, or Kendall's tau. These are specific tests for correlation that you can research. Each has pros and cons, depending on the type of data that is being analyzed. Learning to research various tests and statistics will be commonplace for you as you learn. These are good ones to start with.
- Anscombe's quartet is a common and interesting case that shows that correlation alone may not characterize data well. Perform a quick Internet search to learn why.
- Correlation as measured within the predictors in regression models is called *collinearity* or *multicollinearity*. It can cause problems in your model building and affect the predictive power of your models.

These are the underpinnings of correlation. You will often need to convert your data to numerical format and sometimes add a time component to correlate the data (for example, the number of times you saw high memory in routers correlated with the number of times routers crashed). If you developed a separate correlation for every type of router you have, you would find high correlation of *instances of high memory utilization to crashes* only in the types that exhibit frequent crashes. If you collected instances over time, you would segment this type of data by using a style of data collection called *longitudinal data*.

Longitudinal Data

Longitudinal data is not an algorithm, but an important aspect of data collection and statistical analysis that you can use to find powerful insights. Commonly called *panel data*, *longitudinal data* is data about one or more subjects, measured at different points in time. The subject and the time component are captured in the data such that the effects of time and changes in the subject over time can be examined. Clinical drug testing uses panel data to observe the effects of treatments on individuals over time. You can use panel data analysis techniques to observe the effects of activity (or inactivity) in your network subjects over time.

Panel data is like a large spreadsheet where you pull out only selected rows and columns as special groups to do analysis. You have a copy of the same spreadsheet for each instance of time when the data is collected. Panel data is the type of data that you see from telemetry in networks where the same set of data is pushed at regular intervals (such as memory data). You may see panel data and cross-sectional time series data using similar analytics techniques. Both data sets are about subjects over time, but *subjects* defines the type of data, as shown in Figure 8-7. Cross-sectional time series data is different in that there may be different subjects for each of the time periods, while panel data has the same subjects for all time periods. Figure 8-7 shows what this might look like if you had knowledge of the entire population.

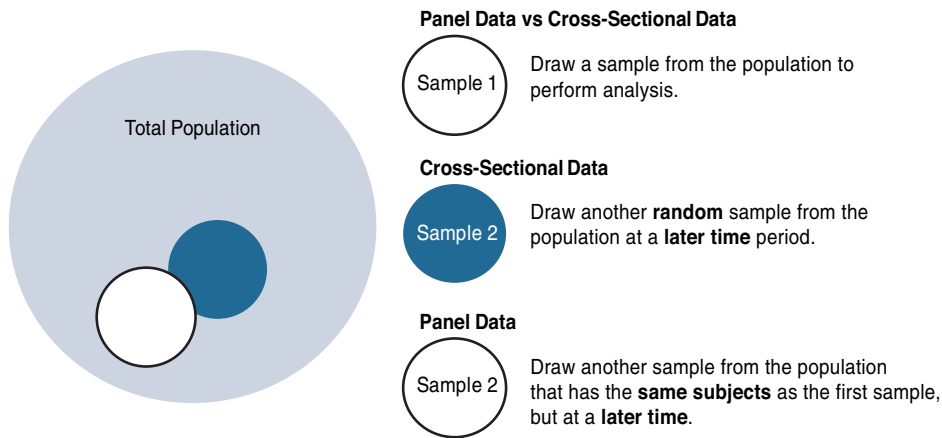


Figure 8-7 *Panel Data Versus Cross-Sectional Time Series*

Here are the things you can do with time series cross-sectional or panel data:

- Pooled regression allows you to look at the entire data set as a single population when you have the cross-sectional data that may be samples from different populations. If you are analyzing data from your ephemeral cloud instances, this comes in handy.
- Fixed effects modeling enables you to look at changes on average across the observations when you want to identify effects that are associated with the different subjects of the study.
- You can look at within-group effects and statistics for each subject.
- You can look at differences between the groups of subjects.
- You can look at variables that change over time to determine if they change the same for all subjects.
- Random effects modeling assumes that the data is not a complete analysis but just a time series cross-sectional sample from a larger population.
- Population-averaged models allow you to see effects across all your data (as opposed to subject-specific analysis).
- Mixed effects models combine some properties of random and fixed effects.

Time series is a special case of panel data where you use analysis of variance (ANOVA) methods for comparisons and insights. You can use all the statistical data mentioned previously and perform comparisons across different slices of the panel data.

ANOVA

ANOVA is a statistical technique used to measure the differences between the means of two or more groups. You can use it with panel data. ANOVA is primarily used in analyzing data sets to determine the statistically significant differences between the groups or times. It allows you to show that things behave differently as a base rate. For instance, in the memory example, the memory of certain routers and switches behaves differently for the same network loop. You can use ANOVA methods to find that these are different devices that have different memory responses to loops and, thus, should be treated differently in predictive models. ANOVA uses well-known scientific methods employing F -tests, t -tests, p -values, and null hypothesis testing.

The following are some key points about using statistics and ANOVA as you go forward into researching algorithms:

- You can use statistics for testing the significance of regression parameters, assuming that the distributions are valid for the assumptions.
- The statistics used are based on sampling theory, where you collect samples and make inferences about the rest of the populations. Analytics models are generalizations of something. You use models to predict what will happen, given some set of input values. You can see the simple correlation.
- F -tests are used to evaluate how well a statistical model fits a data set. You see F -tests in analytics models that are statistically supported.
- p -values are used in some analytics models to indicate the significance of the parameter contributing to the model. A high p -value means the variable does not support the null hypothesis (that is, that you are observing something from a different population rather than the one you are trying to model). With a low p -value, you reject that null hypothesis and assume that the variable is useful for your model.
- Mean squared error (MSE) and sum of squares error (SSE) are other common goodness-of-fit measures that are used for statistical models. You may also see RMSE, which is the square root of the MSE. You want these values to be low.
- R-squared, which is a measure of the amount of variation in the data covered by a model, ranges from zero to one. You want high R-squared values because they indicate models that fit the data well.
- For anomaly detection using statistics, you will encounter outlier terms such as *leverage* and *influence*, and you will see statistics to measure these, such as Cook's D . Outliers in statistical models can be problematic.
- Pay attention to assumptions with statistical models. Many models require that the data be IID, or *independent* (not correlated with other variables) and *identically distributed* (perhaps all normal Gaussian distributions).

Probability

Probability theory is a large part of statistical analysis. If something happens 95% of the time, then there is a 95% chance of it happening again. You derive and use probabilities in many analytics algorithms. Most predictive analytics solutions provide some likelihood of the prediction being true. This is usually a probability or some derivation of probability.

Probability is expressed as $P(X)=Y$, with Y being between zero (no chance) and one (will always happen).

The following are some key points about probability:

- The probability of something being true is the ratio of a given outcome to all possible outcomes. For example, getting heads in a coin flip has a probability of 0.5, or 50%. The simple calculation is $\text{Heads}/(\text{Heads} + \text{Tails}) = 1/(1+1)$, which is $\frac{1}{2}$, or 0.5.
- For the probability of an event A OR an event B, the probabilities are added together, as either event could happen. The probability of heads or tails on a coin flip is 100% because the 0.5 and 0.5 from heads and tails options are added together to get 1.0.
- The probability of an event followed by another event is derived through multiplication. The probability of a coin flip heads followed by another coin flip heads in order is 25%, or $0.5(\text{heads}) \times 0.5(\text{heads}) = 0.25$.
- *Statistical inference* is defined as drawing inferences from the data you have, using the learned probabilities from that data.
- Conditional probability theory takes probability to the next step, adding a prior condition that may influence the probability of something you are trying to examine. $P(A|B)$ is a conditional probability read as “the probability of A given that B has already occurred.” This could be “the probability of router crash given that memory is currently >90%.”
- Bayes’ theorem is a special case of conditional probability used throughout analytics. It is covered in the next section.

The scientific method and hypothesis testing are quite common in statistics. While formal hypothesis testing based on statistical foundations may not be used in many analytics algorithms, it has value for innovating and inverse thinking. Consider the alternative to what you are trying to show with analytics in your use case and be prepared to talk about the opposite. Using good scientific method helps you grow your skills and knowledge. If your use cases output probabilities from multiple places, you can use probability rules to combine them in a meaningful way.

Bayes’ Theorem

Bayes’ theorem is a form of conditional probability. Conditional probability is useful in analytics when you have some knowledge about a topic and want to predict the probability of some event, given your prior knowledge. As you add more knowledge, you

can make better predictions. These become inputs to other analytics algorithms. Bayes' theorem is an equation that allows you to adjust the probability of an outcome given that you have some evidence that changes the probability. For example, what is the chance that any of your routers will crash? Given no other evidence, set the probability as $\frac{\text{number of times you saw crashes in your monthly observations}}{\text{number of routers}}$.

With conditional probability you add evidence and combine that with your model predictions. What is the chance of crash this month, given that memory is at 99%? You gain new evidence by looking at the memory in the past crashes, and you can produce a more accurate prediction of crash.

Bayes' theorem uses the following principles, as shown in Figure 8-8:

- **Bayes' likelihood**—How probable is the evidence, given that your hypothesis is true? This equates to the accuracy of your test or prediction.
- **Prior**—How probable was your hypothesis before you observed the evidence? What is the historical observed rate of crashes?
- **Posterior**—How probable is your hypothesis, given the observed evidence? What is the real chance of a crash in a device you identified with your model?
- **Marginal**—How probable is the new evidence under all possible hypotheses? How many positive predictions will come from my test, both true positive predictions as well as false positives?

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal}}$$
$$P(X|Y) = \frac{P(Y|X) \times P(X)}{P(Y)}$$

Figure 8-8 *Bayes' Theorem Equation*

How does Bayes' theorem work in practice? If you look at what you know about memory crashes in your environment, perhaps you state that you have developed a model with 96% accuracy to predict possible crashes. You also know that only 2% of your routers that experience the high memory condition actually crash. So if your model predicts that a router will crash, can you say that there is a 96% chance that the router will crash? No you can't—because your model has a 4% error rate, and you need to account for that in your prediction. Bayes' theorem provides a more realistic estimate, as shown in Figure 8-9.

In this case, the likelihood is 0.96 that you will crash given your predictions and the prior is that 20 of the 1000 routers will crash, or 2%. This gives you the top of the calculation. Use all cases of correct and possibly incorrect positive predictions to calculate the marginal probability, which is 19.2 true positives and 39.2 for possible false positive predictions. This means 58.2 total positive predictions from your model, which is a probability

of .0584. Using Bayes' theorem and what you know about your own model, notice that the probability of a crash, given that your model predicted that crash, is actually only 32.9%. You and your stakeholders may be thinking that when you predict a device crash, it will occur. But the chance of that identified device crashing is actually only 1 in 3 using Bayes' theorem.

You have an **unknown** router type at 99%. What is the chance of crash?

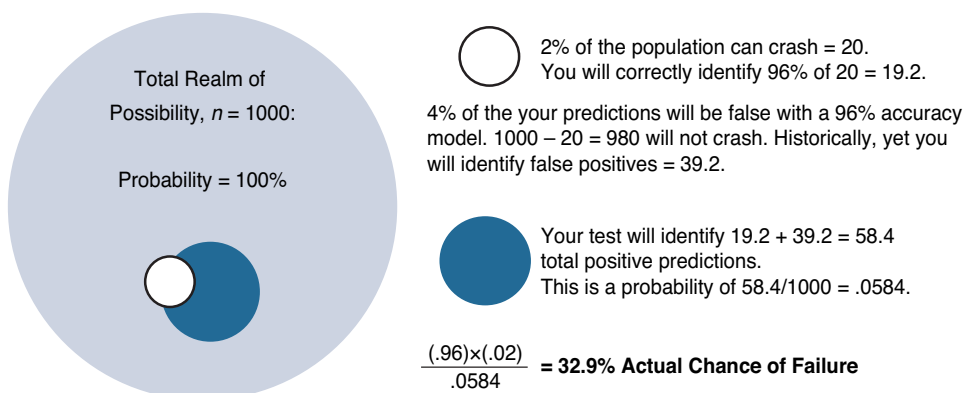


Figure 8-9 Bayes' Theorem Applied

You will see the term *Bayesian* as Bayes' theorem gets combined with many other algorithms. Bayes' theorem is about using some historical or known background information to provide a better probability. Models that use Bayesian methods guide the analysis using historical and known background information in some effective way. Bayes' theorem is heavily used in combination with classification problems, and you will find classifiers in your analytics packages such as naïve Bayes, simple Bayes, and independence Bayes. When used in classification, naïve Bayes does not require a lot of training data, and it assumes that the training data, or input features, are unrelated to each other (thus the term *naïve*). In reality, there is often some type of dependence relationship, but this can complicate classification models, so it is useful to *assume that they are unrelated* and naïvely develop a classifier.

Feature Selection

Proper feature selection is a critical area of analytics. You have a lot of data, but some of that data has no predictive power. You can use feature selection techniques to evaluate variables (variables are features) to determine their usefulness to your goal. Some variables are actually counterproductive and just increase complexity and decrease the effectiveness of your models and algorithms. For example, you have already learned that selecting features that are correlated with each other in regression models can lower the effectiveness of the models. If they are highly correlated, they state the same thing, so you are adding complexity with no benefit. Using correlated features can sometimes manifest by showing (falsely) high accuracy numbers for models. Feature selection

processes are used to identify and remove these types of issues. Garbage-in, garbage-out rules apply with analytics models. The success of your final use case is highly dependent on choosing the right features to use as inputs.

Here are some ways to do feature selection:

- If the value is the same or very close (that is, has low statistical variance) for every observation, remove it. If you are using router interfaces in your memory analysis models and you have a lot of unused interfaces with zero traffic through them, what value can they bring?
- If the variable is entirely unrelated to what you want to predict, remove it. If you include what you had for lunch each day in your router memory data, it probably doesn't add much value.
- Find filter methods that use statistical methods and correlation to identify input variables that are associated with the output variables of interest. Use analytics classification techniques. These are variables you want to keep.
- Use wrapper methods available in the algorithms. Wrapper methods are algorithms that use many sample models to validate the usefulness of actual data. The algorithms use the results of these models to see which predictors worked best.
- The forward selection process involves starting with few features and adding to the model the additional features that improve the model most. Some algorithms and packages have this capability built in.
- Backward elimination involves trying to test a model with all the available features and removing the ones that exhibit the lowest value for predictions.
- Recursive feature elimination or bidirectional elimination methods identify useful variables by repeatedly creating models and ranking the variables, ultimately using the best of the final ranked lists.
- You can use decision trees, random forests, or discriminant analysis to come up with the variable lists that are most relevant.
- You may also encounter the need to develop instrument variables or proxy variables, or you may want to examine omitted variable bias when you are doing feature selection to make sure you have the best set of features to support the type of algorithm you want to use.

Prior to using feature selection methods, or prior to and again after you try them, you may want to perform some of the following actions to see how the selection methods assess your variables. Try these techniques:

- Perform discretization of continuous numbers to integers.
- Bin numbers into buckets, such as 0–10, 11–20, and so on.
- Make transformations or offsets of numbers using mathematical functions.

- Derive your own variables from one or more of your existing variables.
- Make up new labels, tags, or number values; this process is commonly called *feature creation*.
- Use new features from dimensionality reduction such as principal component analysis (PCA) or factor analysis (FA), replacing your large list of old features.
- Try aggregation, averaging, and sampling, using mean, median, mode, or cluster centers as a binning technique.

Once you have a suitable set of features, you can prepare these features for use in analytics algorithms. This usually involves some cleanup and encoding. You may come back to this stage of the process many times to improve your work. This is all part of the 80% or more of analyst time spent on data engineering that is identified in many surveys.

Data-Encoding Methods

For categorical data (for example, small, medium, large, or black, blue, green), you often have to create a numerical representation of the values. You can use these numerical representations in models and convert things back at the end for interpretation. This allows you to use mathematical modeling techniques with categorical or textual data.

Here are some common ways to encode categorical data in your algorithms:

- Label encoding is just replacing the categorical data with a number. For example, small, medium, and large can be 1, 2, and 3. In some cases, order matters; this is called *ordinal*. In other cases, the number is just a convenient representation.
- One-hot encoding involves creating a new data set that has all categorical variables as new column headers. The categorical data entries are rows, and each of the rows uses a 1 to indicate a match to any categorical labels or a 0 to indicate a non-match.
- This one-hot method is also called the *dummy variables* approach in some packages. Some implementations create column headers for all values, which is a ones-hot method, and others leave a column out for one of each categorical class.
- For encoding documents, count encoders create a full data set, with all words as headers and documents as rows. The word counts for each document are used in the cell values.
- *Term frequency/inverse document frequency (TF/IDF)* is a document-encoding technique that provides smoothed scores for rare words over common words that may have high counts in a simple counts data set.
- Some other encoding methods include binary, sum, polynomial, backward difference, and Helmert.

The choice of encoding method you use depends on the type of algorithm you want to use. You can find examples of your candidate algorithms in practice and look at how the variables are encoded before the algorithm is actually applied. This provides some guidance and

insight about why specific encoding methods are chosen for that algorithm type. A high percentage of time spent developing solutions is getting the right data and getting the data right for the algorithms. A simple example of one-hot encoding is shown in Figure 8-10.

Term Document Matrix – One-Hot



	Term 1	Term 2	Term 3	Term 4	Term 5
	The	dog	cat	ran	home
the dog ran home	1	1	0	1	1
the dog is a dog	1	1	0	0	0
the cat	1	0	1	0	0
the cat ran home	1	0	1	1	1

Figure 8-10 *One-Hot Encoding Example*

Dimensionality Reduction

Dimensionality reduction in data science has many definitions. Some dimensionality reduction techniques are related to removing features that don't have predictive power. Other methods involve combining features and replacing them with combination variables that are derived from the existing variables in some way. For example, Cisco Services fingerprint data sets sometimes have thousands of columns and millions of rows. When you want to analyze or visualize this data, some form of dimensionality reduction is needed. For visualizing these data for human viewing, you need to reduce thousands of dimensions down to two or three (using principal component analysis [PCA]).

Assuming that you have already performed good feature selection, here are some dimensionality techniques to use for your data:

- The first thing to do is to remove any columns that are the same throughout your entire set or subset of data. These have no value.
- Correlated variables will not *all* have predictive value for prediction or classification model building. Keep one or replace entire groups of common variables with a new proxy variable. Replace the proxy with original values after you complete the modeling work.
- There are common dimensionality reduction techniques that you can use, such as PCA, shown in Figure 8-11.

PCA is a common technique used to reduce data to fewer dimensions, so that the data can be more easily visualized. For example, a good way to think of this is having to plot data points on the x- and y-axes, as opposed to plotting data points on 100 axes. Converting categorical data to feature vectors and then clustering and visualizing the results allows for a quick comparison-based analysis.

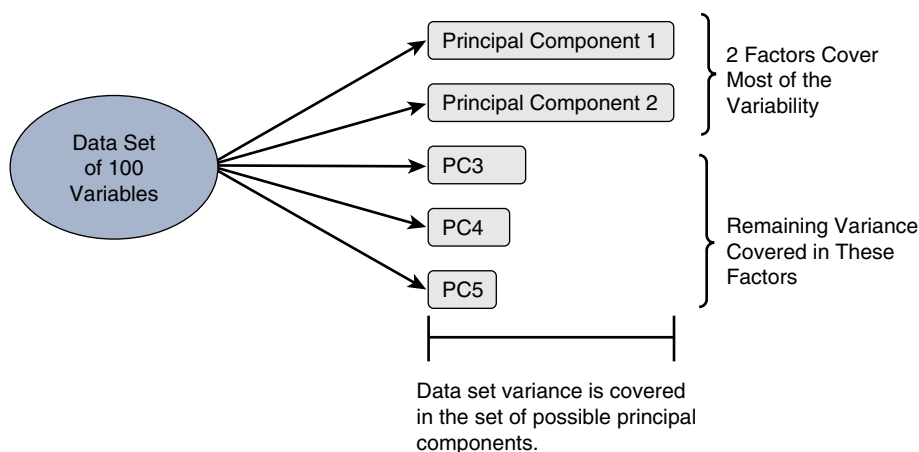


Figure 8-11 *Principal Component Analysis*

Sometimes simple unsupervised learning clustering is also used for dimensionality reduction. When you have high volumes of data, you may only be interested in the general representation of groups within your data. You can use clustering to group things together and then choose representative observations for the group, such as a cluster center, to represent clusters in other analytics models. There are many ways to reduce dimensionality, and your choice of method will depend on the final representation that you need for your data. The simple goal of dimensionality reduction is to maintain the general meaning of the data, but express it in far fewer factors.

Unsupervised Learning

Unsupervised learning algorithms allow you to explore and understand the data you have. Having an understanding of your data helps you determine how best you can use it to solve problems. *Unsupervised* means that you do not have a label for the data, or you do not have an output side to your records. Each set of features is not represented by a label of any type. You have all input features, and you want to learn something from them.

Clustering

Clustering involves using unsupervised learning to find meaningful, sometimes hidden structure in data. Clustering allows you to use data that can be in tens, hundreds, or thousands of dimensions—or more—and find meaningful groupings and hidden structures. The data can appear quite random in the data sets, as shown in Figure 8-12. You can use many different choices of distance metrics and clustering algorithms to uncover meaning.

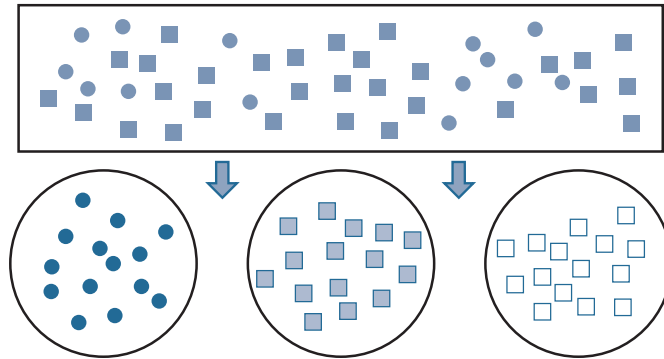
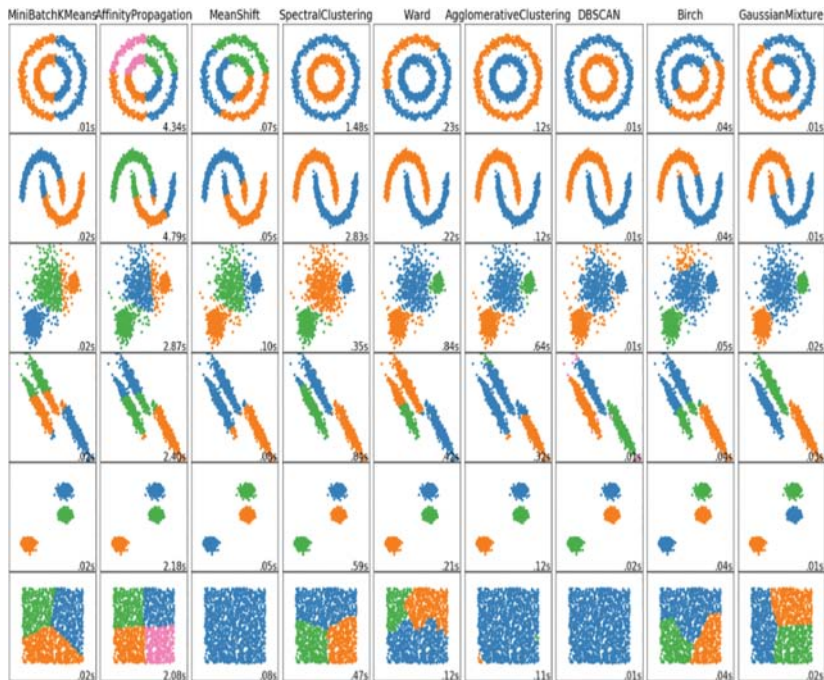


Figure 8-12 *Clustering Insights*

Clustering in practice is much more complex than the simple visualizations that you commonly see. It involves starting with very high-dimensional data and providing human-readable representations. As shown in the diagram from the Scikit-learn website in Figure 8-13, you may see many different types of distributions with your data after clustering and dimensionality reduction. Depending on the data, the transformations that you apply, and the distance metrics you use, your visual representation can vary widely.



A comparison of the clustering algorithms in Scikit-learn

Figure 8-13 *Clustering Algorithms and Distributions*

As shown in the Scikit-learn diagram, certain algorithms work best with various distributions of data. Try many clustering methods to see which one works best for your purpose. You need to do some feature engineering to put the data into the right format for clustering. Different forms of feature selection can result in non-similar cluster representations because you will have different dimensions. For clustering categorical data, you first need to represent categorical items as encoded numerical vectors, such as the one-hot, or dummy variable, encoding.

Distance functions are the heart of clustering algorithms. You can couple them with linkage functions to determine *nearness*. Every clustering algorithm must have a method to determine the *nearness* of things in order to cluster them. You may be trying to determine nearness of things that have hundreds or thousands of features. The choice of distance measure can result in widely different cluster representations, so you need to do research and some experimentation. Here are some common distance methods you will encounter:

- Euclidean distance is difference in space as the crow flies between two points. Euclidean distance is good for clustering points in n -dimensional space and is used in many clustering algorithms.
- Manhattan distance is useful in cases where there may be outliers in the data.
- Jaccard distance is the measure of proportion of the characteristics shared between things. This is useful for one-hot encoded and Boolean encoded values.
- Cosine distance is a measurement of the angle between vectors in space. When the vectors are different lengths, such as variable-length text and document clustering, cosine distance usually provides better results than Euclidean or Manhattan distance.
- Edit distance is a measure of how many edits need to be done to transform one thing into another. Edit distance is good with text analysis when things are closely related. (Recall soup and soap from Chapter 5. In this case, the edit distance is one.) Hamming distance is also a measure of differences between two strings.
- Distances based on correlation metrics such as Pearson's correlation coefficient, Spearman's rank, or Kendall's tau are used to cluster observations that are very highly correlated to each other in terms of features.

There are many more distance metrics, and each has its own nuances. The algorithms and packages you choose provide information about those nuances.

While there are many algorithms for clustering, there are two main categories of approaches:

- Hierarchical agglomerative clustering is bottom-up clustering where every point starts out in its own cluster. Clustering algorithms iteratively combine nearest clusters together until you reach the cutoff number of desired clusters. This can be memory intensive and computationally expensive.
- Divisive clustering starts with everything in a single cluster. Algorithms that use this approach then iteratively divide the groups until the desired number of clusters is reached.

Choosing the number of clusters is sometimes art and sometimes science. The number of desired clusters may not be known ahead of time. You may have to explore the data and choose numbers to try. For some algorithms, you can programmatically determine the number of clusters. Dendograms (see Figure 8-14) are useful for showing algorithms in action. A dendogram can evaluate the number of clusters in the data, given the choice of distance metric. You can use a dendogram to get insights into the number of clusters to choose.

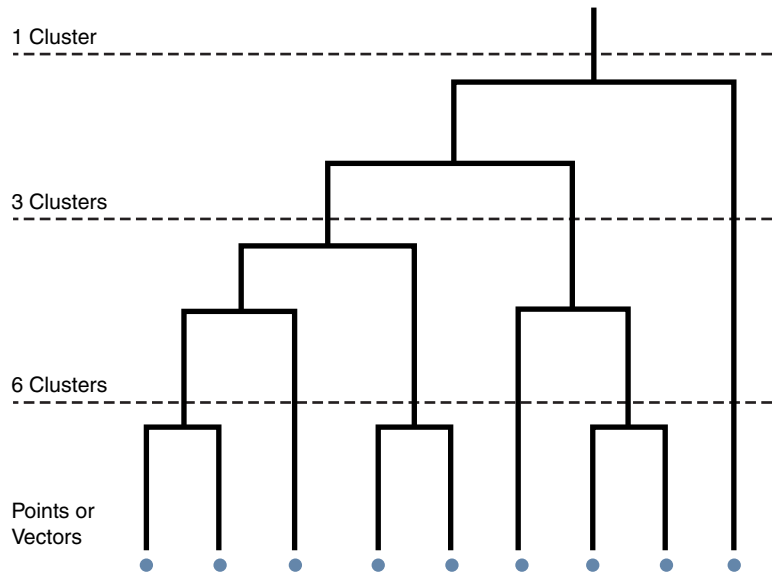


Figure 8-14 *Dendogram for Hierarchical Clustering*

You have many options for clustering algorithms. Following are some key points about common clustering algorithms. Choose the best one for your purpose:

- K-means
 - Very scalable for large data sets
 - User must choose the number of clusters
 - Cluster centers are interesting because new entries can be added to the best cluster by using the closest cluster center.
 - Works best with globular clusters
- Affinity propagation
 - Works best with globular clusters
 - User doesn't have to specify the number of clusters
 - Memory intensive for large data sets

- Mean shift clustering
 - Density-based clustering algorithm
 - Great efficiency for computer vision applications
 - Finds peaks, or centers, of mass in the underlying probability distribution and uses them for cluster centers
 - Kernel-based clustering algorithm, with the different kernels resulting in different clustering results
 - Does not assume any cluster shape
- Spectral clustering
 - Graph-theory-based clustering that clusters on nearest neighbor similarity
 - Good for identifying arbitrary cluster shapes
 - Outliers in the data can impact performance
 - User must choose the number of clusters and the scaling factor
 - Clusters continuous groups of denser items together
- Ward clustering
 - Works best with globular clusters
 - Clusters should be equal size
- Hierarchical clustering
 - Agglomerative clustering, bottom to top
 - Divisive clustering that starts with one large cluster of all and then splits
 - Scales well to large data sets
 - Does not require globular clusters
 - User must choose the number of desired clusters
 - Similar intuition to a dendrogram
- DBSCAN
 - Density-based algorithm
 - Builds clusters from dense regions of points
 - Every point does not have to be assigned to a cluster

- Does not assume globular clusters
- User must tune the parameters for optimal performance
- Birch
 - Hierarchical-based clustering algorithm
 - Builds a full dendrogram of the data set
 - Expects globular clusters
- Gaussian EM clustering and Gaussian mixture models
 - Expectation maximization method
 - Uses probability density for clustering

A case of categorical anomaly detection that you can do with clustering is configuration consistency. Given some number of IT devices that are performing exactly the same IT function, you expect them to have the same configuration. Configurations that are widely different from others in the same group or cluster are therefore anomalous. You can use textual comparisons of the data or convert the text representations to vectors and encode into a dummy variable or one-hot matrix. You can use clustering algorithms or reduce the data yourself in order to visualize the differences. Then outliers are identified using anomaly detection and visual methods, as shown in Figure 8-15.

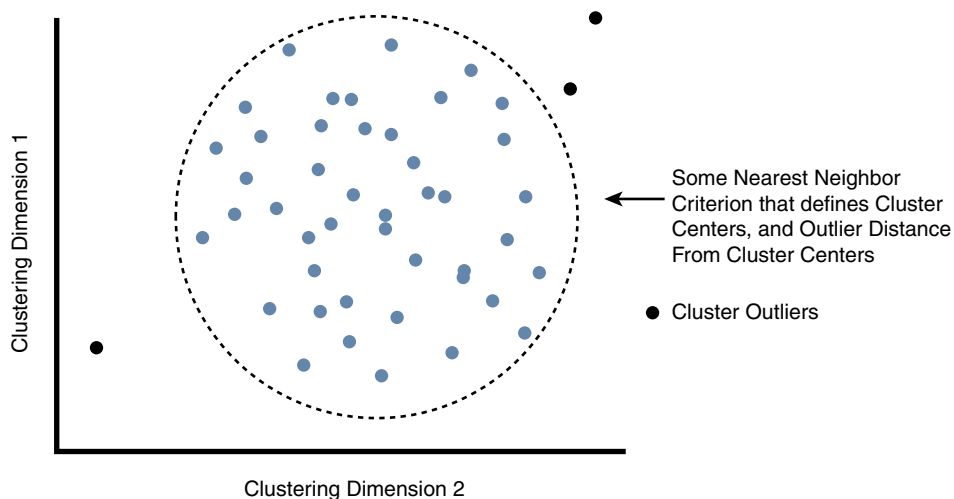


Figure 8-15 *Clustering Anomaly Detection*

This is an example of density-based anomaly detection, or clustering-based anomaly detection. This is just one of many use cases where clustering plays a foundational role. Clustering is used for many cases of exploration and solution building.

Association Rules

Association rules is an unsupervised learning technique for identifying groups of items that commonly appear together. Association rules are used in market basket analysis, where items such as milk and bread are often purchased together in a single basket at checkout. The details of association rules logic are examined in this section. For basic market basket analysis, order of the items or purchases may not matter, but in some cases it does. Understanding association rules is a necessary foundation for understanding sequential pattern mining to look at *ordered* transactions. Sequential pattern mining is an advanced form of the same logic.

To generate association rules, you collect and analyze transactions, as shown in Figure 8-16, to build your data set of things that were seen together in transactions.

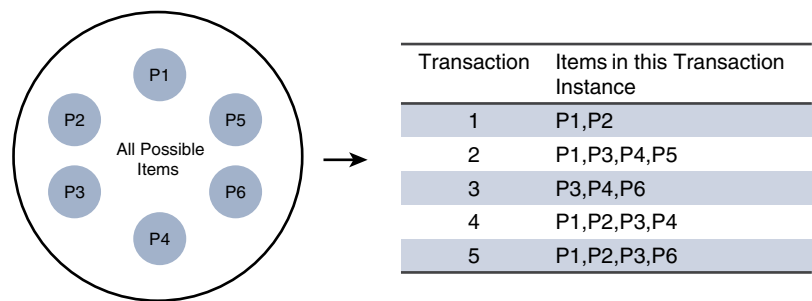


Figure 8-16 Capturing Grouped Transactions

You can think of transactions as groups of items and use this functionality in many contexts. The items in Figure 8-16 could be grocery items, configuration items, or patterns of any features from your domain of expertise. Let’s walk through the process of generating association rules to look at what you can do with these sets of items:

- You can identify frequent item sets of any size with all given transactions, such as milk and bread in the same shopping basket. These are frequent patterns of co-occurrence.
- Infrequent item sets are not interesting for market basket cases but may be interesting if you have some analysis looking for anti-patterns. There is not a lot of value in knowing that 1 person in 10,000 bought milk and ant traps together.
- Assuming that frequent sets are what you want, most algorithms start with all pairwise combinations and scan the data set for the number of times that is seen. Then you examine each triple combination, and then each quadruple combination, up to the highest number in which you have interest. This can be computationally expensive; also, longer, unique item sets occur less frequently.

- You can often set the minimum and maximum size parameters for item set sizes that are most interesting in the algorithms.
- Association rules are provided in the format $X \rightarrow Y$, where X and Y are individual items or item sets that are mutually exclusive (that is, X and Y are different individual items or sets with no common members between them).

Once this data evaluation is done, a number of steps are taken to evaluate interesting rules. First, you calculate the support of each of the item sets, as shown in Figure 8-17, to eliminate infrequent sets. You must evaluate all possible combinations at this step.

Transaction	Items		Set	Support Count	Support
3	P1,P2	→	{P1,P2}	3	3/5 = .6
2	P1,P3,P4,P5		{P4,P6}	1	1/5 = .2
3	P3,P4,P6		{P3,P4}	3	3/5 = .6
4	P1,P2,P3,P4		P5	1	1/5 = .2
5	P1,P2,P3,P6		{P1,P3}	3	3/5 = .6

Figure 8-17 *Evaluating Grouped Transactions*

Support value is the number of times you saw the set across the transactions. In this example, it is obvious that P5 has low counts everywhere, so you can eliminate this in your algorithms to decrease dimensionality if you are looking for frequent occurrences only. Most association rules algorithms have built-in mechanisms to do this for you. You use the remaining support values to calculate the confidence that you will see things together for defining associations, as shown in Figure 8-18.

Transaction	Items		Association Rule $X \rightarrow Y$	Count X	Confidence $X \text{ union } Y / \text{Count } X$
1	P1,P2	→	$P1 \rightarrow P2$	4 P1s	3/4 = .75
2	P1,P3,P4,P5		$P2 \rightarrow P3$	3 P2s	2/3 = .67
3	P3,P4,P6		$P3 \rightarrow P4$	4 P3s	3/4 = .75
4	P1,P2,P3,P4		$P4 \rightarrow P5$	3 P4s	1/3 = .33
5	P1,P2,P3,P6		$\{P1,P2\} \rightarrow P5$	3 × {P1,P2}	0/3 = .0

Figure 8-18 *Creating Association Rules*

Notice in the last entry in Figure 8-18 that you can use sets on either side of the association rules. Also note from this last set that these never appear together in a transaction, so you can eliminate them from your calculations early in your workflow. Lift, shown in Figure 8-19, is a measure to help determine the value of a rule. Higher lift values indicate rules that are more interesting. The lift value of row 4 shows as higher because P5 only appears with P4. But P5 is rare and is not interesting in the first place, so if it were removed, it would not cause any falsely high lift values.

Transaction	Items	Association Rule X → Y	Confidence	Expected Confidence of Y Conf/Support(Y)	Lift
1	P1,P2	P1 → P2	3/4 = .75	.75/.6	1.25
2	P1,P3,P4,P5	P2 → P3	2/3 = .67	.67/.6	1.12
3	P3,P4,P6	P3 → P4	3/4 = .75	.75/.6	1.25
4	P1,P2,P3,P4	P4 → P5	1/3 = .33	.33/.2	1.65
5	P1,P2,P3,P6	{P1,P2} → P5	0/3 = .0	.0/.2	0

Figure 8-19 *Qualifying Association Rules*

You now have sets of items that often appear together, with statistical measures to indicate *how often* they appear together. You can use these rules for prediction when you know that you have some portion of sets in your *baskets* of features. If you have three of four items that always go together, you may also want the fourth. You can also use the generated sets for other solutions where you want to understand common groups of data, such as recommender engines, customer churn, and fraud cases.

There are various algorithms available for association rules, each with its own nuances. Some of them are covered here:

- Apriori
 - Calculates the item sets for you.
 - Has a downward closure property to minimize calculations. A downward closure property simply states that if an item set is frequent, then all subcomponents are frequent. For example, you know that {P1,P2} is frequent, and therefore P1 and P2 individually are frequent.
 - Conversely, if individual items are infrequent, larger sets containing that item are not frequent either.
 - Apriori eliminates infrequent item sets by using a configurable support metric (refer to Figure 8-17).
- FP growth
 - Does not generate all candidate item sets up front and therefore is less computationally intensive than apriori.
 - Passes over the data set and eliminates low-support items before generating item sets.
 - Sorts the most frequent items for item set generation.
 - Builds a tree structure using the most common items at the root and extracts the item sets from the tree.
 - This tree can consume memory and may not fit into memory space.

Other algorithms and variations can be used for generating association rules, but these two are the most well-known and should get you started.

A few final notes about association rules:

- Just because things appear together does not mean they are related. Correlation is not causation. You still need to put on your SME hat and validate your findings before you use the outputs for use cases that you are building.
- As shown in the lift calculations, you can get results that are not useful if you do not tune and trim the data and transactions during the early phases of transaction and rule generation.
- Be careful in item selection because the possible permutations and combinations can get quite large, with a large number of possible items. This can exponentially increase computational load and memory requirements for running the algorithms.

Note that much of this section described a process, and some analytics algorithms were used as needed. This is how you will build analysis that you can improve over time. For example, in the next section, you will see how to take the process and algorithms from this section and use them differently to gain additional insight.

Sequential Pattern Mining

When the order of transactions matters, association rules analysis evolves to a method called sequential pattern mining. With sequential pattern mining you use the same type of process as with association rules but with some enhancements:

- Items and item sets are now mini-transactions, and they are in order. Two items in association rules analysis produce a single set. In sequential transaction analysis, the two items could produce two sets if they were seen in different sequences in the data. {Bread,Milk} becomes {Bread & Milk}, which is different from {Milk & Bread} as a sequential pattern. You can sit at your desk and then take a drink, or you can take a drink and then sit at your desk. These are different transactions for sequential pattern mining.
- Just as with association rules, individual items and item sequences are gathered for evaluation of support. You can still use the apriori algorithm to identify rare items and sets in order to remove rare sequences that contain them. Smaller items or sequences can be subsets of larger sequences.
- Because transactions can occur over time, the data is bounded by a time window. A sliding window mechanism is used to ensure that many possible start/stop time windows are considered. Computer-based transactions in IT may have windows of hours or minutes, while human purchases may span days, months, or years.
- Association rules simply look at the baskets of items. Sequential pattern mining requires awareness of the subjects responsible for the transactions so that transactions related to the same subject within the same time windows can be assembled.

- There are additional algorithms available for sequential mining in addition to the apriori and FPgrowth approaches, such as generalized sequential pattern (GSP), sequential pattern discovery using equivalence class (SPADE), FreeSpan, and PrefixSpan.
- Episode mining is performed on the items and sequences to find serial episodes, parallel episodes, relative order, or any combination of the patterns in sequences. Regular expressions allow for identifying partial sequences with or without constraints and dependencies.

Episode mining is the key to sequential pattern mining. You need to identify small sequences of interest to find instances of larger sequences that contain them or identify instances of the larger sequences. You want to identify sequences that have most, but not all, of the subsequences or look for patterns that end in subsequences of interest, such as a web purchase after a sequence of clicks through the site. There are many places to go from here in using your patterns:

- Identify and monitor your ongoing patterns for patterns of interest. Cisco Network Early Warning systems look for early subsequences of patterns that result in undesirable end sequences.
- Use statistical methods to identify the commonality of patterns and correlate those pattern occurrences to other events in your environment.
- Identify and whitelist frequent patterns associated with normal behavior to remove noise from your data. Then you have a dimension-reduced data set to take forward for more targeted analysis.
- Use sequential pattern mining anywhere you like to predict probability of specific ends of transactions based on the sequences at the beginning.
- Identify and rank all transactions by commonality to recognize rare and new transactions using your previous work.
- Identify and use partial pattern matches as possible incomplete transactions (some incomplete transactions could be DDoS attacks, where transaction sessions are opened but not closed.).

These are just a few broad cases for using the patterns from sequential pattern mining. Many of the use cases in Chapter 7 have sequenced transaction and time-based components that you can build using sequential pattern mining.

Collaborative Filtering

Collaborative filtering and recommendations systems algorithms use correlation, clustering, supervised learning classification, and many other analytics techniques. The algorithm choices are domain specific and related to the relationships you can identify. Consider the simplified diagram in Figure 8-20, which shows the varying complexity levels you can choose for setting up your collaborative filtering groups. In this example, you can look at possible

purchases by an individual and progressively segment until you get to the granularity that you want. You can identify a cluster of users and the clusters of items that are most correlated.

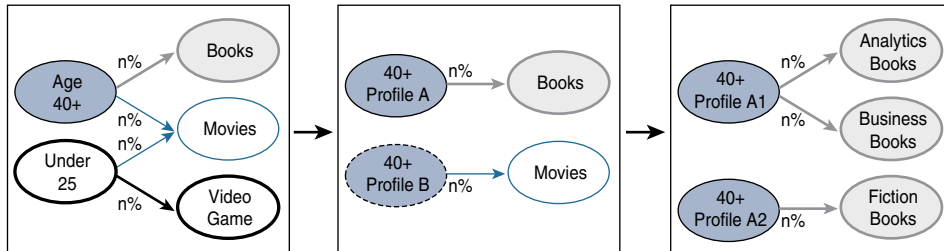


Figure 8-20 *Identifying User and Item Groups to Build Collaborative Filters*

Note that you can choose how granular your groups may be, and you can use both supervised and unsupervised machine learning to further segment into the domains of interest. If your groups are well formed, you can make recommendations. For example, if a user in profile A1 buys an analytics book, he or she is probably interested in other analytics books purchased by similar users. You can use the same types of insights for network configuration analysis, as shown in Figure 8-21, segmenting out routers and router configuration items.

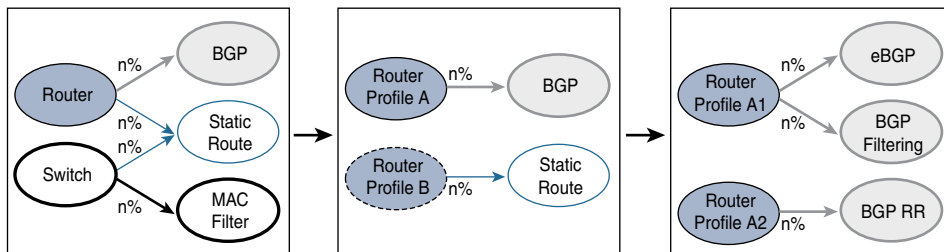


Figure 8-21 *Identifying Router and Technology Groups to Build Collaborative Filters*

Collaborative filtering solutions have multiple steps. Here is a simplified flow:

1. Use clustering to cluster users, items, or transactions to analyze individually or in relationship to each other.
 - a. User-based collaborative filtering infers that you are similar to other users in some way, so you will like what they like. This is others in the same cluster.
 - b. Item-based collaborative filtering is identifying items that appear together in frequent transactions, as found by association rules analysis.
 - c. Transaction-based collaborative filtering is identifying sets of transactions that appear together, in sequence or clusters.

2. Use correlation techniques to find the nearness of the groups of users to groups of items.
3. Use market basket and sequential pattern matching techniques to identify transactions that show matches of user groups to item groups.

Recommender systems can get quite complex, and they are increasing in complexity and effectiveness every day. You can find very detailed published work to get you started on building your own system using a collection of algorithms that you choose.

Supervised Learning

You use supervised learning techniques when you have a set of features and a label for some output of interest for that set of features. Supervised learning includes classification for discrete or categorical machine learning and regression techniques to use when the output is a continuous number value.

Regression Analysis

Regression is used for modeling and predicting continuous, numerical variables. You can use regression analysis to confirm a mathematical relationship between inputs and outputs—for example, to predict house or car prices or prices of gadgets that contain features that you want, as shown in Figure 8-22. Using the regression line, you can predict that your gadget will cost about \$120 with 12 features or \$200 with 20 features.

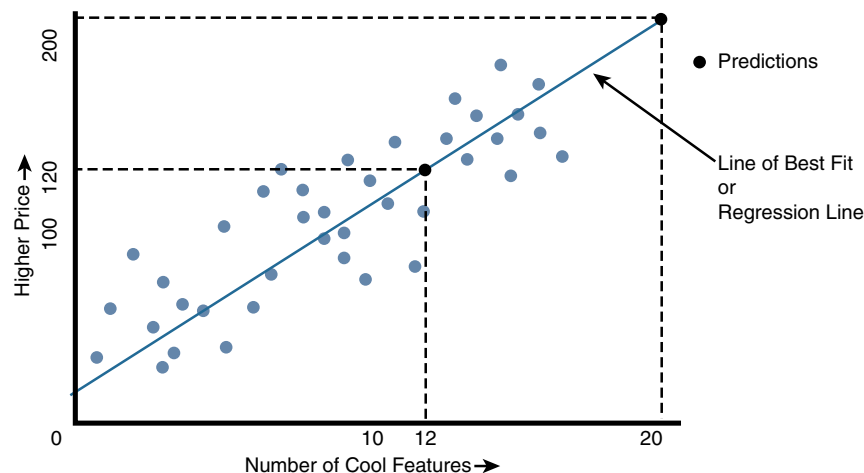


Figure 8-22 *Linear Regression Line*

Regression is also very valuable for predicting outputs that become inputs to other models. Regression is about estimating the relationship between two or more variables. Regression intuition is simply looking at an equation of a set of independent variables and a dependent variable in order to determine the impacts of independent variable changes on the dependent variable.

The following are some key points about linear regression:

- Linear regression is a best-fit straight line that is used for looking for linear relationships between the predictors and continuous or discrete output numbers.
- You can use both sides of regression equations for value. First, if you are interested in seeing how much impact an input has on the dependent variable, the coefficients of the input variables in regression models can tell you that. This is model explainability.
- Given the simplistic regression equation $x + 2y = z$, you can easily see that changes in value x will have about half the impact of changes in y on the output z .
- You can use the output side of the equation for prediction by using different numbers with the input variables to see what your predicted price would be. There are other considerations, such as error terms and graph intercept, for you to understand; you can learn about them from your modeling software.
- Linear regression performs poorly if there are nonlinear relationships.
- You need to pay attention to assumptions in regression models. You can use linear regression very easily if you have met assumptions. Common assumptions are the assumption of linearity of the predicted value and having predictors that are continuous number values.

Many algorithms contain some form of regression and are more complex than simple linear regression. The following are some common ones:

- Logistic regression is not actually regression but instead a classifier that predicts the probability of an outcome, given the relationships among the predictor variables as a set.
- Polynomial regression is used in place of linear regression if a relationship is found to be nonlinear and a curved-line model is needed.
- Stepwise regression is an automated wrapper method for feature selection to use for regression models. Stepwise regression adds and removes predictors by using forward selection, backward elimination, or bidirectional elimination methods.
- Ridge regression is a linear regression technique to use if you have collinearity in the independent variable space. Recall that collinearity is correlation in the predictor space.
- Lasso regression *lassos* groups of correlated predictor variables into a single predictor.
- ElasticNet regression is a hybrid of lasso and ridge regression.

Regression usually provides a quantitative prediction of how much (for example, housing prices). Classification and regression are both supervised learning, but they differ in that classification predicts a yes or no, sometimes with added probability.

Classification Algorithms

Classification algorithms learn to classify instances from a training data set. The resulting classification model is used to classify new instances based on that training. If you saw a man and woman walking toward you, and you were asked to classify them, how would you do it? A man and woman? What if a dog is also walking with them, and you are asked you to classify again? People and animals? You don't know until you are trained to provide the proper classification.

You train models with labeled data to understand the dimensions to use for classification. If you have input parameters collected, cleaned, and labeled for sets of known parameters, you can choose among many algorithms to do the work for you. The idea behind classification is to take the provided attributes and identify things as part of a known class. As you saw earlier in this chapter, you can cluster the same data in a wide array of possible ways. Classification algorithms also have a wide variety of options to choose from, depending on your requirements.

The following are some considerations for classification:

- Classification can be binomial (two class) or multi-class. Do you just need a yes/no classification, or do you have to classify more, for example man, woman, dog, or cat?
- The boundary for classification may be linear or nonlinear. (Recall the clustering diagram from Scikit-learn, shown in Figure 8-13.)
- The number of input variables may dictate your choice of classification algorithms.
- The number of observations in the training set may also dictate algorithm choice.
- The accuracy may differ depending on the preceding factors, so plan to try out a few different methods and evaluate the results using contingency tables, described later in this chapter.

Logistic regression is a popular type of regression for classification. A quick examination of the properties is provided here to give you insight into the evaluation process to use for choosing algorithms for your classification solutions.

- Logistic regression is used for probability of classification of a categorical output variable.
- Logistic regression is a linear classifier. The output depends on the sum or difference of the input parameters.
- You can have two-class or multiclass (one versus all) outputs.
- It is easy to interpret the model parameters or the coefficients on the model to see the high-impact predictors.
- Logistic regression can have categorical and numerical input parameters. Numerical predictors are continuous or discrete.

- Logistic regression does not work well with nonlinear decision boundaries.
- Logistic regression uses maximum likelihood estimation, which is based on probability.
- There are no assumptions of normality in the variables.
- Logistic regression requires a large data set for training.
- Outliers can be problematic, so the training data needs to be good.
- Log transformations are used to interpret, so there may be transformations required on the model outputs to make them more user friendly.

You can use the same type of process for evaluating any algorithms that you want to use. A few more classifiers are examined in the following sections to provide you with insight into some key methods used for these algorithms.

Decision Trees

Decision trees partition the set of input variables based on the finite set of known values within the input set. Classification trees are commonly used when the variables are categorical and unordered. Regression trees are used when the variables are discretely ordered or continuous numbers.

Decision trees are built top down from a root node, and the features from the training data become decision nodes. The classification targets are leaf nodes in the decision tree. Figure 8-23 shows a simple example of building a classifier for the router memory example. You can use this type of classifier to predict future crashes.

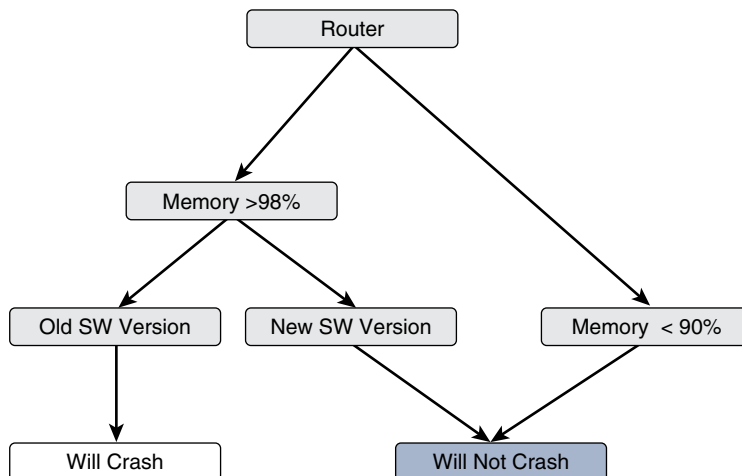


Figure 8-23 Simple Decision Tree Example

The main algorithm used for decision trees is called ID3, and it works on a principal of entropy and information gain. Entropy, by definition, is chaos, disorder, or unpredictability. A decision tree is built by calculating an entropy value for each decision node as you work top to bottom and choosing splits based on the most information gain. Information gain is defined as the best decrease in entropy as you move closer to the bottom of the tree. When entropy is zero at any node, it becomes a leaf node. The entire data set can be evaluated, and many classes, or leaves, can be identified.

Consider the following additional information about decision trees and their uses:

- Decision trees can produce a classification alone or a classification with a probability value. This probability value is useful to carry onward to next level of analysis.
- Continuous values may have to be binned to reduce the number of decision nodes. For example, you could have binned memory in 1% or 10% increments.
- Decision trees are prone to overfitting. You can perfectly characterize a data set with a decision tree. Tree pruning is necessary to have a usable model.
- Root node selection can be biased toward features that have a large number of values over features that have a small number of values. You can use gain ratios to address this.
- You need to have data in all the features. You should remove empty or missing data from the training set or estimate it in some way. See Chapter 4, “Accessing Data from Network Components,” for some methods to use for filling missing data.
- C4.5, CART, RPART, C5.0, CHAID, QUEST, and CRUISE are alternative algorithms with enhancements for improving decision tree performance.

You may choose to build rules from the decision tree, such as *Router with memory greater than 98% and old software version WILL crash*. Then you can use the findings from your decision trees in your expert systems.

Random Forest

Random forest is an ensemble method for classification or regression. Ensemble methods in analytics work on the theory that multiple weak learners can be run on the same data set, using different groups of the variable space, and each learner gets a vote toward the final solution. The idea of ensemble models is that this *wisdom of the crowd* method of using a collection of weak learners to form a *group-based strong learner* produces better results. In random forest, hundreds or thousands of decision tree models are used, and different features are chosen at random for each, as shown in Figure 8-24.

Random forest works on the principle of bootstrap aggregating, or *bagging*. Bagging is the process of using a bunch of independent predictors and combining the weighted outputs into a final vote.

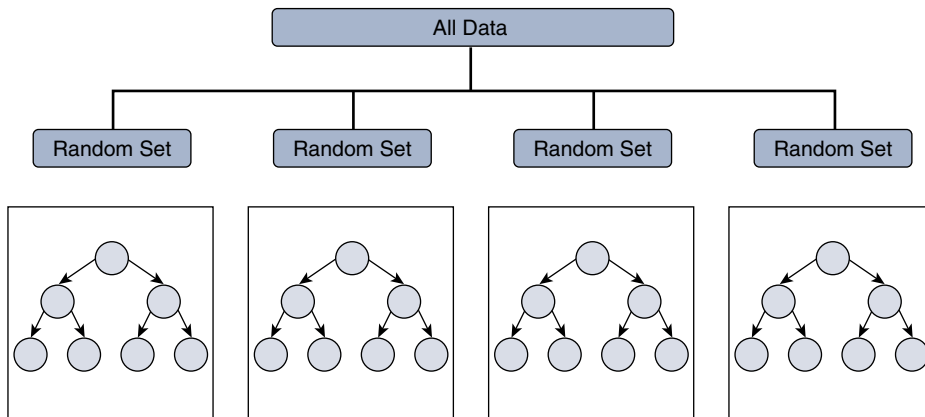


Figure 8-24 *A Collection of Decision Trees in Random Forest*

This type of ensemble works in the following way:

1. Random features are chosen from the underlying data, and many trees are built using the random sets. This could result in many different root nodes as features are left out of the random sets.
2. Each individual tree model in the ensemble is built independently and in parallel.
3. Simple voting is performed, and each classifier votes to obtain a final outcome.

Bagging is an important concept that you will see again. The following are a few key points about the purpose of bagging:

- The goal is to decrease the variance in the data to get a better-performing model.
- Bagging uses a parallel ensemble, with all models built independently and with replacement in a data set. “With replacement” means that you copy out a random part of the data instead of removing it from the set. Many parallel models can have similar randomly chosen data.
- Bagging is good for high-variance, low-bias models and is associated with overfitting.

Random forest is also useful for simple feature selection tasks when you need to find feature importance from the data set for use in other algorithms.

Gradient Boosting Methods

Gradient boosting is another ensemble method that uses multiple weaker algorithms to create a more powerful, more accurate algorithm. As you just learned, bagging models are independent learners, as used in random forest. Boosting is an ensemble method that involves making new predictors sequentially, based on the output of the previous model

step. Subsequent predictors learn from the misclassifications of the previous predictors, reducing the error each time a new predictor is created. The boosting predictors do not have to be the same type, as in bagging. Predictor models are decision trees, regression models, or other classifiers that add to the accuracy of the model.

There are several gradient-boosting algorithms, such as AdaBoost, XGBoost, and LightGBM. You could also use boosting intuition to build your own boosted methods.

Boosting has several other advantages:

- The goal of boosting is to increase the predictive capability by decreasing bias instead of variance.
- Original data is split into subsets, and new subsets are made from previously misclassified items (not random, as with bagging).
- Boosting is realized through sequential addition of new models to the ensemble by adding models where previous models lacked.
- Outputs of smaller models are aggregated and boosted using a function, such as simple voting, or weighting combined with voting.

Boosting and bagging of models are interesting concepts, and you should spend some time researching these topics. If you do not have massive amounts of training data, you will need to rely on boosting and bagging for classification. If you do have massive amounts of training data examples, then you can use neural networks for classification.

Neural Networks

With the rise in availability of computing resources and data, neural networks are now some of the most common algorithms used for classification and prediction of multiclass problems. Neural network algorithms, which were inspired by the human brain, allow for large, complex patterns of inputs to be used all at once. Image and speech recognition are two of the most popular use cases for neural networks. You often see simple diagrams like Figure 8-25 used to represent neural networks, where some number of inputs are passed through hidden layer nodes (known as *perceptrons*) that pass their outputs (that is, votes toward a particular output) on to the next layer.

So how do neural networks work? If you think of each layer as voting, then you can see the ensemble nature of neural networks as many different perspectives are passed through the network of nodes. Figure 8-25 shows a feed-forward neural network. In feed-forward neural networks, mathematical operations are performed at each node as the results are fed in a single direction through the network. During model training, weights and biases are generated to influence the math at each node, as shown in Figure 8-26. The weights and biases are aggregated with the inputs, and some activation function determines the final output to the next layer.

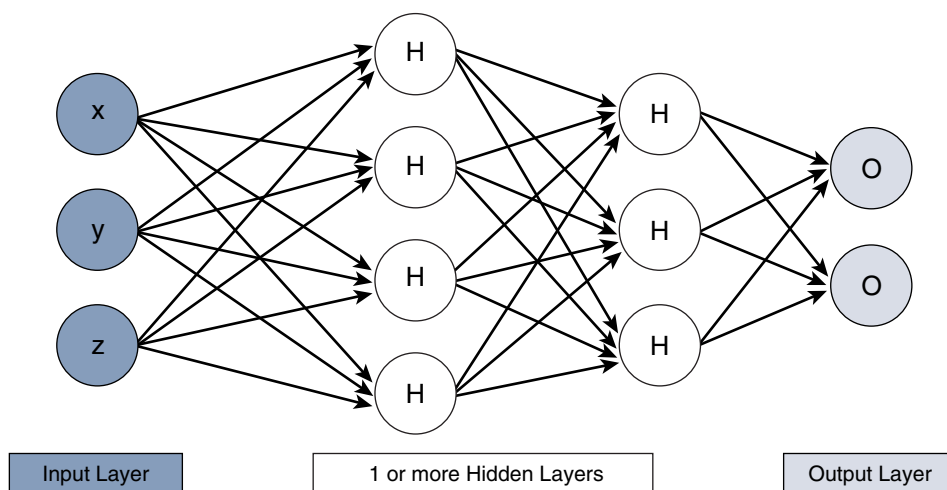


Figure 8-25 *Neural Networks Insights*

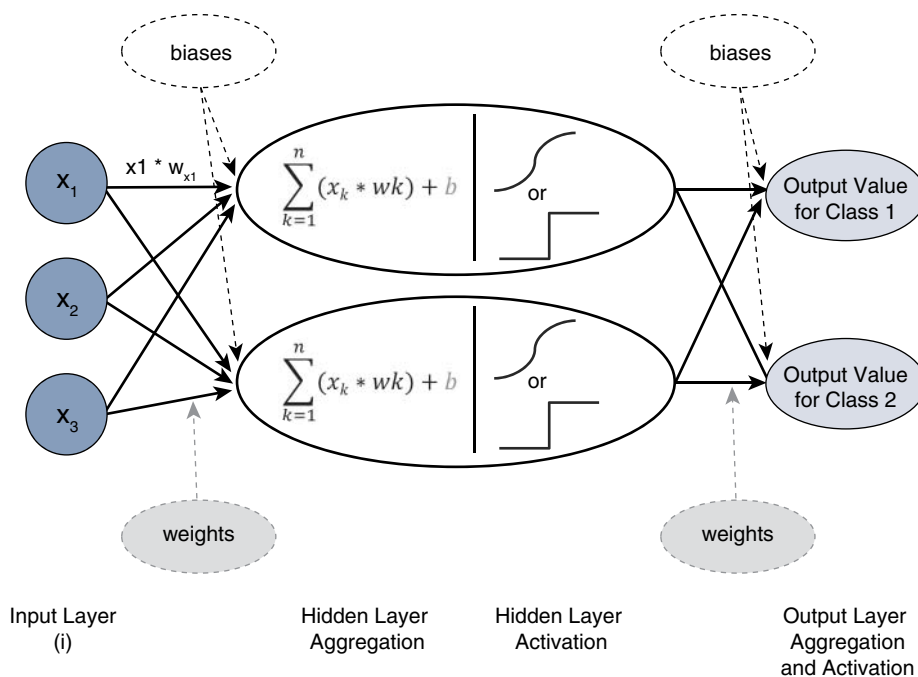


Figure 8-26 *Node-Level Activity of a Neural Network*

Using a process called *back-propagation*, the network performs backward passes using the error function observed from the network predictions to update the weights and biases to apply to every node in the network; this continues until the error in predicting the training set is minimized. The weights and biases are applied at the levels of the network, as shown in Figure 8-27 (which shows just a few nodes of the full network).

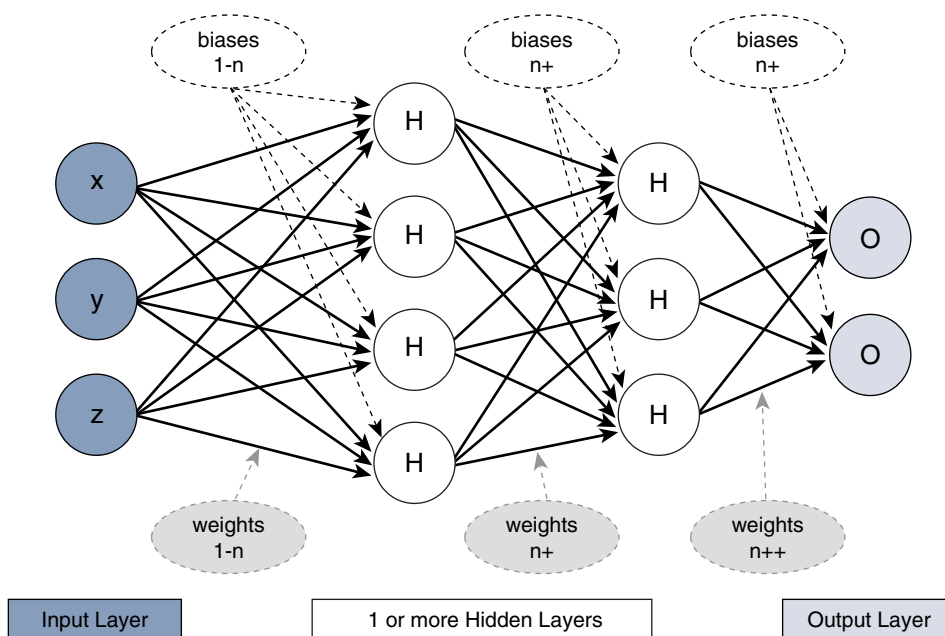


Figure 8-27 *Weights and Biases of a Neural Network*

Each of the nodes in the neural network has a method for aggregating the inputs and providing output to the next layer, and some neural networks get quite large. The large-scale calculation requirements are one reason for the resurgence and retrofitting of neural networks to many use cases today. Compute power is readily available to run some very large networks. Neural networks can be quite complex, with mathematical calculations numbering in the millions or billions.

The large-scale calculation requirement increases complexity of the network and, therefore, makes neural networks black boxes when trying to examine the predictor space for inference purposes. Networks can have many hidden layers, with different numbers of nodes per layer.

There are several types of neural network uses: artificial neural networks (ANNs) are the foundational general purpose algorithm, and are expanded upon for uses such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and very advanced Long Short Term Memory (LSTM) networks. A few key points and use cases for each are discussed next.

The following are some key points to know about artificial neural networks (ANNs):

- One hidden layer is often enough, but more complex tasks such as image recognition often use many more.
- Within a layer, the number of nodes chosen can be tricky. With too few, you can't learn, and with too many, you can be overfitting or not generalizing the process enough to use on new data.
- ANNs generally require a lot of training data. Different types of neural networks may require more or less data.
- ANNs uncover and predict nonlinear relationships between the inputs and the outputs.
- ANNs are thinned using a process called *dropout*. Dropout, which involves randomly dropping nodes and their connections from the network layers, is used to reduce overfitting.

Neural networks have evolved over the years for different purposes. CNNs, for example, involve a convolution process to add a feature mapping function early in the network that is designed to work well for image recognition. Figure 8-28 shows an example. Only one layer of convolution and pooling is shown in Figure 8-28, but multiple layers are commonly used.

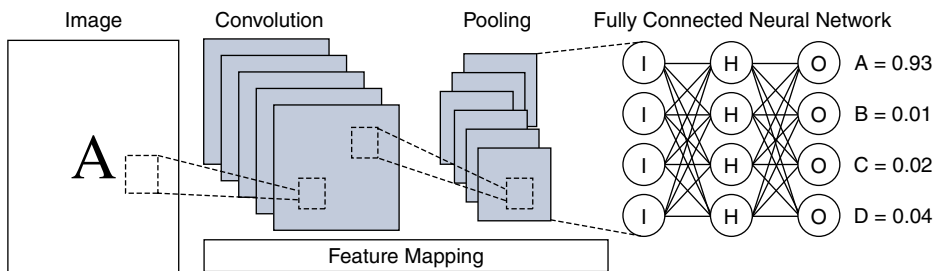


Figure 8-28 *Convolutional Neural Networks*

CNNs are primarily used for audio and image recognition, require a lot of training data, and have heavy computational requirements to do all the convolution. GPUs (graphics processing units) are commonly used for CNNs, which can be much more complex than the simple diagram in Figure 8-28 indicates. CNNs use filters and smaller portions of the data to perform an ensemble method of analysis. Individual layers of the network examine different parts of the image to generate a vote toward the final output. CNNs are not good for unordered data.

Another class of neural networks, RNNs, are used for applications that examine sequences of data, where some knowledge of the prior item in the sequence is required to examine the current inputs. As shown in Figure 8-29, an RNN is a single neural network with a feedback loop. As new inputs are received, the internal state from the previous time is

combined with the input, the internal state is updated again, and an output from that stage is produced. This process is repeated continuously as long as there is input.

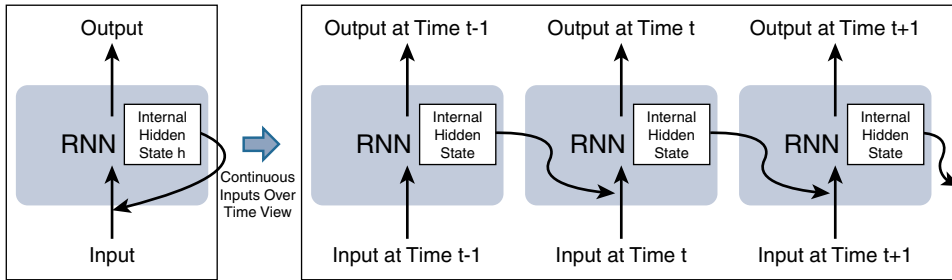


Figure 8-29 *Recurrent Neural Networks with Memory State*

Consider the following additional points about RNNs:

- RNNs are used for fixed or variably sized data where sequence matters.
- Variable-length inputs and outputs make RNN very flexible. Image captioning is a primary use case.
- Sentiment output from sentence input is another example of input lengths that may not match output length.
- RNNs are commonly used for language translation.

LSTM networks are an advanced use of neural networks. LSTMs are foundational for artificial intelligence, which often employs them in a technique called *reinforcement learning*. Reinforcement learning algorithms decide the next best action, based on the current state, using a reward function that is maximized based on possible choices. Reinforcement learning algorithms are a special case of RNNs. LSTM is necessary because the algorithm requires knowledge of specific information from past states (sometimes a long time in the past) in order to make a decision about what to do given the historical state combined with the current set of inputs.

Reinforcement learning algorithms continuously run, and state is carried through the system. As shown in Figure 8-30, the state vector is instructed at each layer about what to forget, what to update in the state, and how to filter the output for the next iteration. There is both a cell state for long-term memory and the hidden internal state similar to RNNs.

The functions and combinations with the previous input, cell state, hidden state, and new inputs are much more complex than this simple diagram illustrates, but Figure 8-30 provides you with the intuition and purpose of the LSTM mechanism. Some data are used to update local state, some are used to update long-term state, and some are forgotten when no longer needed. This makes the LSTM method extremely flexible and powerful.

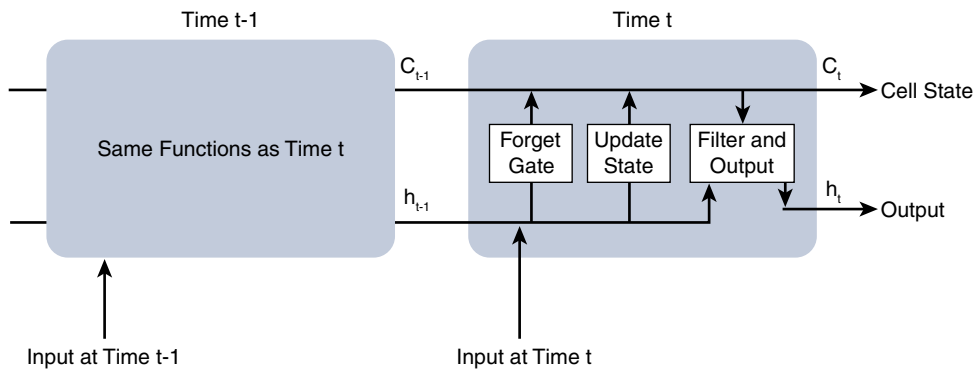


Figure 8-30 *Long Short-Term Memory Neural Networks*

The following are a few key points to know about LSTM and reinforcement learning:

- Reinforcement learning operates in a trial-and-error paradigm to learn the environment. The goal is to optimize a reward function over the entire chain.
- Decisions made now can result in a good or bad reward many steps later. You may only retrospectively get feedback. This feedback delay is why the long-term memory capability is required.
- Sequential data and time matters for reinforcement learning. Reinforcement learning has no value for unordered inputs.
- Reinforcement learning influences its own environment through the output decisions it makes while trying to maximize the reward function.
- Reinforcement learning is used to maximize the cumulative reward over the long term. Short-term rewards can be higher and misleading and may not be the right actions to maximize the long-term reward. Actions may have long-term consequences.
- An example of a long-term reward is using reinforcement learning to maximize point scores for game playing.
- Reinforcement learning history puts together many sets of observations, actions, and rewards in a timeline.
- Reinforcement learning may not know the state of the environment and must learn it through its own actions.
- Reinforcement learning does know its own state, so it uses its own state with what it has learned so far to choose the next action.
- Reinforcement learning may have a policy function to define behavior, which it uses to choose its actions. The policy is a map of states to actions.

- Reinforcement learning may have value functions, which are predictions of expected future rewards for taking an action.
- A reinforcement learning representation of the environment may be policy based, value based, or model based. Reinforcement learning can combine them and use all of them, if available.
- The balance of exploration and exploitation is a known problem that is hard to solve. Should with reinforcement learning learn the environment or always maximize reward?

This very short summary of reinforcement learning is enough to show that it is a complex topic. The good news is that packages abstract most of the complexity away for you, allowing you to focus on defining the model hyperparameters that best solve your problem. If you are going to move into artificial intelligence analytics, you will see plenty of reinforcement learning and will need to do some further research.

Neural networks of any type are optimized by tuning hyperparameters. Performance, convergence, and accuracy can all be impacted by the choices of hyperparameters. You can use automated testing to run through sets of various parameters when you are building your models in order to find the optimal parameters to use for deployment. There could be thousands of combinations of hyperparameters, so automated testing is necessary.

Neural networks take on the traditional task of feature engineering. Carefully engineered features in other model-building techniques are fed to a neural network, and the network determines which ones are important. It takes a lot of data to do this, so it is not always feasible. Don't quit your feature selection and engineering day job just yet.

Deep learning is a process of replacing a collection of models in a flow and using neural networks to go directly to final output. For example, a model that takes in audio may first turn the audio to text, then extract meaning, and then do mapping to outputs. Image models may identify shapes, then faces, and then backgrounds and bring it all together in the end. Deep learning replaces all the interim steps with some type of neural network that does it all in a single model.

Support Vector Machines

Support vector machines (SVMs) are supervised machine learning algorithms that are good for classification when the input data has lots of variables (that is, high dimensionality). Neural networks are a good choice if you have a large number of data observations, and SVM can be used if you don't have a lot of data observations. A general rule of thumb I use is that neural networks need 50 observations per input variable.

SVMs are primarily two-class classifiers, but multi-class methods exist as well. The idea behind SVM is to find the optimal hyperplane in n -dimensional space that provides the widest separation between the classes. This is much like finding the widest road space between crowds of people, as shown in Figure 8-31.

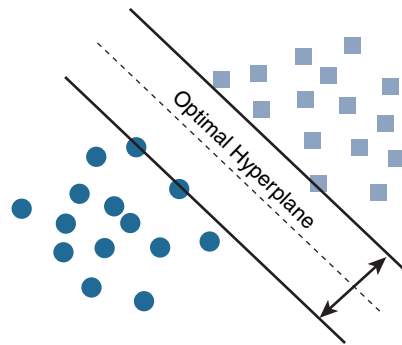


Figure 8-31 *Support Vector Machines Goal*

SVMs require explicit feature engineering to ensure that you have the dimensions that matter most for your classification. Choose SVMs over neural network classification methods when you don't have a lot of data, or your resources (such as memory) are limited. When you have a lot of data and sufficient resources and require multiple classes, neural networks may perform better. As you are learning, you may want to try them both on the same data and compare them using contingency tables.

Time Series Analysis

Time series analysis is performed for data that looks quite different at different times (for example, usage of your network during peak times versus non-peak times). Daily oscillations, seasonality on weekends or quarter over quarter, or time of year effects all come into play. This oscillation of the data over time is a leading indicator that time series analysis techniques are required.

Time series data has a lot of facets that need to be addressed in the algorithms. There are specific algorithms for time series analysis that address the following areas, as shown in Figure 8-32.

1. The data may show as cyclical and oscillating; for example, a daily chart of a help desk that closes every night shows daily activity but nothing at night.
2. There may be weekly, quarterly, or annual effects that are different from the rest of the data.
3. There may be patterns for hours when the service is not available and there is no data for that time period. (Notice the white gaps showing between daily spikes of activity in Figure 8-32.)
4. There may be longer-term trends over the entire data set.

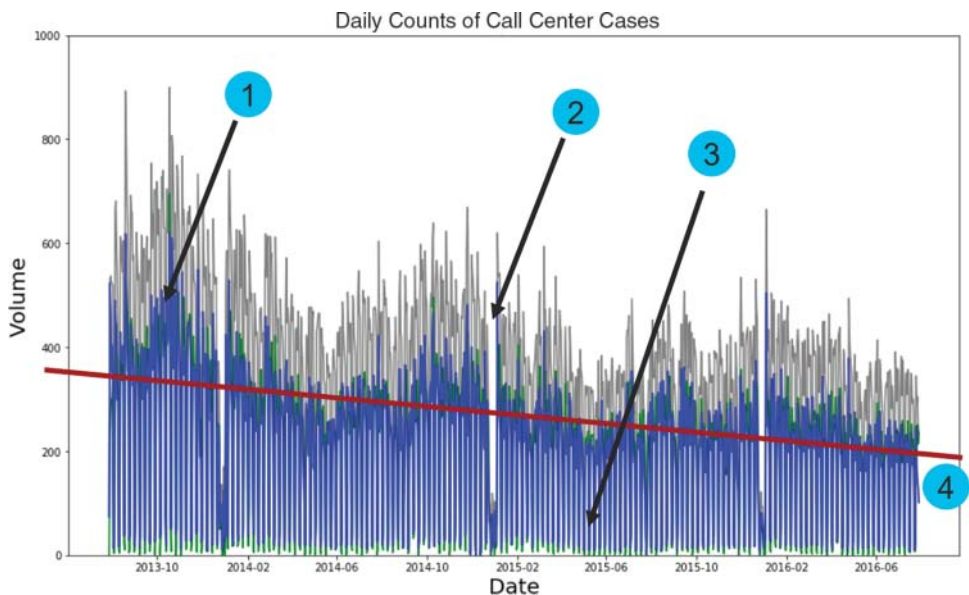


Figure 8-32 *Time Series Factors to Address in Analysis*

When you take all these factors into account, you can generate predictions that have all these components in the prediction, as shown in Figure 8-33. This prediction line was generated from an autoregressive integrated moving average (ARIMA) model.

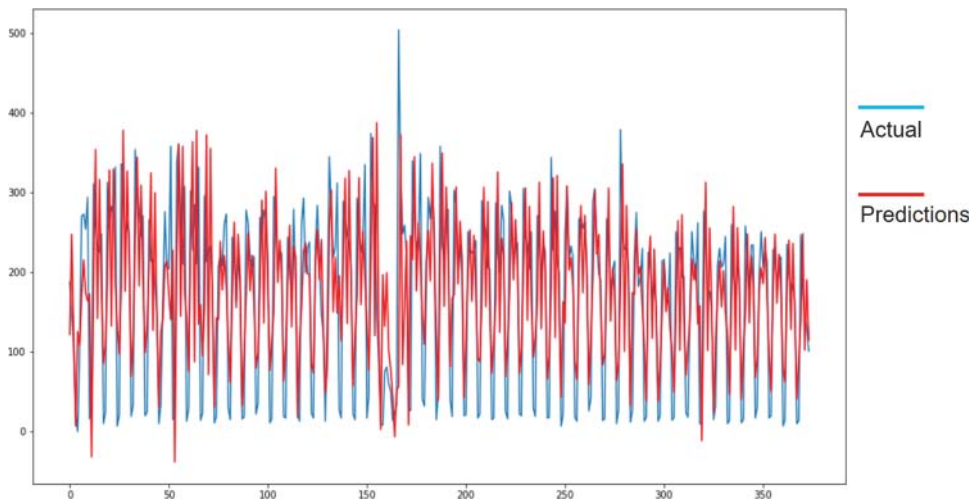


Figure 8-33 *Example of Time Series Predictions*

If you don't use time series models on this type of data, your predictions may not be any better than a rolling average. In Figure 8-34, the rolling average crosses right over the low sections that are clearly visible in the data.

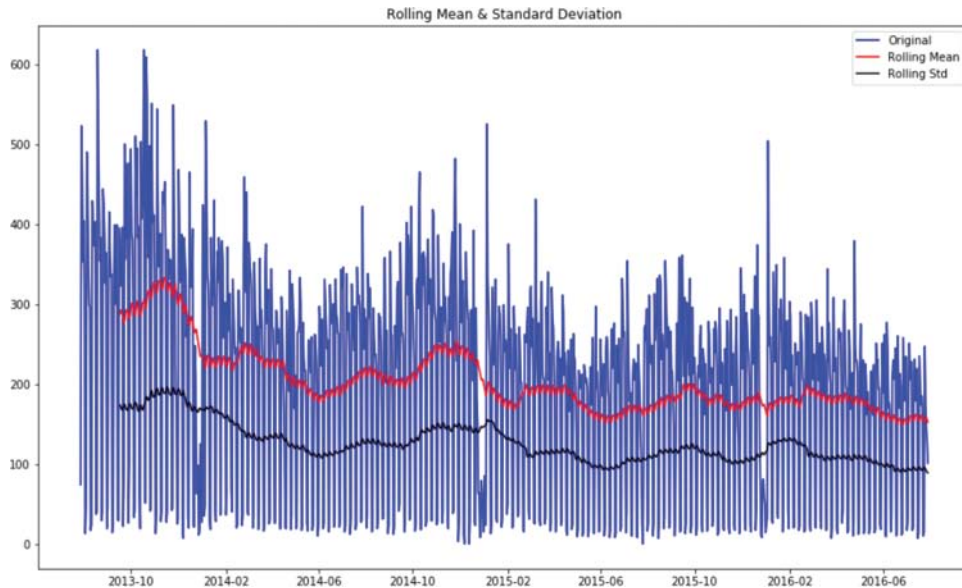


Figure 8-34 *Rolling Average Missing Dropout in a Time Series*

Many components must be taken into account in time series analysis. Here are some terms to understand as you explore:

- *Dependence* is the association of two observations to some variable at prior time points.
- *Stationarity* is the mean (average) value of a time series. You seek to adjust stationarity to level out the series for analysis.
- *Seasonality* is seasonal dependency in the data that is indicated by changes in amplitude of the oscillations in the data over time.
- *Exponential smoothing techniques* are used for forecasting the next time period based on the current and past time periods, taking into account effects by using alpha, gamma, phi, and delta components. These components give insight into what the algorithms must address in order to increase accuracy.
- *Alpha* defines the degree of smoothing to use when using past data and current data to develop forecasts.
- *Gamma* is used to smooth out long-term trends from the past data in linear and exponential trend models.

- *Phi* is used to smooth out long-term trends from the past data in damped trend models.
- *Delta* is used to smooth seasonal components in the data, such as a holiday sales component in a retail setting.
- *Lag* is a measure of seasonal autocorrelation, or the amount of correlation a current prediction has with a past (lagged) variable.
- *Autocorrelation function (ACF)* and *partial autocorrelation function (PACF)* charts allow you to examine seasonality of data.
- *Autoregressive process* means that current elements in a time series may be related to some past element in the past data (lag).
- *Moving average* adjusts for past errors that cannot be accounted for in the autoregressive modeling.
- *Autoregressive integrated moving average (ARIMA)*, also known as the *Box–Jenkins method*, is a common technique for time series analysis that is used in many packages. All the preceding factors are addressed during the modeling process.
- ARCH, GARCH, and VAR are other models to explore for time series work.

As you can surmise from this list, quite a few adjustments are made to the time series data as part of the modeling process. Time series modeling is useful in networking data plane analysis because you generally have well-known busy hours for most environments that show oscillations. There may or may not be a seasonal component, depending on the application. As you have seen in the diagrams in this section, call center cases also exhibit time series behaviors and require time series awareness for successful forecasting and prediction.

Text and Document Analysis

Whether you are analyzing documents or performing feature engineering, you need to manipulate text. Preparing data and features for analysis requires the encoding of documents into formats that fit the algorithms. Once you perform these encodings, there are many ways to use the representations in your use cases.

Natural Language Processing (NLP)

NLP includes cleaning and setting up text for analysis, and it has many parts, such as regular expressions, tokenizing, *N*-gram generation, replacements, and stop words. The core value of NLP is getting to the meaning of the text. You can use NLP techniques to manipulate text and extract that meaning.

Here are some important things to know about NLP:

- If you split up this sentence into the component words with no explicit order, you would have a bag of words. This representation is used in many types of document and text analysis.
- The words in sentences are *tokenized* to create the bag of words. Tokenizing is splitting the text into tokens, which are words or *N*-grams.
- *N*-grams are created by splitting your sentences into bigrams, trigrams, or longer sets of words. They can overlap, and the order of words can contribute to your analysis. For example, the trigrams in the phrase “The cat is really fat” are as follows:
 - The cat is
 - Cat is really
 - Is really fat
- With *stop words* you remove common words from the analysis so you can focus on the meaningful words. In the preceding example, if you remove “the” and “really,” you are left with “cat is fat.” In this case, you have reduced the trigrams by two-thirds yet maintained the essence of the statement.
- You can stem and lemmatize words to reduce the dimensionality and improve search results. *Stemming* is a process of chopping off words to the word stem. For example, the word *stem* is the stem of *stems*, *stemming*, and *stemmed*.
- *Lemmatization* involves providing proper contextual meaning to a word rather than just chopping off the end. You could replace *stem* with *truncate*, for example, and have the same meaning.
- You can use part-of-speech tagging to identify nouns, verbs, and other parts of speech in text.
- You can create term-document and document-term matrices for topic modeling and information retrieval.

Stanford CoreNLP, OpenNLP, RcmdrPlugin.temis, tm, and NLTK are popular packages for doing natural language processing. You are going to spend a lot of time using these types of packages in your future engineering efforts and solution development activities. Spend some time getting to know the functions of your package of choice.

Information Retrieval

There are many ways to develop information retrieval solutions. Some are as simple as parsing out your data and putting it into a database and performing simple database queries against it. You can add regular expressions and fuzzy matching to get great results. When building information retrieval using machine learning from sets of unstructured text (for example, Internet documents, your device descriptions, your custom strings of valuable information), the flow generally works as shown in Figure 8-35.

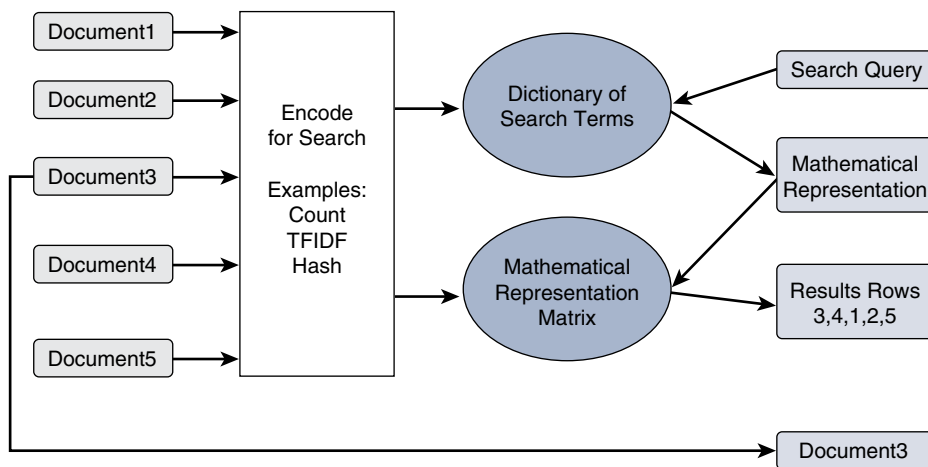


Figure 8-35 *Information Retrieval System*

In this common method, documents are parsed and key terms of interest are gathered into a dictionary. Using numerical representations from the dictionary, a full collection of encoded mathematical representations is saved as a set from which you can search. There are multiple choices for the encoding, such as term frequency/inverse document frequency (TF/IDF) and simple term counts. New documents can be easily added to the index as you develop or discover them.

Searches against your index are performed by taking your search query, developing a mathematical representation of it, and comparing that to every row in the matrix, using some similarity metric. Each row represents a document, and the row numbers of the closest matches indicate the original document numbers to be returned to the user.

Here are a few tricks to use to improve your search indexes:

- Develop a list of stop words to leave out of the search indexes. It can include common words such as *the*, *and*, *or*, and any custom words that you don't want to be searchable in the index.
- Choose to return the original document or use the dictionary and matrix representation if you are using the search programmatically.
- Research enhanced methods if the order of the terms in your documents matters. This type of index is built on a simple bag of words premise where order does not matter. You can build the same index with *N*-grams included (small phrases) to add some rudimentary order awareness.

The Python Gensim package makes this very easy and is the basis for a fingerprinting example you will build in Chapter 11, “Developing Real Use Cases: Network Infrastructure Analytics.”

Topic Modeling

Topic modeling attempts to uncover common topics that occur in documents or sets of text. The underlying idea is that every document is a set of smaller topics, just as everything is composed of atoms. You can find similar documents by finding documents that have similar topics. Figure 8-36 shows how we use topic modeling with configured features in Cisco Services, using latent Dirichlet allocation (LDA) from the Gensim package.

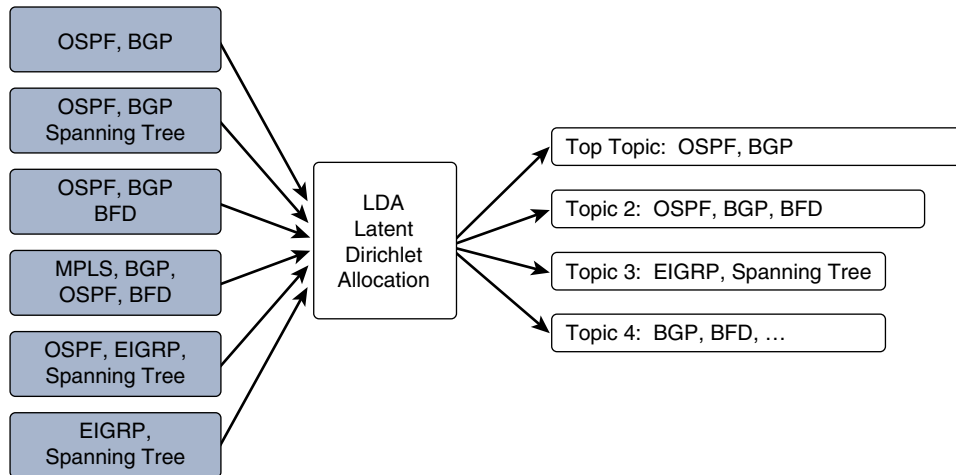


Figure 8-36 *Text and Document Topic Mining*

LDA identifies atomic units that are found together across the inputs. The idea is that each input is a collection of some number of groups of these atomic topics. As shown in the simplified example in Figure 8-36, you can use configuration documents to identify common configuration themes across network devices. Each device representation on the left has specific features represented. Topic modeling on the right can show common topics among network devices.

Latent semantic analysis (LSA) is another method for document evaluation. The idea is that there are latent factors that relate the items, and techniques such as singular value decomposition (SVD) are used to extract these latent factors. Latent factors are things that cannot be measured but that explain related items. Human intelligence is often described as being latent because it is not easy to measure, yet you can identify it when comparing activities that you *can* describe.

SVD is a technique that involves extracting concepts from the document inputs and then creating matrices of input row (document) and concepts strength. Documents with similar sets of concepts are similar because they have similar affinity toward that concept. SVD is used for solutions such as movie-to-user mappings to identify movie concepts.

Latent semantic indexing (LSI) is an indexing and retrieval method that uses LSA and SVD to build the matrices and creates indexes that you can search that are much more

advanced than simple keyword searches. The Gensim package is very good for both topic modeling and LSI.

Sentiment Analysis

Earlier in this chapter, as well as in earlier chapters, you read about soft data, and making up your own features to improve performance of your models. Sentiment analysis is an area that often contains a lot of soft data. Sentiment analysis involves analyzing positive or negative feeling toward an entity of interest. In human terms, this could be how you feel about your neighbor, dog, or cat.

In social media, Twitter is fantastic for figuring out the sentiment on any particular topic. *Sentiment*, in this context, is how people feel about the topic at hand. You can use NLP and text analytics to segment out the noun or topic, and then you can evaluate the surrounding text for feeling by scoring the words and phrases in that text. How does sentiment analysis relate to networking? Why does this have to be written language linguistics? Who knows the terminology and slang in your industry better than you?

What is the noun in your network? Is it your servers, your routers or switches, or your stakeholders? What if it is your Amazon cloud-deployed network functions virtualization stack? Regardless of the noun, there are a multitude of ways it can speak to you, and you can use sentiment analysis techniques to analyze what it is saying. Recall the push data capabilities from Chapter 4: You can have a constant “Twitter feed” (syslog) from any of your devices and use sentiment analysis to analyze this feed. Further, using machine learning and data mining, you can determine the factors most loosely associated with negative events and automatically assign negative weights to those items most associated with the events.

You may choose to associate the term *sentiment* with models such as logistic regression. If you have negative factor weights to predict a positive condition, can you determine that the factor is a negative sentiment factor? You can also use the push telemetry, syslog, and any “neighbor tattletale” functions to get outside perspective about how the device is acting. Anything that is data or metadata about the noun can contribute to sentiment. You can tie this directly to health. If you define metrics or model inputs that are positive and negative categorical descriptors, you can then use them to come up with a health metric: Sentiment = Health in this case.

Have you ever had to fill out surveys about how you feel about something? If you are a Cisco customer, you surely have done this because customer satisfaction is a major metric that is tracked. You can ask a machine questions by polling it and assigning sentiment values based on your knowledge of the responses. Why not have regular survey responses from your network devices, servers, or other components so they can tell you how they feel? This is a telemetry use case and also a monitoring case. However, if you also view this as a sentiment case, you now have additional ways to segment your devices into ones that are operating fine and ones that need your attention.

Sentiment analysis on anything is accomplished by developing a scoring dictionary of positive/negative data values. Recognize that this is the same as turning your expert

systems into algorithms. You already know what is good and bad in the data, but do you score it in aggregate? By scoring sentiment, you identify the highest (or lowest) scored network elements relative to the sentiment system you have defined.

Other Analytics Concepts

This final section touches on a few additional areas that you will encounter as you research algorithms.

Artificial Intelligence

I subscribe to the simple view that making decisions historically made by humans with a machine is low-level artificial intelligence. Some view artificial intelligence as thinking, talking robots, which is also true but with much more sophistication than simply automating your expert systems. If a machine can *understand* the current state and make a decision about what to do about it, then it fits my definition of simple artificial intelligence. Check out Andrew Ng, Ray Kurzweil, or Ben Goertzel on YouTube if you want some other interesting perspectives. The alternative to my simple view is that artificial intelligence can uncover and *learn* the current state on its own and then respond accordingly, based on response options gained through the use of reward functions and reinforcement learning techniques. *Artificial general intelligence* is a growing field of research that is opening the possibility for artificial intelligence to be used in many new areas.

Confusion Matrix and Contingency Tables

When you are training your predictive models on a set of data that is split into training and test data, a contingency table (also called confusion matrix), as shown in Figure 8-37, allows you to characterize the effectiveness of the model against the training and test data. Then you can change parameters or use different classifier models against the same data. You can collect contingency tables from models and compare them to find the best model for characterizing your input data.

Contingency Table		Observed	
		Yes	No
Model Predicted	Yes	A (TP)	B (FP)
	No	C (FN)	D (TN)

Figure 8-37 Contingency Table for Model Validation

You can get a wealth of useful data from this simple table. Many of the calculations have different descriptions when used for different purposes:

- A and D are the correct predictions of the model that matched yes or no predictions from the model test data from the training/test split. These are true positives (TP) and true negatives (TN).
- B and C are the incorrect predictions of your model as compared to the training/test data cases of yes or no. These are the false positives (FP) and false negatives (FN).
- Define hit rate, sensitivity, recall, or true positive rate (correctly predicted yes) as the ratio of true positives to all cases of yes in the test data, defined as $A/(A+C)$.
- Define specificity or true negative rate (correctly predicted no) as the ratio of true negatives to all negatives in the test data, defined as $D/(B+D)$.
- Define false alarms or false positive rate (wrongly predicted yes) as the ratio of false positives that your model predicted over the total cases of yes in the training data, defined as $B/(B+D)$.
- Define false negative rate (wrongly predicted no) as the ratio of false negatives that your model predicted over the total cases of no in the training data, defined as $C/(A+C)$.
- The accuracy of the output is the ratio of correct predictions to either yes or no cases, which is defined as $(A+D)/(A+B+C+D)$.
- Precision is the ratio of true positives out of all positives predicted, defined as $A/(A+B)$.
- Error rate is the opposite of accuracy, and you can get it by calculating $(1-\text{Accuracy})$, which is the same as $(B+C)/(A+B+C+D)$.

Why so many calculations for a simple table? Because knowledge of the domain is required with these numbers to determine the best choice of models. For example, a high false positive rate may not be desired if you are evaluating a choice that has significant cost with questionable benefit when your model predicts a positive. Alternatively, if you don't want to miss any possible positive case, then you may be okay with a high rate of false positives. So how do people make evaluations? One way is to use a receiver operating characteristic (ROC) diagram that evaluates all the characteristics of many models in one diagram, as shown in Figure 8-38.

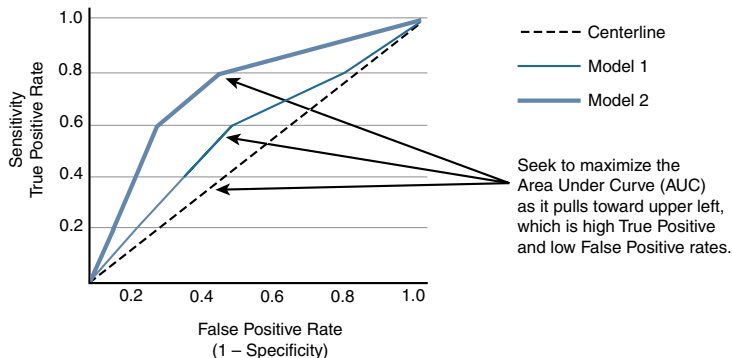


Figure 8-38 Receiver Operating Characteristic (ROC) Diagram

Cumulative Gains and Lift

When you have a choice to take actions based on models that you have built, you sometimes want to rank those options so you can work on those that have the greatest impacts first. In the churn model example shown in Figure 8-39, you may seek to rank the customers for which you need to take action. You can rank the customers by value and identify which ones your models predict will churn. You ultimately end up with a list of items that your models and calculations predict will have the most benefit.

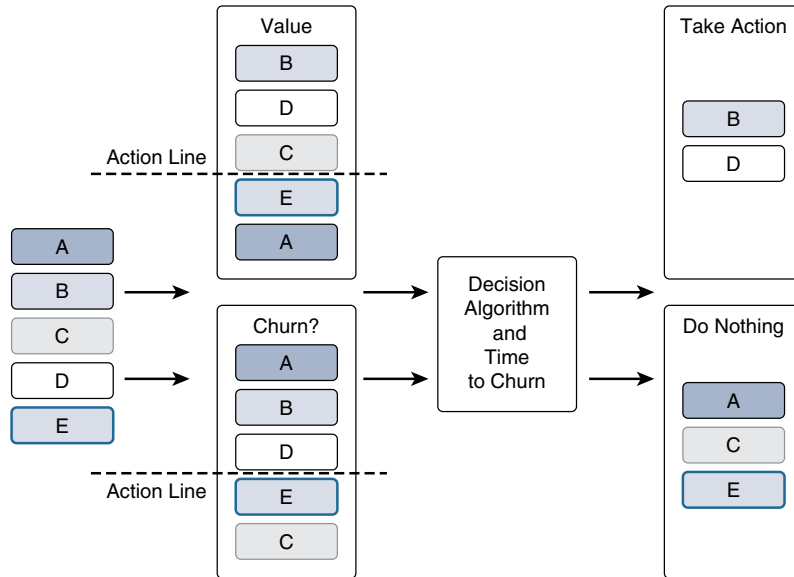


Figure 8-39 Churn Model Workflow Example

You use cumulative gains and lift charts to help with such ranking decisions. You determine what actions have the most impact by looking at the lift of those actions. Your value of those customers is one type of calculation, and you can assign values to actions and use the same lift-and-gain analysis to evaluate those actions. A general process for using lift and gain is as follows:

1. You can use your classification models to assign a score for observations in the validation sets. This works with classification models that predict some probability, such as propensity to churn or fail.
2. You can assign the random or average unsorted value as the baseline in a chart.
3. You can rank your model predictions by decreasing probability that the predicted class (churn, crash, fail) will occur.
4. At each increment of the chart (1%, 5% 10%), you can compare the values from the ranked predictions to the baseline and determine how much better the predictions are at that level to generate a lift chart.

Figure 8-40 is a lift chart that provides a visual representation of these steps.

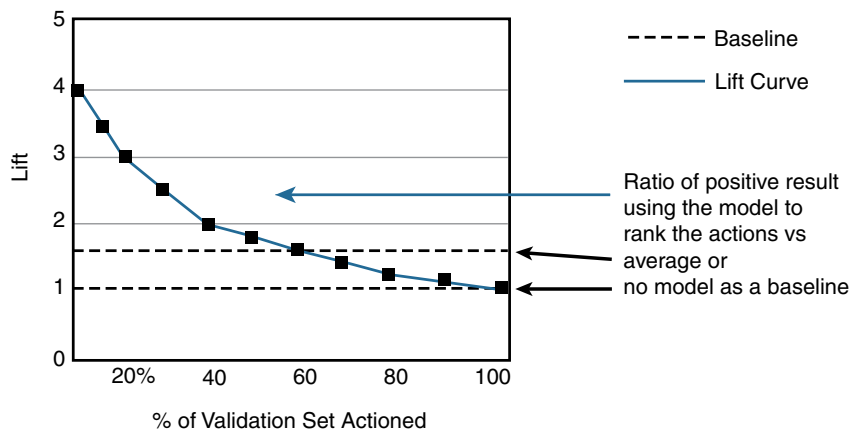


Figure 8-40 *Lift Chart Example*

Notice that the top 40% of the predictions in this model show a significant amount of lift over the baseline using the model. You can use such a chart for any analysis that fits your use case. For example, the middle dashed line may represent the place where you decide to take action or not. You first sort actions by value and then use this chart to examine lift.

If you work through every observation, you can generate a cumulative gains chart against all your validation data, as shown in Figure 8-41.

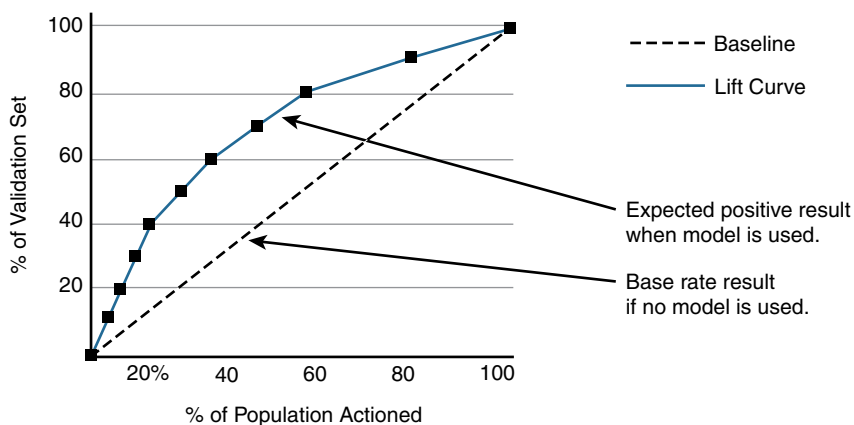


Figure 8-41 *Cumulative Gains Chart*

Cumulative gains charts are used in many facets of analytics. You can use these charts to make decisions as well as to provide stakeholders with visual evidence that your analysis provides value. Be creative with what you choose for the axis.

Simulation

Simulation involves using computers to run through possible scenarios when there may not be an exact science for predicting outcomes. This is a typical method for predicting sports event outcomes where there are far too many variables and interactions to build a standard model. This also applies to complex systems that are built in networking.

Monte Carlo simulation is used when systems have a large number of inputs that have a wide range of variability and randomness. You can supply the analysis with the ranges of possible value for the inputs and run through thousands of simulations in order to build a set of probable outcomes. The output is a probability distribution where you find the probabilities of any possible outcome that the simulation produced.

Markov Chain Monte Carlo (MCMC) systems use probability distributions for the inputs rather than random values from a distribution. In this case, your simulated inputs that are more common are used more during the simulations. You can also use random walk inputs with Monte Carlo analysis, where the values move in stepwise increments, based on previous values or known starting points.

Summary

In Chapters 5, 6, and 7, you stopped to think by learning about cognitive bias, to expand upon that thinking by using innovation techniques, and to prime your brain with ideas by reviewing use-case possibilities. You collected candidate ideas throughout that process. In this chapter, you have learned about many styles of algorithms that you can use to realize your ideas in actual models that provide value for your company. You now have a broad perspective about algorithms that are available for developing innovative solutions.

You have learned about the major areas of supervised and unsupervised machine learning and how to use machine learning for classification, regression, and clustering. You have learned that there are many other areas of activity, such as feature selection, text analytics, model validation, and simulation. These ancillary activities help you use the algorithms in a very effective way. You now know enough to choose candidate algorithms to solve your problem. You need to do your own detailed research to see how to make an algorithm fit your data or make your data fit an algorithm.

You don't always need to use an algorithm. Sometimes you do not need *analytics* algorithms. If you take the knowledge in your expert systems and build an algorithm from it that you can programmatically apply, then you have something of value. You have something of high value that is unique when you take the outputs of your expert algorithms as inputs to analytics algorithms. This is a large part of the analytics success in Cisco Services. Many years of expert systems have been turned into algorithms as the basis for next-level models based on machine learning techniques, as described in this chapter.

This is the final chapter in this book about collecting information and ideas. Further research is up to you and depends on your interests. Using what you have learned in the book to this point, you should have a good idea of the algorithms and use cases that you can research for your own innovative solutions. The following chapters move into what it takes to develop those ideas into real use cases by doing detailed walkthroughs of building solutions.

This page intentionally left blank

Building Analytics Use Cases

As I moved from being a network engineer to being a network engineer *with some data science skills*, I spent my early days trying to figure out how to use my network engineering and design skills to do data science work. After the first few years, I learned that simply building an architecture to gather data did not lead to customer success like building resilient network architectures enabled new business success. I could build a dozen big data environments. I could get the quick wins of setting up full data pipelines into centralized repositories. But I learned that was not enough. The real value comes from applying additional data feature engineering, analysis, trending, and visualization to uncover the unknowns and solve business problems.

When a business does not know how to use data to solve problems, the data just sits in repositories. The big data storage environments become big-budget drainers and data sinkholes rather than data analytics platforms. You need to approach data science solutions in a different way from network engineering problems. Yes, you can still start with data as your guide, but you must be able to manipulate the data in ways that allows you to uncover things you did not know. The traditional approach of experiencing a problem and building a rule-based system to find that problem is still necessary, but it is no longer enough. Networks are transforming, abstraction layers (controller-based architectures) are growing, and new ways must be developed to optimize these environments. Data science and analytics combined with automation in full-service assurance systems provide the way forward.

This short chapter introduces the next four chapters on use cases (Chapter 10, “Developing Real Use Cases: The Power of Statistics,” Chapter 11, “Developing Real Use Cases: Network Infrastructure Analytics,” Chapter 12, “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry,” and Chapter 13, “Developing Real Use Cases: Data Plane Analytics”) and shows you what to expect to learn from them. You will spend a lot of time manipulating data and writing code if you choose to follow along with your own analysis. In this chapter you can start your 10,000 hours of deliberate practice on the many foundational skills you need to know to be successful. The point of

the following four chapters is not to show you the results of something I have done; they are very detailed to enable you to use the same techniques to build your own analytics solutions using your own data.

Designing Your Analytics Solutions

As outlined in Chapter 1, “Getting Started with Analytics,” the goal of this book is to get you to enough depth to design analytics use cases in a way that guides you toward the low-level data design and data representation that you need to find insights. Cisco uses a narrowing scope design method to ensure that all possible options and requirements are covered, while working through a process that will ultimately provide the best solution for customers. This takes breadth of focus, as shown in Figure 9-1.

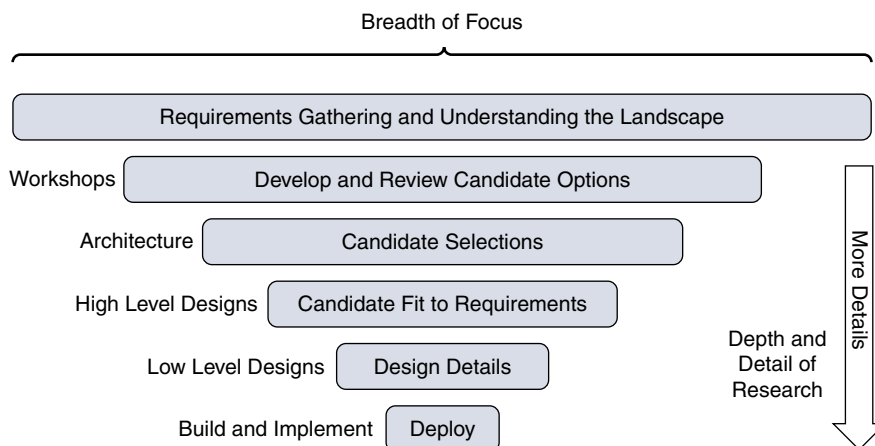


Figure 9-1 *Breadth of Focus for Analytics Solution Design*

Once you have data and a high-level idea of the use case you want to address with that data, it seems like you are almost there. Do not have the expectation that it is easier from that point. It may or may not be harder for you from here, but it is going to take more time. Your time spent actually building the use case that you design is the inverse of the scope scale, as shown in Figure 9-2. You will have details and research to do outside this book to refine the details of your use case based on your algorithm choices.

You will often see analytics solutions stop before the last deployment step. If you build something useful, take the time to put it into a production environment so that you and others can benefit from it and enhance it over time. If you implement your solution and nobody uses it, you should learn why they do not use it and pivot to make improvements.

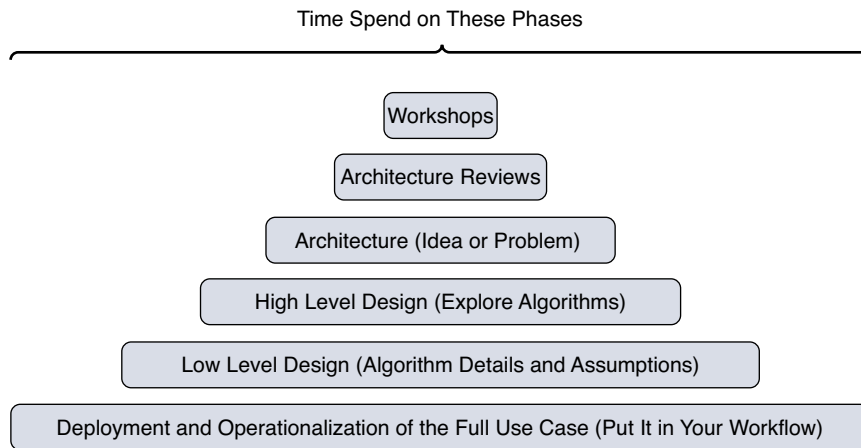


Figure 9-2 *Time Spend for Phases of Analytics Execution*

Using the Analytics Infrastructure Model

As you learned in Chapter 2, “Approaches for Analytics and Data Science,” you can use the analytics infrastructure model for simplified conversations with your stakeholders and to identify the initial high-level requirements. However, you don’t stop using it there. Keep the model in your mind as you develop use cases. For example, it is very common to develop data transformations or filters using data science tools as you build models. For data transformation, normalization, or standardization, it is often desirable to do that work closer to the source of data. You can bring in all the data to define and build these transformations as a first step and then push the transformations back into the pipeline as a second step, as shown in Figure 9-3.

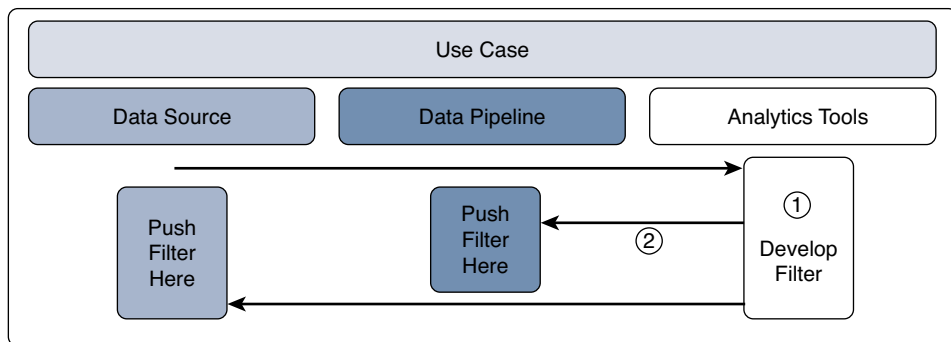


Figure 9-3 *Using the Analytics Infrastructure Model to Understand Data Manipulation Locations*

Once you develop a filter or transformation, you might want to push it back to the storage layer—or even all the way back to the source of the data. It depends on your specific scenario. Some telemetry data from large networks can arrive at your systems with volumes of terabytes per day. You may desire to push a filter all the way back to the source of the data to drop useless parts of data in that case. Oftentimes you can apply preprocessing at the source to save significant cost. Understanding the analytics infrastructure model components for each of your use cases helps you understand the optimal place to deploy your data manipulations when you move your creations to production.

About the Upcoming Use Cases

The use cases described in the next four chapters teach you how to use a variety of analytics tools and techniques. They focus on the analytics tools side of the analytics infrastructure model. You will learn about Jupyter Notebook, Python, and many libraries you can use for data manipulation, statistics, encoding, visualization, and unsupervised machine learning.

Note There are no supervised learning, regression, or predictive examples in these chapters. Those are advanced topics that you will be ready to tackle on your own after you work through the use cases in the next four chapters.

The Data

The data for the first three use cases is anonymized data from environments within Cisco Advanced Services. Some of the data is from very old platforms, and some is from newer instances. This data will not be shared publicly because it originated from various customer networks. The data anonymization is very good on a per-device basis, but sharing the overall data set would provide insight about sizes and deployment numbers that could raise privacy concerns. You will see the structure of the data so you can create the same data from your own environment. Anonymized historical data is used for Chapters 10, 11, and 12. You can use data from your own environment to perform the same activities done here. Chapter 13 uses a publicly available data set that focuses on packet analysis; you can download this data set and follow along.

All the data you will work with in the following chapters was preprocessed. How? Cisco established data connections with customers, including a collector function that processes locally and returns important data to Cisco for further analysis. The Cisco collectors, using a number of access methods, collect the data from selected customer network devices and securely transport the data (some raw, some locally processed and filtered) back to Cisco. These individual collections are performed using many access mechanisms for millions of devices across Cisco Advanced Services customers, using the process shown in Figure 9-4.

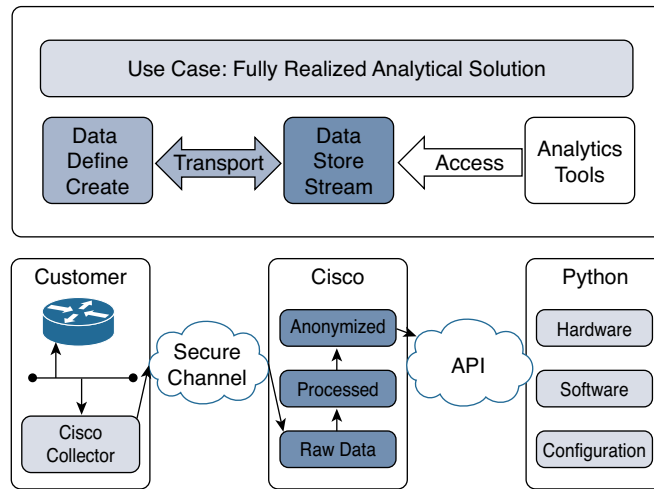


Figure 9-4 *Analytics Infrastructure Model Mapped to Cisco Advanced Services Data Acquisition*

After secure transmission to Cisco, the data is processed using expert systems. These expert systems were developed over many years by thousands of Cisco engineers and are based on the lessons learned from actual customer engagements. This book uses some anonymized data from the hardware, software, configuration, and syslog modeling capabilities.

Chapters 10 and 11 use data from the management plane of the devices. Figure 9-5 shows the high-level flow of the data cleansing process.

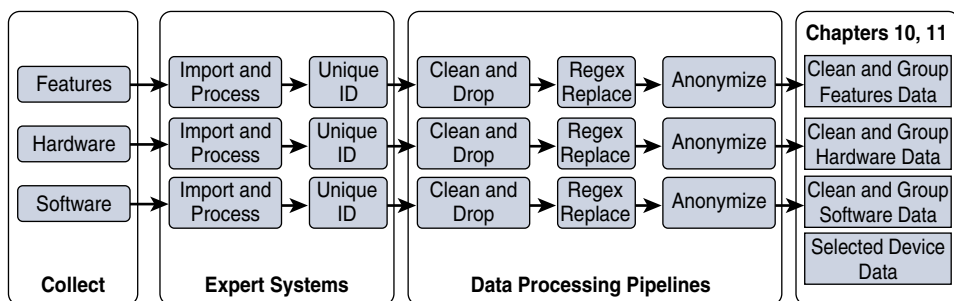


Figure 9-5 *Data Processing Pipeline for Network Device Data*

For the statistical use case, statistical analysis techniques are learned using the selected device data on the lower right in Figure 9-5. This data set contains generalized hardware, software, and last reload information. Then some data science techniques are learned using the entire set of hardware, software, and feature information from the upper-right side of Figure 9-5 in Chapter 11.

The third use case moves from the static metadata to event log telemetry. Syslog data was gathered and prepared for analysis using the steps shown in Figure 9-6. Filtering was applied to remove most of the noise so you can focus on a control plane use case.

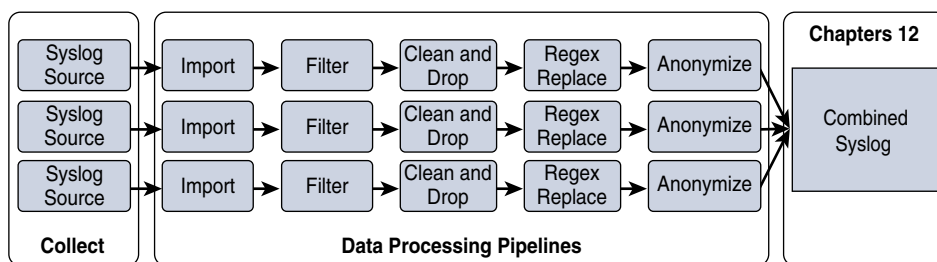


Figure 9-6 Data Processing Pipeline for Syslog Data

Multiple pipelines in the syslog case are gathered over the same time window so that a network with multiple locations can be simulated.

The last use case moves into the data plane for packet-level analysis. The packet data used is publicly available at <http://www.netresec.com/?page=MACCDC>.

The Data Science

As you go through the next four chapters, consider what you wrote down from your innovation perspectives. Be sure to spend extra time on any use-case areas that relate to solutions you want to build. The goal is to get enough to be comfortable getting hands-on with data so that you can start building the parts you need in your solutions.

Chapter 10 introduces Python, Jupyter, and many data manipulation methods you will need to know. Notice in Chapter 10 that the cleaning and data manipulation is ongoing and time-consuming. You will spend a significant amount of time working with data in Python, and you will learn many of the necessary methods and libraries. From a data science perspective, you will learn many statistical techniques, as shown in Figure 9-7.

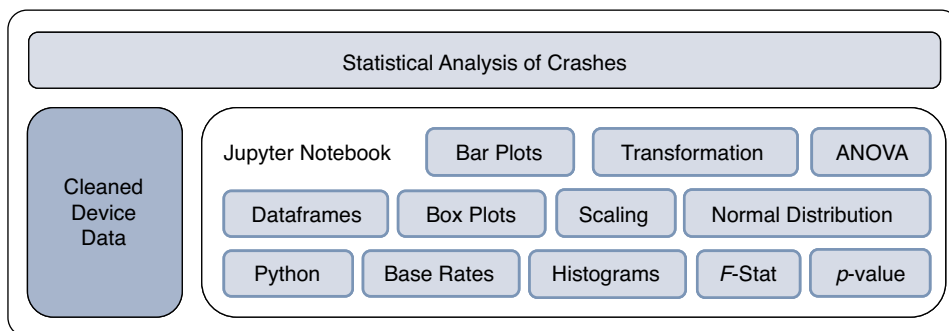


Figure 9-7 Learning in Chapter 10

Chapter 10 uses the statistical methods shown in Figure 9-7 to help you understand stability of software versions. Statistics and related methods are very useful for analyzing network devices; you don't always need algorithms to find insights.

Chapter 11 uses more detailed data than Chapter 10; it adds hardware, software, and configuration features to the data. Chapter 11 moves from the statistical realm to a machine learning focus. You will learn many data science methods related to unsupervised learning, as shown in Figure 9-8.

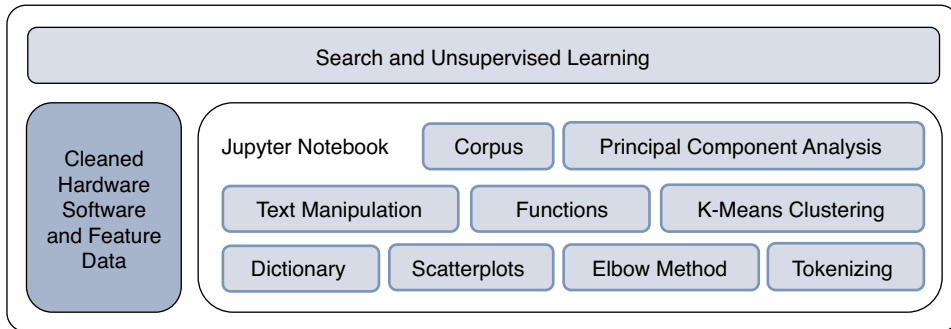


Figure 9-8 *Learning in Chapter 11*

By the end of Chapter 11 you will have the skills needed to build a search index for anything that you can model with a set of data. You will also learn how to visualize your devices using machine learning.

Chapter 12 shifts focus to looking at a control plane protocol, using syslog telemetry data. Recall that telemetry, by definition, is data pushed by a device. This data shows what the device says is happening via a standardized message format. The control plane protocol used for this chapter is the Open Shortest Path First (OSPF) routing protocol. The logs were filtered to provide only OSPF data so you can focus on the control plane activity of a single protocol. The techniques shown in Figure 9-9 are examined.

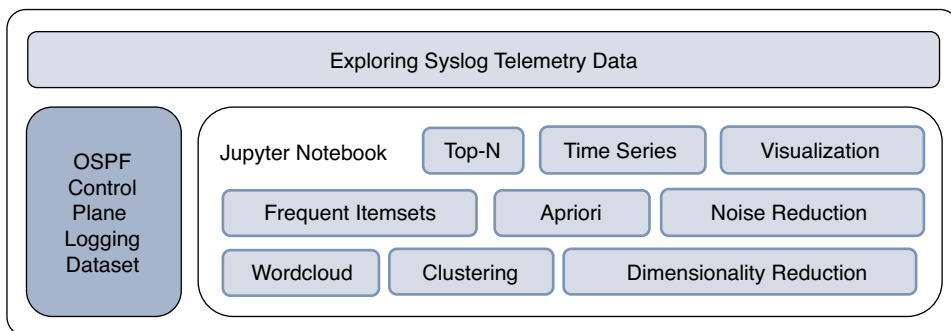


Figure 9-9 *Learning in Chapter 12*

The use case in Chapter 13 uses a public packet capture (pcap)-formatted data file that you can download and use to build your packet analysis skills. Figure 9-10 shows the steps required to gather this type of data from your own environment for your use cases. Pcap files can get quite large and can consume a lot of storage, so be selective about what you capture.

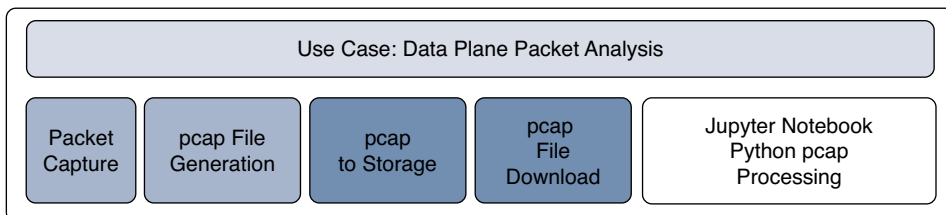


Figure 9-10 *Chapter 13 Data Acquisition*

In order to analyze the detailed packet data, you will develop scripting and Python functions to use in your own systems for packet analysis. Chapter 13 also shows how to combine what you know as an SME with data encoding skills you have learned to provide hybrid analysis that only SMEs can do. You will use the information in Chapter 13 to capture and analyze packet data right on your own computer. You will also gain rudimentary knowledge of how port scanning shows up as performed by bad actors on computer networks and how to use packet analysis to identify this activity (see Figure 9-11).

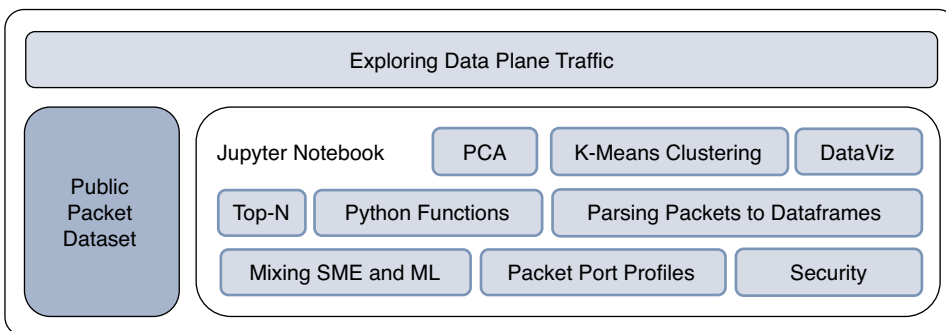


Figure 9-11 *Learning in Chapter 13*

The Code

There are probably better, faster, and more efficient ways to code many of the things you will see in the upcoming chapters. I am a network engineer by trade, and I have learned enough Python and data science to be proficient in those areas. I learn enough of each to do the analysis I wish to do, and then, after I find something that works well enough to prove or disprove my theories, I move on to my next assignment. Once I find something

that works, I go with it, even if it is not the most optimal solution. Only when I have a complete analysis that shows something useful do I optimize the code for deployment or ask my software development peers to do that for me.

From a data science perspective, there are also many ways to manipulate and work with data, algorithms, and visualizations. Just as with my Python approach, I use data science techniques that allow me to find insights in the data, whether I use them in a proper way or not. Yes, I have used a flashlight as a hammer, and I have used pipe wrenches and pliers instead of sockets to remove bolts. I find something that works enough to move me a step forward. When that way does not work, I go try something else. It's all deliberate practice and worth the exploration for you to improve your skills.

Because I am an SME in the space where I am using the tools, I am always cautious about my own biases and mental models. You cannot stop the availability cascades from popping into your head, but you can take multiple perspectives and try multiple analytics techniques to prove your findings. You will see this extra validation manifest in some of the use cases when you review findings more than one time using more than one technique.

As you read the following chapters, follow along with Internet searches to learn more about the code and algorithms. I try to explain each command and technique that I use as I use it. In some cases, my explanations may not be good enough to create understanding for you. Where this is the case, pause and go do some research on the command, code, or algorithm so you can see why I use it and how it did what it did to the data.

Operationalizing Solutions as Use Cases

The following four chapters provide ways that you can operationalize the solutions or develop reusable components. These chapters include many Python functions and loops as part of the analysis. One purpose is to show you how to be more productive by scripting. A secondary purpose is to make sure you get some exposure to automation, scripting, or coding if you do not already have skills in those areas.

As you work through model building exercises, you often have access to visualizations and validations of the data. When you are ready to deploy something to production so that it works all the time for you, you may not have those visualizations and validations. You need to bubble up your findings programmatically. Seek to generate reusable code that does this for you.

In the solutions that you build in the next four chapters, many of the findings are capabilities that enhance other solutions. Some of them are useful and interesting without a full implementation. Consider operationalizing anything that you build. Build it to run continuously and periodically send you results. You will find that you can build on your old solutions in the future as you gain more data science skills.

Finally, revisit your deployments periodically and make sure they are still doing what you designed them to do. As data changes, your model and analysis techniques for the data may need to change accordingly.

Understanding and Designing Workflows

In order to maximize the benefit of your creation, consider how to make it best fit the workflow of the people who will use it. Learn where and when they need the insights from your solution and make sure they are readily available in their workflow. This may manifest as a button on a dashboard or data underpinning another application.

In the upcoming chapters, you will see some of the same functionality used repeatedly. When you build workflows and code in software, you often reuse functionality. You can codify your expertise and analysis so that others in your company can use it to start finding insights. In some cases, it might seem like you are spending more time writing code than analyzing data. But you have to write the code only one time. If you intend to use your analysis techniques repeatedly, script them out and include lots of comments in the code so you can add improvements each time you revisit them.

Tips for Setting Up an Environment to Do Your Own Analysis

The following four chapters employ many different Python packages. Python in a Jupyter Notebook environment is used for all use cases. The environment used for this work was a CentOS7 virtual machine in a Cisco data center with Jupyter Notebook installed on that server and running in a Chrome browser on my own computer.

Installing Jupyter Notebook is straightforward. Once you have a working Notebook environment set up, it is very easy to install any packages that you see in the use-case examples, as shown in Figure 9-12. You can run any Linux command-line interface (CLI) from Jupyter by using an exclamation point preceding the command.

```
1 !pip install --upgrade pip

Collecting pip
  Downloading https://files.pythonhosted.org/packages/0f/74/ecd13431bcc456ed390
b44c8a6e917c1820365cbebc6a8974d1cd045ab4/pip-10.0.1-py2.py3-none-any.whl (1.3M
B)
    100% |#####| 1.3MB 931kB/s eta 0:00:01
Installing collected packages: pip
  Found existing installation: pip 9.0.3
    Uninstalling pip-9.0.3:
      Successfully uninstalled pip-9.0.3
  Successfully installed pip-10.0.1
```

Figure 9-12 *Installing Software in Jupyter Notebook*

If you are not sure if you have a package, just try to load it, and your system will tell you if it already exists, as shown in Figure 9-13.

```

1 !pip install pandas

Requirement already satisfied: pandas in /scratch/virtualenv/johng_notebooks/lib/python2.7/site-packages (0.20.3)
Requirement already satisfied: pytz>=2011k in /scratch/virtualenv/johng_notebooks/lib/python2.7/site-packages (from pandas) (2017.2)
Requirement already satisfied: numpy>=1.7.0 in /scratch/virtualenv/johng_notebooks/lib/python2.7/site-packages (from pandas) (1.13.1)
Requirement already satisfied: python-dateutil in /scratch/virtualenv/johng_notebooks/lib/python2.7/site-packages (from pandas) (2.6.1)
Requirement already satisfied: six>=1.5 in /scratch/virtualenv/johng_notebooks/lib/python2.7/site-packages (from python-dateutil->pandas) (1.10.0)

```

Figure 9-13 *Installing Required Packages in Jupyter Notebook*

The following four chapters use the packages listed in Table 9-1. If you are not using Python, you can find packages in your own preferred environment that provide similar functionality. If you want to get ready beforehand, make sure that you have all of these packages available; alternatively, you can load them as you encounter them in the use cases.

Table 9-1 *Python Packages Used in Chapters 10–13*

Package	Purpose
pandas	Dataframe; used heavily in all chapters
scipy	Scientific Python for stats and calculations
statsmodels	Common stats functions
pylab	Visualization and plotting
numpy	Python arrays and calculations
NLTK	Text processing
Gensim	Similarity indexing, dictionaries
sklearn (Scikit-learn)	Many analytics algorithms
matplotlib	Visualization and plotting
wordcloud	Visualization
mlextract	Transaction analysis

Even if you are spending a lot of time learning the coding parts, you should still take some time to focus on the intuition behind the analysis. Then you can repeat the same procedures in any language of your choosing, such as Scala, R, or PySpark, using the proper syntax for the language. You will spend extra time porting these commands over, but you can take solace in knowing that you are adding to your hours of deliberate practice. Researching the packages in other languages may have you learning multiple

languages in the long term if you find packages that do things in a way that you prefer in one language over another. For example, if you want high performance, you may need to work in PySpark or Scala.

Summary

This chapter provided a brief introduction to the four upcoming use-case chapters. You have learned where you will spend your time and why you need to keep the simple analytics infrastructure model in the back of your mind. You understand the sources of data. You have an idea of what you will learn about coding and analytics tools and algorithms in the upcoming chapters. Now you're ready to get started building something.

Developing Real Use Cases: The Power of Statistics

In this chapter, you will start developing real use cases. You will spend a lot of time getting familiar with the data, data structures, and Python programming used for building use cases. In this chapter you will also analyze device metadata from the management plane using statistical analysis techniques.

Recall from Chapter 9, “Building Analytics Use Cases,” that the data for this chapter was gathered and prepared using the steps shown in Figure 10-1. This figure is shared again so that you know the steps to use to prepare your own data. Use available data from your own environment to follow along. You also need a working instance of Jupyter Notebook in order to follow step by step.

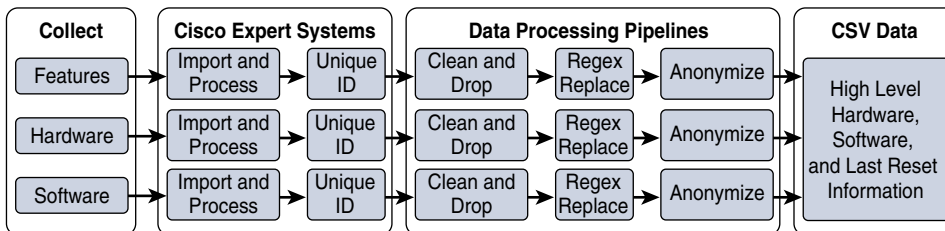


Figure 10-1 *Data for This Chapter*

This example uses Jupyter Notebook, and the use case is exploratory analysis of device reset information. Seek to determine where to focus your time for the limited downtime available for maintenance activities. You can maximize the benefit of that limited time by addressing the upgrades that remove the most risk of crashes in your network devices.

Loading and Exploring Data

For the statistical analysis in this chapter, router software versions and known crash statistics from the comma-separated variables (CSV) files are used to show you how to do descriptive analytics and statistical analysis using Python and associated data science libraries, tools, and techniques. You can use this type of analysis when examining crash rates for the memory case discussed earlier in the book. You can use the same statistics you learn here for many other types of data exploration.

Base rate statistics are important due to the law of small numbers and context of the data. Often the numbers people see are not indicative of what is really happening. This first example uses a data set of 150,000 anonymized Cisco 2900 routers. Within Jupyter Notebook, you start by importing the Python pandas and numpy libraries, and then you use pandas to load the data, as shown in Figure 10-2. The last entry in a Jupyter Notebook cell prints the results under the command window.

```
1 import pandas as pd
2 import numpy as np
3 df=pd.read_csv("/scratch/data/booktest/detailed_devices_anonymized38.csv")
4 df.columns
```

Index([u'Unnamed: 0', u'configRegister', u'productFamily', u'productId',
u'productType', u'resetReason', u'swVersion', u'id'],
dtype='object')

Figure 10-2 Loading Data from Files

The input data to the analysis server was pulled using application programming interfaces (APIs) to deliver the CSV format and then load it into Jupyter Notebook. Dataframes are much like spreadsheets. The `columns` command allows you to examine the column headers for a dataframe. In Figure 10-3, notice that a few of the rows of data in the data set were loaded from the file and were obtained by asking for a slice of the first two rows, using the square bracket notation.

```
1 df[:2]
```

configRegister	productFamily	productId	productType	resetReason	swVersion
02102	Cisco_VG_Series_Gateways	VG224	Unified_Communications	power_on	15_0_1_M
02102	Cisco_VG_Series_Gateways	VG224	Unified_Communications	power_on	15_0_1_M

Figure 10-3 Examining Data with Slicing

Dataframes are a very common data representation used for storing data for exploration and model building. Dataframes are a foundational structure used in data science, so they are used extensively in this chapter to help you learn. The pandas dataframe package is

powerful, and this section provides ample detail to show you how to use many common functions. If you are going to use Python for data science, you must learn pandas. This book only touches on the power of the package, and you might choose to learn more about pandas.

The first thing you need to do here is to drop an extra column that was generated through the use of the CSV format and that you saved without removing the previous dataframe index. Figure 10-4 shows this old index column dropped. You can verify that it was dropped by checking your columns again.

```
1 df.drop(['Unnamed: 0'], axis=1, inplace=True)

1 df.columns
Index([u'configRegister', u'productFamily', u'productId', u'productType',
      u'resetReason', u'swVersion', u'id'],
      dtype='object')
```

Figure 10-4 *Dropping Columns from Data*

There are many ways to drop columns from dataframes. In the method used here, you drop rows by index number or columns by column name. An axis of zero drops rows and an axis of one drops columns. The **inplace** parameter makes the changes in the current dataframe rather than generating a new copy of the dataframe. Some pandas functions happen in place and some create new instances. (There are many new instances created in this chapter so you can follow the data manipulations, but you can often just use the same dataframe throughout.)

Dataframes have powerful filtering capabilities. Let's analyze a specific set of items and use the filtering capability to select only rows that have data of interest for you. Make a selection of only 2900 Series routers and create a new dataframe of only the first 150,000 entries of that selection in Figure 10-5. This combines both filtering of a dataframe column and a cutoff at a specific number of entries that are true for that filter.

```
1 df2=df[df.productFamily=="Cisco_2900_Series_Integrated_Services_Routers"]\
2 [:150000].copy()

1 df2[:2]
```

	configRegister	productFamily	productId	productType	rese
325	unknown	Cisco_2900_Series_Integrated_Services_Routers	CISCO2911_K9	Routers	
346	unknown	Cisco_2900_Series_Integrated_Services_Routers	CISCO2911_K9	Routers	

Figure 10-5 *Filtering a Dataframe*

The first thing to note is that you use the backslash (\) as a Python continuation character. You use it to split commands that belong together on the same line. It is suggested to use the backslash for a longer command that does not fit onto the screen in order to see the full command. (If you are working in a space with a wider resolution, you can remove the backslashes and keep the commands together.) In this case, assign the output of a filter to a new dataframe, `df2`, by making a copy of the results. Notice that `df2` now has the 2900 Series routers that you wish to analyze. Your first filter works as follows:

- `df.productFamily` indicates that you want to examine the `productFamily` column.
- The double equal sign is the Python equality operator, and it means you are looking for values in the `productFamily` column that match the string provided for 2900 Series routers.
- The code inside the square bracket provides a **True** or **False** for every row of the dataframe.
- The `df` outside the bracket provides you with rows of the dataframe that are true for the conditions inside the brackets.
- You already learned that the square brackets at the end are used to select rows by number. In this case, you are selecting the first 150,000 entries.
- The `copy` at the end creates a new dataframe. Without the `copy`, you would be working on a view of the original dataframe. You want a new dataframe with just your entries of interest so you can manipulate it freely. In some cases, you might want to pull a slice of a dataframe for a quick visualization.

Base Rate Statistics for Platform Crashes

You now have a dataframe of 150,000 Cisco 2900 Series routers. You can see what specific 2900 model numbers you have by using the dataframe `value_counts` function, as shown in Figure 10-6. Note that there are two ways to identify columns of interest.

1	<code>df2.productId.value_counts()</code>
	CISCO2911_K9 64070
	CISCO2921_K9 47031
	CISCO2951_K9 27752
	CISCO2901_K9 11147
	Name: productId, dtype: int64
1	<code>df2['productId'].value_counts()</code>
	CISCO2911_K9 64070
	CISCO2921_K9 47031
	CISCO2951_K9 27752
	CISCO2901_K9 11147
	Name: productId, dtype: int64

Figure 10-6 *Two Ways to View Column Data*

The `value_counts` function finds all unique values in a column and provides the counts for them. In this case, the `productId` column is used to see the model types of 2900 routers that are in the data. Both methods shown are valid for selecting columns from dataframes for viewing. Using this selected data, you can perform your first visualization as shown in Figure 10-7.

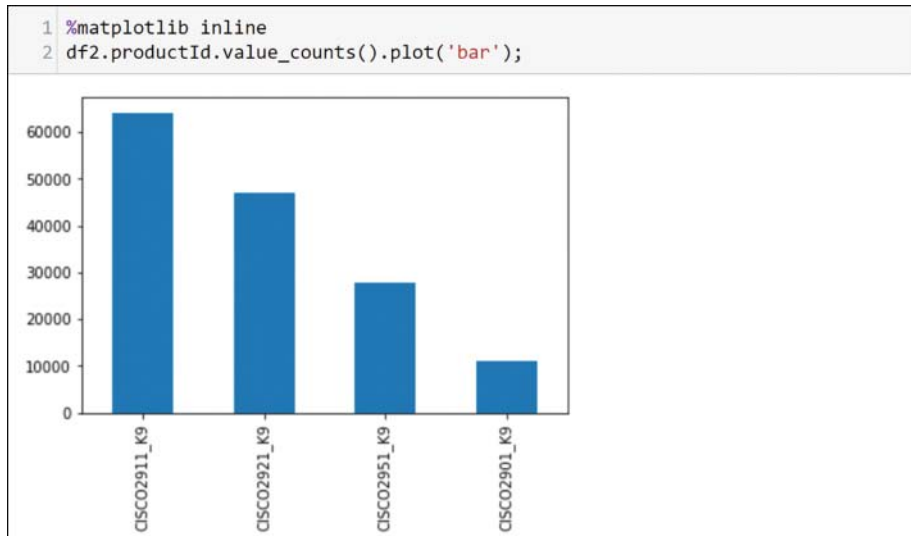


Figure 10-7 Simple Bar Chart

Using this visualization, you can quickly see the relative counts of the routers from `value_counts` and can intuitively compare them in the quick bar chart. Jupyter Notebook offers plotting in the notebook when you enable it using the `matplotlib inline` command shown here. You can plot directly from a dataframe or a pandas series (that is, a single column of a dataframe). You can improve the visibility of this chart by using the horizontal option `barh`, as shown in Figure 10-8.

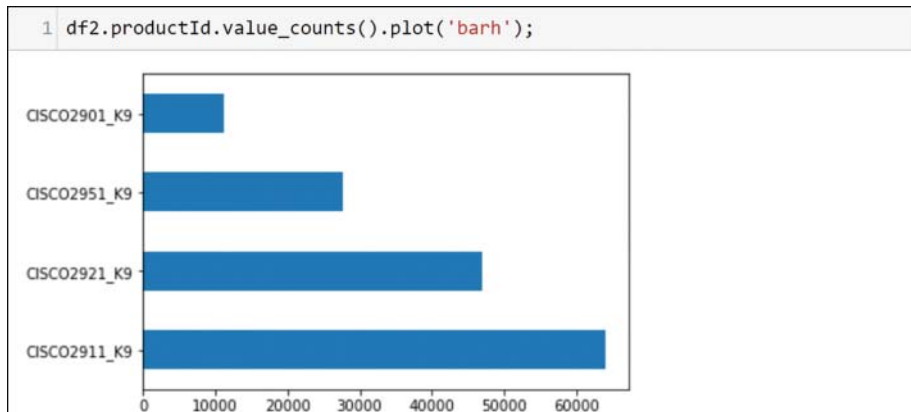


Figure 10-8 Horizontal Bar Chart

For this first analysis, you want to understand the crash rates shown with this platform. You can use **value_counts** and look at the top selections to see what crash reasons you have, as shown in Figure 10-9. Cisco extracts the crash reason data from the **show version** command for this type of router platform. You could have a column with your own labels if you are using a different mechanism to track crashes or other incidents.

1 df2.resetReason.value_counts().head(10)	
power_on	80623
reload	56016
unknown	12025
address_error_at_PC_0x9EF4DBC__address_0x7C9342D2	54
error__a_Software_forced_crash_PC_0x4E4845C	48
bus_error_at_PC_0x0__address_0x0	29
watchdog_timer_expired	27
error__a_Software_forced_crash_PC_0x4E78CA0	26
error__a_Software_forced_crash_PC_0x4ABD208	14
bus_error_at_PC_0x6F5E86FD__address_0x0	13
Name: resetReason, dtype: int64	

Figure 10-9 Router Reset Reasons

Notice that there are many different reasons for device resets, and most of them are from a **power cycle** or a **reload** command. In some cases, you do not have any data, so you see **unknown**. In order to analyze crashes, you must identify the devices that showed a crash as the last reason for resetting. Now you can examine this by using the simple string-matching capability shown in Figure 10-10.

1 df2[df2.resetReason.str.contains('error')].resetReason.value_counts()[:5]	
address_error_at_PC_0x9EF4DBC__address_0x7C9342D2	54
error__a_Software_forced_crash_PC_0x4E4845C	48
bus_error_at_PC_0x0__address_0x0	29
error__a_Software_forced_crash_PC_0x4E78CA0	26
error__a_Software_forced_crash_PC_0x4ABD208	14
Name: resetReason, dtype: int64	

Figure 10-10 Filtering a Single Dataframe Column

Here you see additional filtering inside the square brackets. Now you take the value from the dataframe column and define true or false, based on the existence of the string within that value. You have not yet done any assignment, only exploration filtering to find a method to use. This method seems to work. After iterating through **value_counts** and the possible strings, you find a set of strings that you like and can use them to filter out a new dataframe of crashes, as shown in Figure 10-11. Note that there are 1325 historical crashes identified in the 150,000 routers.

```

1 crashes=["error","watchdog","kernel","0x","abort","crash","ailure__",\
2          "generic_failure","s_w_reset","fault","reload_at_0",\
3          "reload_at_@", "reload_at$"]
4 df3=df2[df2.resetReason.str.contains('|'.join(crashes))].copy()
5 len(df3)
1325

1 '|'.join(crashes)
'error|watchdog|kernel|0x|abort|crash|ailure__|generic_failure|s_w_reset|fault|
reload_at_0|reload_at_@|reload_at$'

1 df3=df2[df2.resetReason.str.contains('|'.join(crashes))].copy()
2 len(df3)
1325

```

Figure 10-11 *Filtering a Dataframe with a List*

A few more capabilities are added for you here. All of your possible crash reason sub-strings have been collected into a list. Because pandas uses regular expression syntax for checking the strings, you can put them all together into a single string separated by a pipe character by using a Python join, as shown in the middle. The `join` command alone is used to show you what it produces. You can use this command in the string selection to find anything in your crash list. Then you can assign everything that it finds to the new dataframe `df3`.

For due diligence, check the data to ensure that you have captured the data properly, as shown in Figure 10-12, where the remaining data that did not show crashes is captured into the `df4` dataframe.

```

1 df4=df2[~df2.resetReason.str.contains('|'.join(crashes))].copy()
2 len(df4)
148675

1 df4.resetReason.value_counts()
power_on          80623
reload            56016
unknown          12025
INVALID_Switchover    10
PowerOn             1
Name: resetReason, dtype: int64

```

Figure 10-12 *Validation of Filtering Operation*

Note that the `df4` from `df2` creation command looks surprisingly similar to the previous command, where you collected the crashes into `df3`. In fact, it is the same except for one character, which is the tilde (`~`) after the first square bracket. This tilde inverts the logic

ahead of it. Therefore, you get everything where the string did *not* match. This inverts the true and false defined by the square bracket filtering. Notice that the reset reasons for the df4 do not contain anything in your crash list, and the count is in line with what you expected. Now you can add labels for crash and noncrash to your dataframes, as shown in Figure 10-13.

```

1 df3['crashed']=1
2 df4['crashed']=0

1 crash_length=len(df3)
2 print(crash_length)
3 nocrashlength=len(df4)
4 print(nocrashlength)
5 alldata_length=len(df2)
6 print(alldata_length)

1325
148675
150000

```

Figure 10-13 *Using Dataframe Length to Get Counts*

When printing the length of the crash and noncrash dataframes, notice how many crashes you assigned. Adding new columns is as easy as adding the column names and providing some assignments. This is a static value assignment, but you can add data columns in many ways. You should now validate here that you have examined all the crashes. Your first simple statistic is shown in Figure 10-14.

```

1 crashrate = float(crash_length)/float(alldata_length) * 100.0
2 print(" Percent of routers that crash is: " + str(round(crashrate,2)) + "%")

Percent of routers that crash is: 0.88%

```

Figure 10-14 *Overall Crash Rates in the Data*

Notice the base rate, which shows that fewer than 1% of routers reset on their own during their lifetime. Put on your network SME hat, and you should recognize that repeating crashes or crash reasons lost due to active upgrades or power cycles are not available for this analysis. Routers overwrite the command that you parsed for reset reasons on reload. This is the overall crash rate for routers that are known to be running the same software that they crashed with, which makes it an interesting subset for you to analyze as a sample from a larger population.

Now there are three different dataframes. You do not have to create all new dataframes at each step, but it is useful to have multiple copies as you make changes in case you want to come back to a previous step later to check your analysis. You are still in the *model*

building phase. Additional dataframes consume resources, so make sure you have the capacity to save them. In Figure 10-15, a new dataframe is assembled by concatenating the crash and noncrash dataframes and your new labels back together.

```

1 df5=pd.concat([df3,df4])
2 print("Concatenated dataframe is now this long: " + str(len(df5)))

Concatenated dataframe is now this long: 150000

1 df5.columns

Index([u'configRegister', u'productFamily', u'productId', u'productType',
      u'resetReason', u'swVersion', u'id', u'crashed'],
      dtype='object')
```

Figure 10-15 *Combining Crash and Noncrash Dataframes*

A quick look at the columns again validates that you now have a crashed column in your data. Now group your data by this new column and your productId column, as shown in Figure 10-16.

```

1 dfgroup1=df5.groupby(['productId','crashed'])
2 df6=dfgroup1.size().reset_index(name='count')
3 df6
```

	productId	crashed	count
0	CISCO2901_K9	0	11037
1	CISCO2901_K9	1	110
2	CISCO2911_K9	0	63583
3	CISCO2911_K9	1	487
4	CISCO2921_K9	0	46661
5	CISCO2921_K9	1	370
6	CISCO2951_K9	0	27394
7	CISCO2951_K9	1	358

Figure 10-16 *Dataframe Grouping of Crashes by Platform*

df6 is a dataframe made by using the **groupby** object, which pandas generates to segment groups of data. Use the **groupby** object for a summary such as the one generated here or as a method to access the groups within the original data, as shown in Figure 10-17, where the first five rows of a particular group are displayed.

```

1 dfgroup1.get_group(('CISCO2901_K9', 1))\
2 [['productId', 'resetReason', 'crashed']][:5]

```

	productId	resetReason	crashed
29369	CISCO2901_K9	processor_memory_parity_error_at_PC_0x30EC9C3C...	1
161856	CISCO2901_K9	error__a_Software_forced_crash_PC_0x30FD8834	1
194451	CISCO2901_K9	error__a_Software_forced_crash_PC_0x30E85BF4	1
242212	CISCO2901_K9	processor_memory_parity_____	1
246420	CISCO2901_K9	processor_memory_parity_error_at_PC_0x30325668...	1

Figure 10-17 Examining Individual Groups of the `groupby` Object

Based on your grouping selections of `productId` and `crashed` columns, select the **groupby** object that matches selections of interest. From that object, use the double square brackets to select specific parts of the data that you want to use to generate a new dataframe to view here. You do not generate one here (note that a new dataframe was not assigned) but instead just look at the output that it would produce.

Let's work on the dataframe made from the summary object to dig deeper into the crashes. This is a dataframe that describes what the **groupby** object produced, and it is not your original 150,000-length dataframe. There are only eight unique combinations of `crashed` and `productId`, and the **groupby** object provides a way to generate a very small data set of just these eight.

In Figure 10-18, only the crash counts are collected into a new dataframe. Take a quick look at the crash counts in a plot. The new dataframe created for visualization is only four lines long: four product IDs and their crash counts.

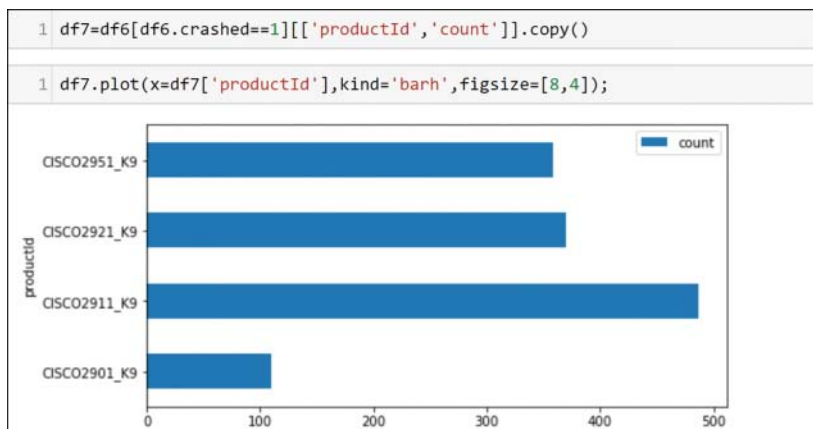


Figure 10-18 Plot of Crash Counts by Product ID

If you look at crash counts, the 2911 routers appear to crash more than the others. However, you know that there are different numbers for deployment because you looked at the base rates for deployment, so you need to consider those. If you had not explored

the base rates, you would immediately assume that the 2911 is bad because the crash counts are much higher than for other platforms. Now you can do more grouping to get some total deployment numbers for comparison of this count with the deployment numbers included. Begin this by grouping the individual platforms as shown in Figure 10-19. Recall that you had eight rows in your dataframe. When you look at productId only, there are four groups of two rows each in a **groupby** object.

```
1 dfgroup2=df6.groupby(['productId'])
2 dfgroup2.size()

productId
CISCO2901_K9    2
CISCO2911_K9    2
CISCO2921_K9    2
CISCO2951_K9    2
dtype: int64
```

Figure 10-19 *groupby Descriptive Dataframe Size*

Now that you have grouped by platform, you can use those grouped objects to get some total counts for the platforms. The use of functions with dataframes to perform this counting is introduced in Figure 10-20.

```
1 def myfun(x):
2     x['totals'] = x['count'].agg('sum')
3     return x

1 df6 = dfgroup2.apply(myfun)
2 df6
```

	productId	crashed	count	totals
0	CISCO2901_K9	0	11037	11147
1	CISCO2901_K9	1	110	11147
2	CISCO2911_K9	0	63583	64070
3	CISCO2911_K9	1	487	64070
4	CISCO2921_K9	0	46661	47031
5	CISCO2921_K9	1	370	47031
6	CISCO2951_K9	0	27394	27752
7	CISCO2951_K9	1	358	27752

Figure 10-20 *Applying Functions to Dataframe Rows*

The function **myfun** takes each **groupby** object, adds a totals column entry that sums up the values in the count column, and returns that object. When you apply this by using the **apply** method, you get a dataframe that has a totals column from the summed counts

by product family. You can use this **apply** method with any functions that you create to operate on your data.

You do not have to define the function outside and apply it this way. Python also has useful lambda functionality that you can use right in the apply method, as shown in Figure 10-21, where you generate the percentage of total for crashes versus noncrashes.

```

1 df6['rate'] = df6.apply\
2 (lambda x: round(float(x['count'])/float(x['totals']) * 100.0,2), axis=1)
3 df6

```

	productid	crashed	count	totals	rate
0	CISCO2901_K9	0	11037	11147	99.01
1	CISCO2901_K9	1	110	11147	0.99
2	CISCO2911_K9	0	63583	64070	99.24
3	CISCO2911_K9	1	487	64070	0.76
4	CISCO2921_K9	0	46661	47031	99.21
5	CISCO2921_K9	1	370	47031	0.79
6	CISCO2951_K9	0	27394	27752	98.71
7	CISCO2951_K9	1	358	27752	1.29

Figure 10-21 Using a lambda Function to Apply Crash Rate

In this command, you add the new column **rate** to your dataframe. Instead of using static assignment, you use a function to apply some transformation with values from other columns. **lambda** and **apply** allow you to do this row by row. Now you have a column that shows the rate of crash or uptime, based on deployed numbers, which is much more useful than simple counts.

You can select only the crashes and generate a dataframe to visualize the relative crash rates as shown in Figure 10-22.

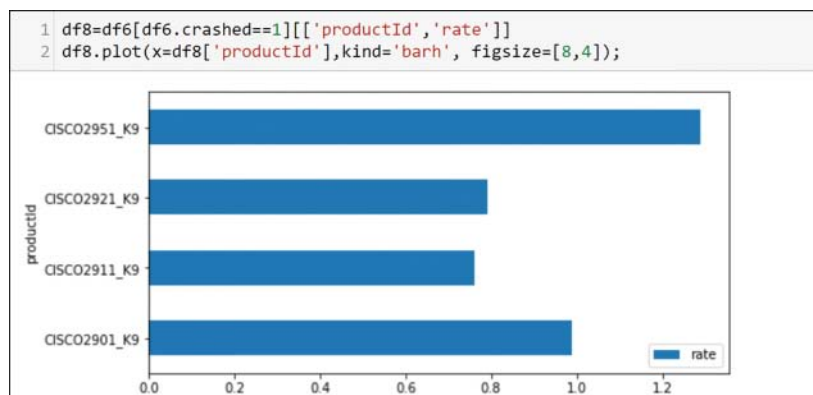


Figure 10-22 Plot of Crash Rate by Product ID

Notice that the 2911 is no longer the leader here. This leads you to want to compare the crash rates to the crash counts in a single visualization. Can you do that? Figure 10-23 shows what you get when you try that with your existing data.

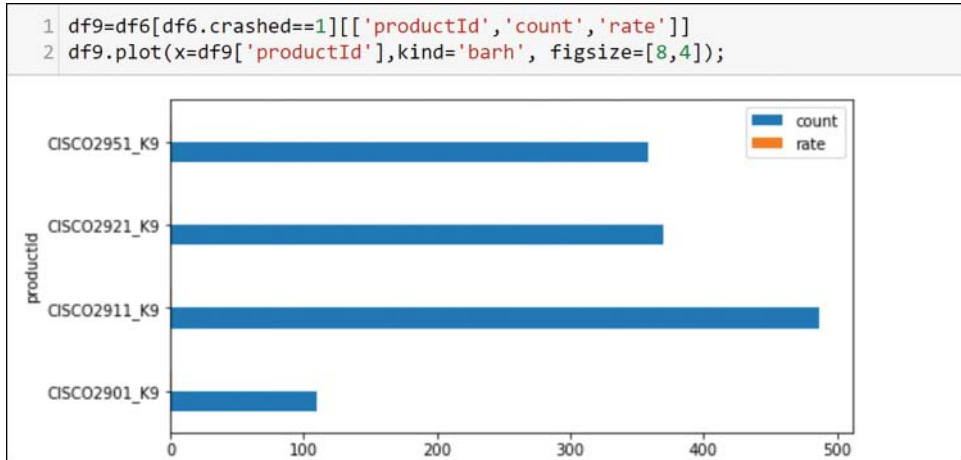


Figure 10-23 *Plotting Dissimilar Values*

What happened to your crash rates? They show in the plot legend but do not show in the plot. A quick look at a box plot of your data in Figure 10-24 reveals the answer.

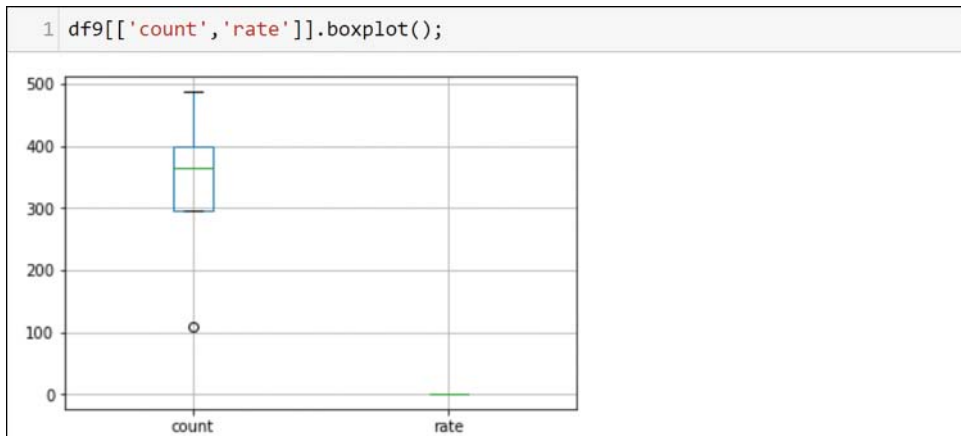


Figure 10-24 *Box Plot for Variable Comparison*

Box plots are valuable for quickly comparing numerical values in a dataframe. The box plot in Figure 10-24 clearly shows that your data is of different scales. Because you are working with linear data, you should go find a scaling function to scale the values. Then you can scale up the rate to match the count using the equation from the comments in Figure 10-25. The variables to use in the equation were assigned to make it easier to follow the addition of the new `rate_scaled` column to your dataframe.

```
1 #scaled_value = (((value_to_scale - a) * r2) / r1) + c
2 a=df9['rate'].min()
3 b=df9['rate'].max()
4 c=float(df9['count'].min())
5 d=float(df9['count'].max())
6 r1=b-a
7 r2=d-c

1 df9['rate_scaled']=(((df9['rate'] - a) * r2) / r1 + c)
```

Figure 10-25 *Scaling Data*

This creates the new `rate_scaled` column, as shown in a new box plot in Figure 10-26. Note how the min and max are aligned after applying the scaling. This is enough scaling to allow for a visualization.

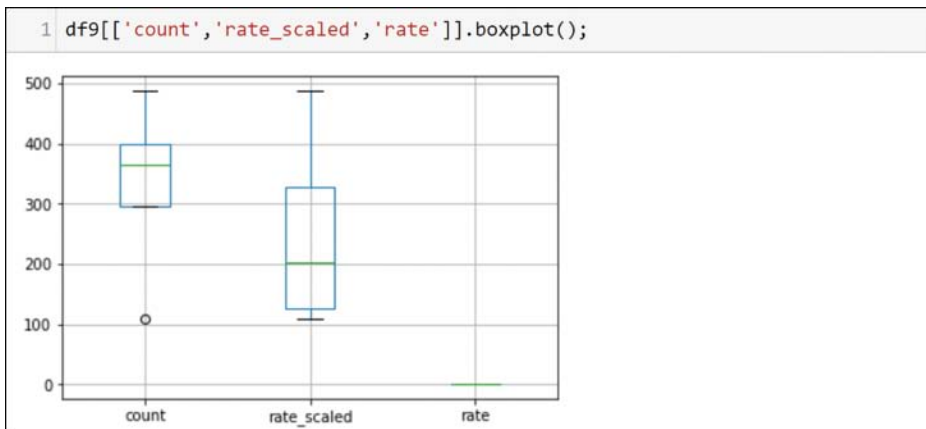


Figure 10-26 *Box Plot of Scaled Data*

Now you can provide a useful visual comparison, as shown in Figure 10-27.

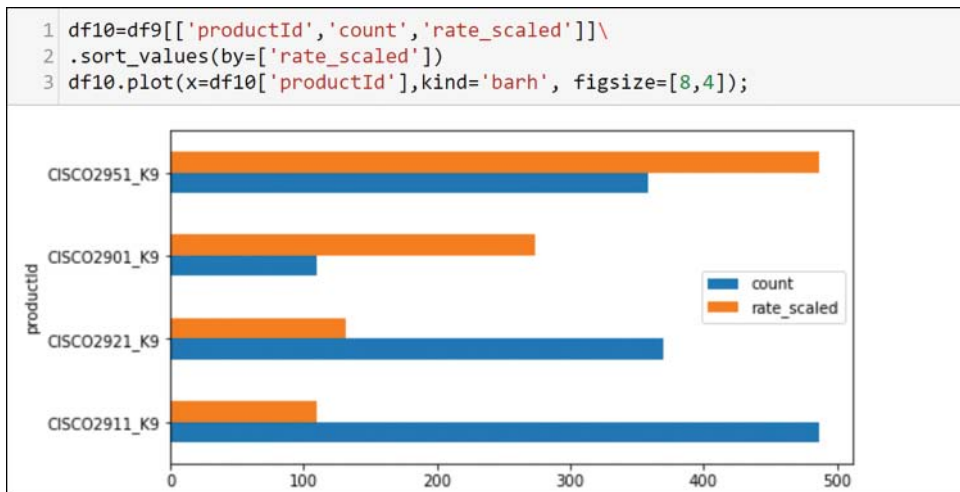


Figure 10-27 *Plot of Crash Counts and Crash Rates*

In Figure 10-27, you can clearly see that the 2911 having more crashes is a misleading number without comparison. Using the base rate for actual known crashes clearly shows that the 2911 is actually the most stable platform in terms of rate of crash. The third-ranked platform from the counts data, the 2951, actually has the highest crash rate. You can see from this example why it is important to understand base rates and how things actually manifest in your environment.

Base Rate Statistics for Software Crashes

Let's move away from the hardware and take a look at software. Figure 10-28 goes back to the dataframe before you split off to do the hardware and shows how to create a new dataframe grouped by software versions rather than hardware types.



Figure 10-28 *Grouping Dataframes by Software Version*

Notice that you have data showing both crashes and noncrashes from more than 260 versions. Versions with no known crashes are not interesting for this analysis, so you can drop them. You are only interested in examining crashes, so you can filter by crash and create a new dataframe, as shown in Figure 10-29.

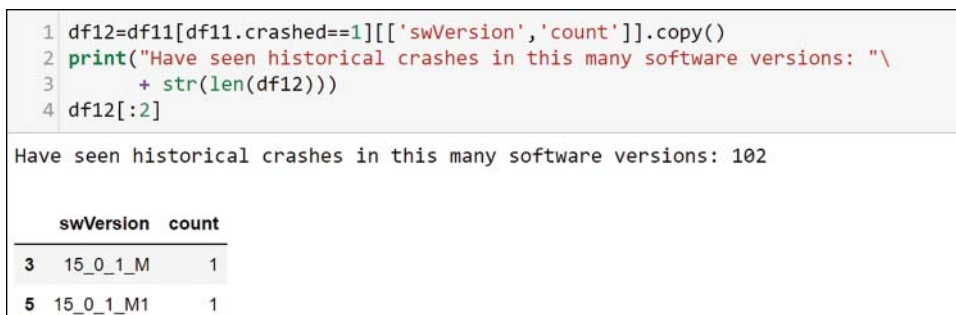


Figure 10-29 *Filtering Dataframes to Crashes Only*

The quick box plot in Figure 10-30 shows a few versions that have high crash counts. As you learned earlier in this chapter, the count may not be valuable without context.

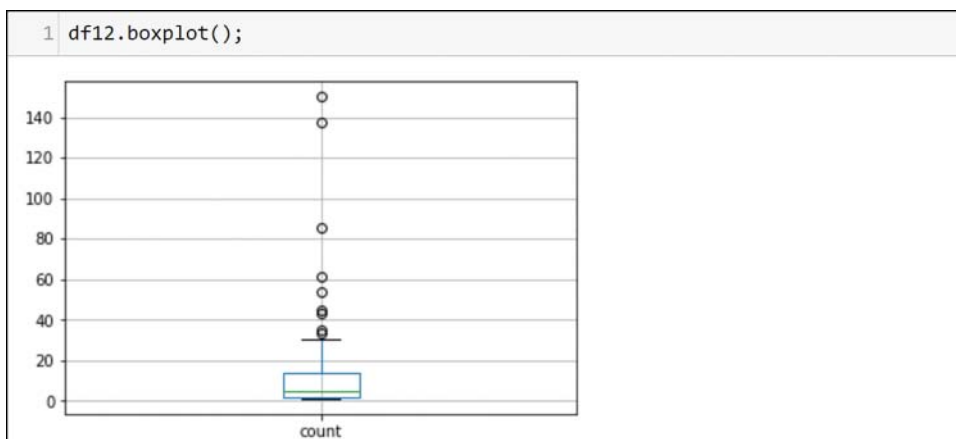


Figure 10-30 *Box Plot for Variable Evaluation*

With the box plot and your data, you do not know how many values are in the specific areas. As you work with more data, you will quickly recognize that this data has a skewed distribution when looking at the data represented in box plots. You can create a histogram as shown in Figure 10-31 to see this distribution.

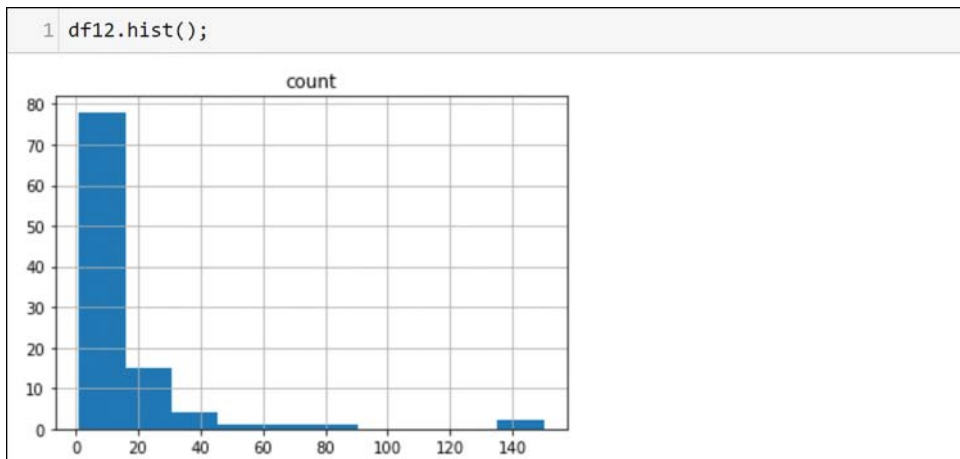


Figure 10-31 *Histogram of Skewed Right Data*

In this histogram, notice that almost 80% of your remaining 100 software versions show fewer than 20 crashes. Figure 10-32 shows a plot of the 10 highest of these counts.

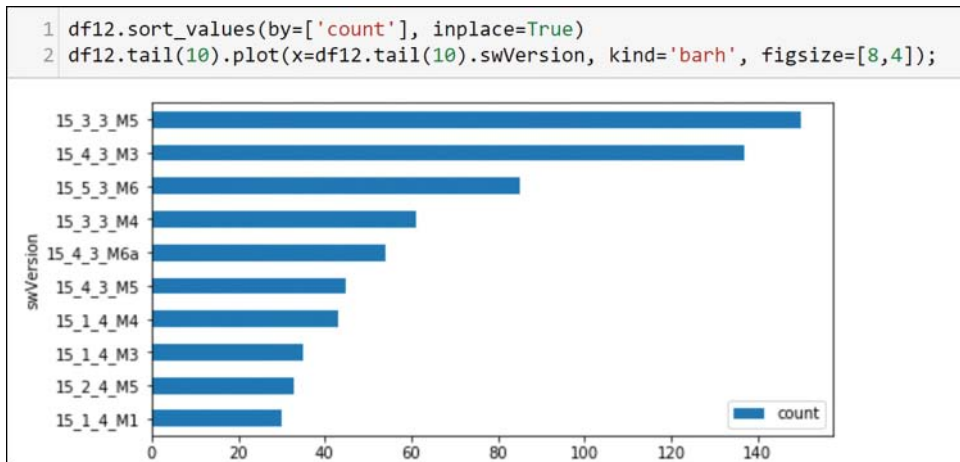


Figure 10-32 *Plot of Crash Counts by Software Version*

Comparing to the previous histogram in Figure 10-31, notice a few versions that show high crash counts and skewing of the data. You know that you also need to look at crash rate based on deployment numbers to make a valid comparison. Therefore, you should perform grouping for software and create dataframes with the right numbers for comparison, as shown in Figure 10-33. You can reuse the same method you used for the eight-row dataframe earlier in this chapter. This time, however, you group by software version.

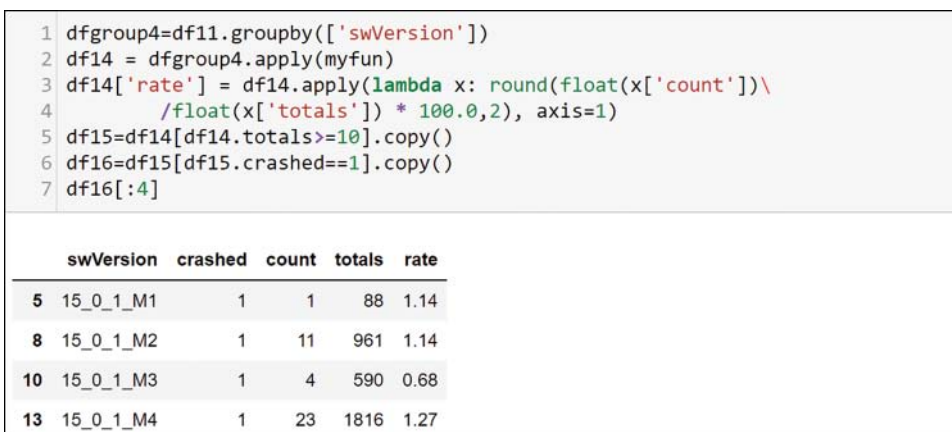


Figure 10-33 *Generating Crash Rate Data*

Note the extra filter in row 5 of the code in this section that only includes software versions with totals greater than 10. In order to avoid issues with using small numbers, you should remove any rows of data with versions of software that are on fewer than 10 routers. If you sort the rate column to the top, as you can see in Figure 10-34, you get an entirely different chart from what you saw when looking at counts only.

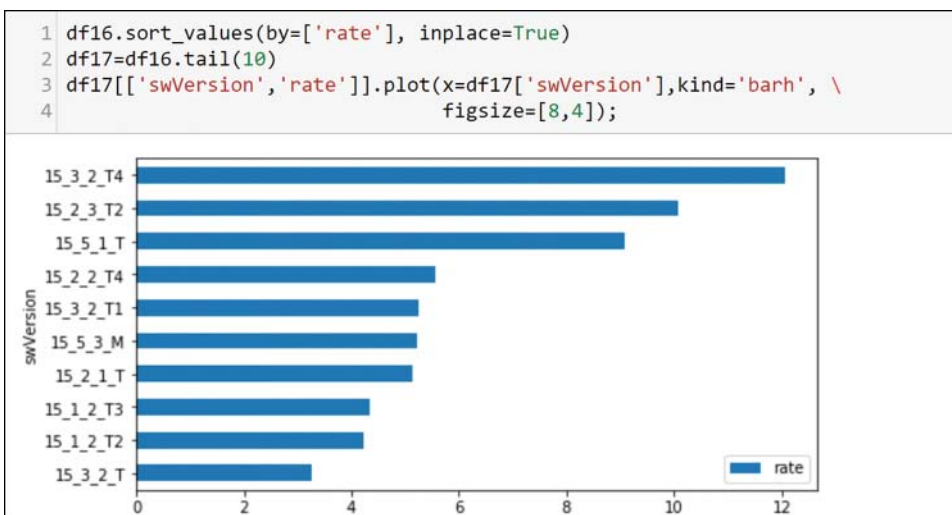


Figure 10-34 *Plot of Highest Crash Rates per Version*

In Figure 10-35 the last row in the data, which renders at the top of the plot, is showing a 12% crash rate. Because you sort the data here, you are only interested in the last one, and you use the bracketed -1 to select only the last entry.

```
1 df17[-1:]
```

	swVersion	crashed	count	totals	rate
145	15_3_2_T4	1	7	58	12.07

Figure 10-35 *Showing the Last Row from the Dataframe*

This is an older version of software, and it is deployed on only 58 devices. As an SME, you would want to investigate this version if you had it in your network. Because it is older and has low deployment numbers, it's possible that people are not choosing to use this version or are moving off it.

Now let's try to look at crash rate and counts. You learned that you must first scale the data into a new column, as shown in Figure 10-36.

```
1 a=df16['rate'].min()
2 b=df16['rate'].max()
3 c=float(df16['count'].min())
4 d=float(df16['count'].max())
5 r1=b-a
6 r2=d-c
7 df16['rate_scaled']=((df16['rate'] - a) * r2) / r1 + c)
```

Figure 10-36 *Scaling Up the Crash Rate*

Once you have scaled the data, you can visualize it, as shown in Figure 10-37. Do not try too hard to read this visualization. This diagram is intentionally illegible to make a case for filtering your data before visualizing it. As an SME, you need to choose what you want to show.

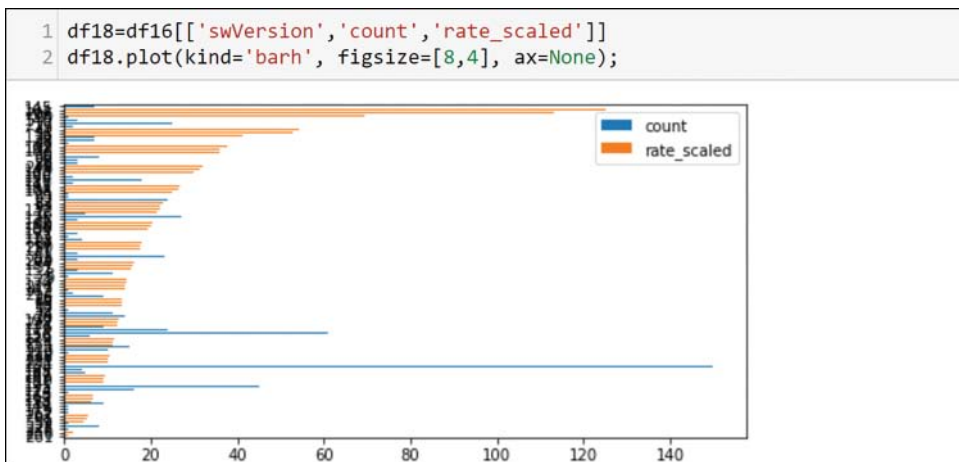


Figure 10-37 *Displaying Too Many Variables on a Visualization*

This chart includes all your crash data, is sorted by crash rate descending, and shows the challenges you will face with visualizing so much data. The scaled crash rates are at the top, and the high counts are at the bottom. It is not easy to make sense of this data. Your options for what to do here are use-case specific. What questions are you trying to answer with this particular analysis?

One thing you can do is to filter to a version of interest. For example, in Figure 10-38, look at the version that shows at the top of the high counts table.

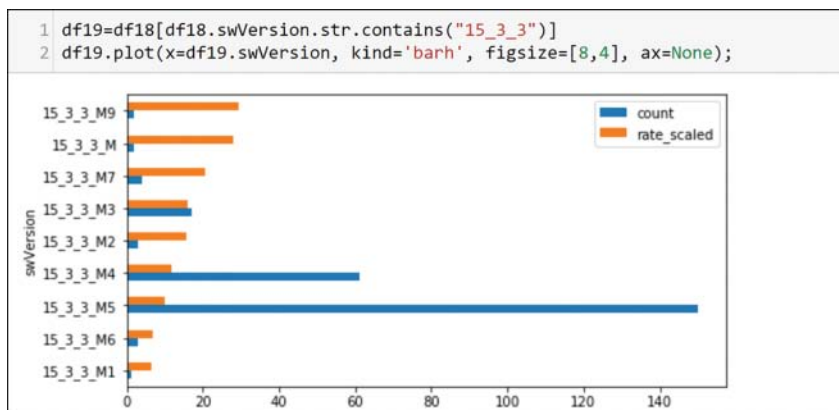


Figure 10-38 *Plot Filtered to a Single Software Version*

Notice that the version with the highest counts is near the bottom, and it is not that bad. It was much worse in the versions that showed the highest crash count. However, it actually has the third best crash rate within its own software train. This is not a bad version.

If you back off the regex filter to include the version that showed the highest crash rate in the same chart in Figure 10-39, see that some versions of the 15_3 family have significantly lower crash rates than do other versions.

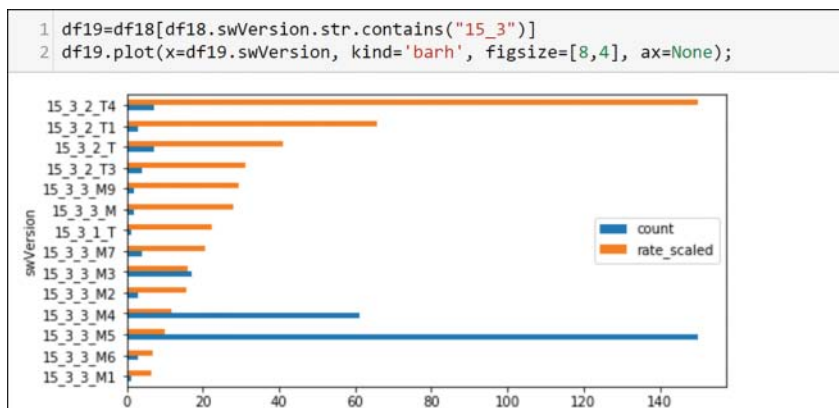


Figure 10-39 *Plot Filtered to Major Version*

You can be very selective with the data you pull so that you can tell the story you need to tell. Perhaps you want to know about software that is very widely deployed, and you want to compare that crash rate to the high crash rate seen with the earlier version, 15_3_2_T4. You can use dataframe OR logic with a pipe character to filter, as shown in Figure 10-40.

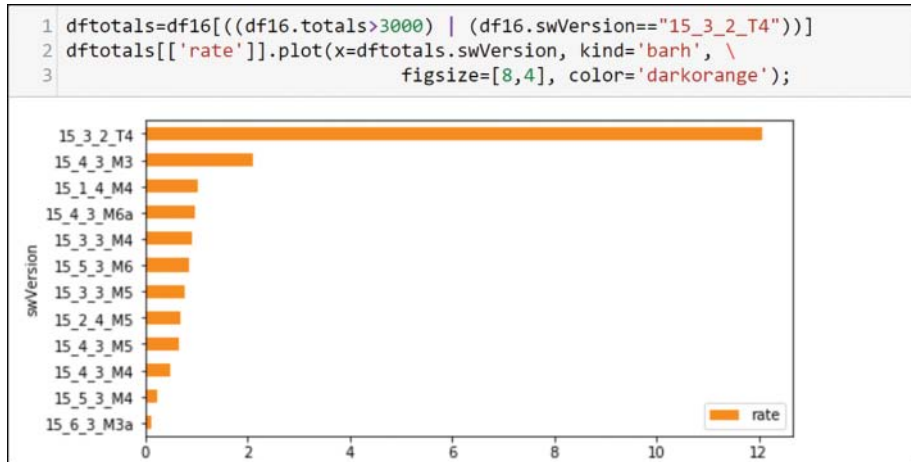


Figure 10-40 *Combining a Mixed Filter on the Same Plot*

In the filter for this plot, you add the pipe character and wrap the selections in parentheses to give a choice of highly deployed code or the high crash rate code seen earlier. This puts it all in the same plot for valid comparison. All of the highly deployed codes are much less risky in terms of crash rate compared to the 15_3_2_T4. You now have usable insight about the software version data that you collect. You can examine the stability of software in your environment by using this technique.

ANOVA

Let's shift away from the 2900 Series data set in order to go further into statistical use cases. This section examines analysis of variance (ANOVA) methods that you can use to explore comparisons across software versions. Recall that ANOVA provides statistical analysis of variance and seeks to show significant differences between means in different groups. If you use your intuition to match this to mean crash rates, this method should have value for comparing crash rates across software versions. That is good information to have when selecting software versions for your devices.

In this section you will use the same data set to see what you get and dig into the 15_X train that bubbled up in the last section. Start by selecting any Cisco devices with software version 15, as shown in Figure 10-41. Note that you need to go all the way back to your original dataframe df to make this selection.

```

1 #anova get groups
2 df1515=df[(df.swVersion.str.startswith("15_") & \
3           (df.productFamily.str.contains("Cisco")))].copy()
4 df1515['ver']=df1515.apply(lambda x: x['swVersion'][:4], axis=1)
5 df1515[:2]

```

	configRegister	productFamily	productId	productType	resetReason	swVe
0	02102	Cisco_VG_Series_Gateways	VG224	Unified_Communications	power_on	15_0_
1	02102	Cisco_VG_Series_Gateways	VG224	Unified_Communications	power_on	15_0_

Figure 10-41 *Filtering Out New Data for Analysis*

You use the ampersand (&) here as a logical **AND**. This method causes your square bracket selection to look for two conditions for filtering before you make your new dataframe copy. For grouping the software versions, create a new column and use a **lambda** function to fill it with just the first four characters from the swVersion column. Check the numbers in Figure 10-42.

```

1 df1515.ver.value_counts()

```

15_0	201311
15_2	176036
15_4	114846
15_1	102837
15_3	92256
15_5	72583
15_6	34075
15_7	209

Name: ver, dtype: int64

```

1 df1515_2=df1515[~(df1515.ver=="15_7")].copy()
2 len(df1515_2)

```

793944

Figure 10-42 *Exploring and Filtering the Data*

Notice that there is a very small sample size for the 15_7 version, so you can remove it by copying everything else into a new dataframe. This will still be five times larger than the last set of data that was just 2900 Series routers. This data set is close to 800,000 records, so the methods used previously for marking crashes work again, and you perform them as shown in Figure 10-43.

```

1 c15=df1515_2[df1515_2.resetReason.str.contains('|'.join(crashes))].copy()
2 nc15=df1515_2[~df1515_2.resetReason.str.contains('|'.join(crashes))].copy()
3 print(str(len(c15)) + " crashes in " + str(len(df1515_2)) + " records")
4 c15['crashed']=1
5 nc15['crashed']=0
6 df1515_3=pd.concat([c15,nc15])

```

6883 crashes in 793944 records

Figure 10-43 *Labeling Known Crashes in the Data*

Once the dataframes are marked for crashes and concatenated, you can summarize, count, and group the data for statistical analysis. Figure 10-44 shows how to use the **groupby** command and totals function again to build a new counts dataframe for your upcoming ANOVA work.

```

1 dfgroup5=df1515_3.groupby(['ver','productFamily','crashed'])
2 adf=dfgroup5.size().reset_index(name='count')
3 dfgroup6=adf.groupby(['productFamily','ver'])
4 adf2 = dfgroup6.apply(myfun)
5 adf2[0:2]

```

	ver	productFamily	crashed	count	totals
0	15_0	Cisco_1800_Series_Integrated_Services_Routers	0	624	626
1	15_0	Cisco_1800_Series_Integrated_Services_Routers	1	2	626

Figure 10-44 *Grouping by Two and Three Columns in a Dataframe*

This data set is granular enough to do statistical analysis down to the platform level by grouping the software by productFamily. You should focus on the major version for this analysis, but you may want to take it further and explore down to the platform level in your analysis of data from your own environment. Figure 10-45 shows that you clean the data a bit by dropping platform totals that are less than one-tenth of 1% of your data. Because you are doing statistical analysis on generalized data, you want to remove the risk of small-number math influencing your results.

```

1 print(len(adf2))
2 low_cutoff=len(df1515_3) * .0001
3 adf3=adf2[adf2.totals>int(low_cutoff)].copy()
4 print(len(adf3))
5 adf3['rate'] = adf3.apply\
6 (lambda x: round(float(x['count'])/float(x['totals']) * 100.0,2), axis=1)
7 adf3[0:3]

```

336
244

	ver	productFamily	crashed	count	totals	rate
0	15_0	Cisco_1800_Series_Integrated_Services_Routers	0	624	626	99.68
1	15_0	Cisco_1800_Series_Integrated_Services_Routers	1	2	626	0.32
2	15_0	Cisco_1900_Series_Integrated_Services_Routers	0	1888	1896	99.58

Figure 10-45 *Dropping Outliers and Adding Rates*

Now that you have the rates, you can grab only the crash rates and leave the noncrash rates behind. Therefore, you should drop the crashed and count columns because you do not need them for your upcoming analysis. You can look at what you have left by using the `describe` function, as shown in Figure 10-46.

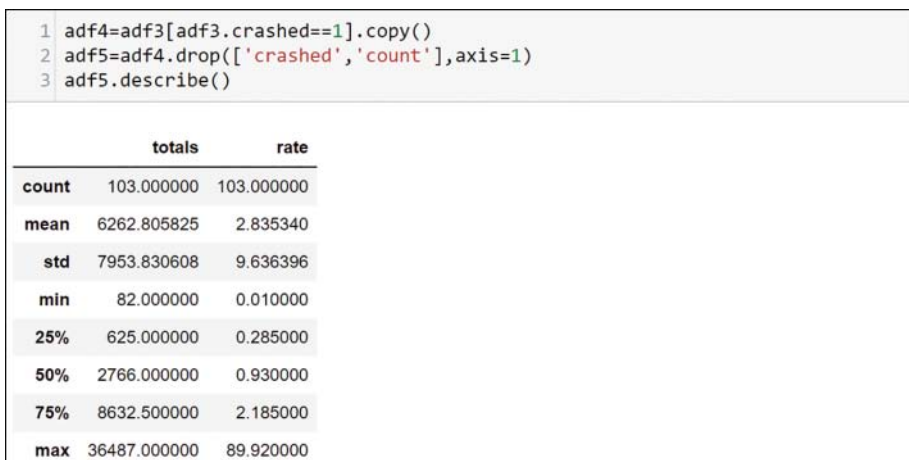


Figure 10-46 Using the `pandas describe` Function to Explore Data

`describe` provides you with numerical summaries of numerical columns in the dataframe.

The max rate of 89% and standard deviation of 9.636 should immediately jump out at you. Because you are going to be doing statistical tests, this clear outlier at 89% is going to skew your results. You can use a histogram to take a quick look at how it fits with all the other rates, as shown in Figure 10-47.

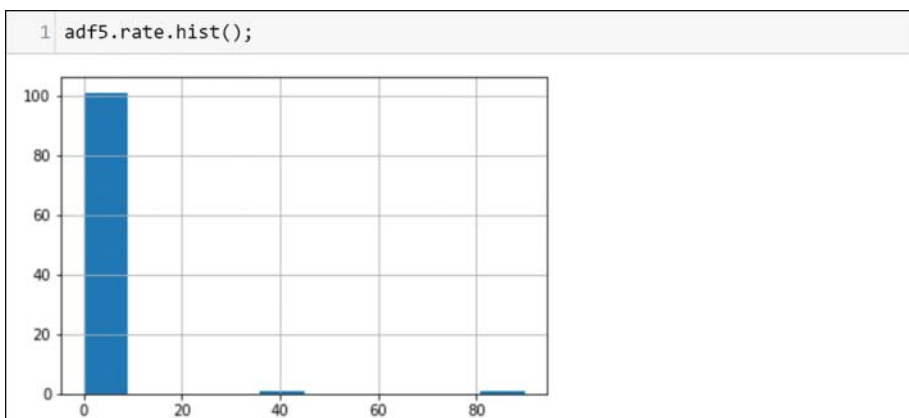


Figure 10-47 Histogram to Show Outliers

This histogram clearly shows that there are at least two possible outliers in terms of crash rates. Statistical analysis can be sensitive to outliers, so you want to remove them. This is accomplished in Figure 10-48 with another simple filter.

```
1 adf5[adf5.rate>25.0]
```

	ver	productFamily	totals	rate
237	15_3	Cisco_Aironet_1140_Series	90	40.00
297	15_5	Cisco_5900_Series_Embedded_Services_Routers	248	89.92

```
1 adf5.drop([237,297], inplace=True)
```

Figure 10-48 Filter Generated from Histogram Viewing

For the **drop** command, axis zero is the default, so this command drops rows. The first thing that comes to mind is that you have two versions that you should probably go take a closer look at to ensure that the data is correct. (It is not—see note below.) If these were versions and platforms that you were interested in learning more about in this analysis, your task would now be to validate the data to see if these versions are very bad for those platforms. In this case, they are not platforms of interest, so you can just remove them by using the **drop** command and the index rows. You can capture them as findings as is.

Note The 5900 routers shown in Figure 10-48 actually have no real crashes. The reset reason filter used to label crashes picked up a non-traditional reset reason for this platform. It is left in the data here to show you what highly skewed outliers can look like. Recall that you should always validate your findings using SME analysis.

The new histogram shown in Figure 10-49, without the outliers, is more like what you expected.

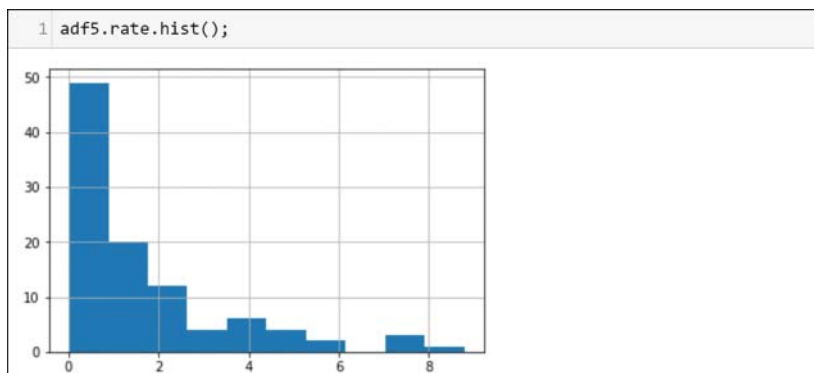


Figure 10-49 Histogram with Outliers Removed

This histogram is closer to what you expected to see after dropping the outliers, but it is not very useful for ANOVA. Recall that you must investigate the assumptions for proper use of the algorithms. One assumption of ANOVA is that the outputs are normally distributed. Notice that your data is skewed to the right. *Right skewed* means the tail of the distribution is longer on the right; *left skewed* would have a longer tail on the left. What can you do with this? You have to transform this to something that resembles a normal distribution.

Data Transformation

If you want to use something that requires a normal distribution of data, you need to use a transformation to make your data look somewhat normal. You can try some of the common *ladder of powers* methods to explore the available transformations. Make a copy of your dataframe to use for testing, create the proper math for applying the ladder of power transforms as functions, and apply them all as shown in Figure 10-50.

```
1 tt=adf5[['ver', 'rate']].copy()
2 tt['rate2']=tt.apply(lambda x: int(x['rate']*100), axis=1)
3 tt['cube']=tt.apply(lambda x: int(x['rate2'])**3, axis=1)
4 tt['square']=tt.apply(lambda x: int(x['rate2']**2, axis=1)
5 tt['sqrt']=tt.apply(lambda x: np.sqrt(x['rate2']), axis=1)
6 tt['cuberoot']=tt.apply(lambda x: np.cbrt(x['rate2']), axis=1)
7 tt['log']=tt.apply(lambda x: np.log(x['rate2']), axis=1)
8 tt['invnegsq']=tt.apply(lambda x: -(1.0/np.sqrt(x['rate2'])), axis=1)
9 tt['recip']=tt.apply(lambda x: 1.0/x['rate2'], axis=1)
10 del tt['rate']
11 tt[tt.ver=="15_0"].hist(grid=False,xlabelsize=0,ylabelsize=0);
```

Figure 10-50 *Ladder of Powers Implemented in Python*

It's a good idea to investigate the best methods behind each possible transformation and be selective about the ones you try. Let's start with some sample data and testing. Note that in line 2 in Figure 10-50, testing turned up that you needed to scale up the rate to an integer value from the percentages that you were using. They were multiplied by 100 and converted to integer values. There are many ways to transform data. For line 11 in the code, you might choose a quick visual inspection, as shown in Figure 10-51.



Figure 10-51 *Histogram of All Dataframe Columns*

Tests for Normality

None of the plots from the previous section have a nice clean transformation to a normal bell curve distribution, but a few of them appear to be possible candidates. Fortunately, you do not have to rely on visual inspection alone. There are statistical tests you can run to determine if the data is normally distributed. The Shapiro–Wilk test is one of many available tests for this purpose. Figure 10-52 shows a small loop written in Python to apply the Shapiro–Wilk test to all the transformations in the test dataframe.

```
1 import scipy.stats as stats
2 for x in tt.columns:
3     try:
4         print(str(x) + " " + str(stats.shapiro(tt[x])))
5     except:
6         pass
```

rate2 (0.775720477104187, 4.1034533410089225e-11)
cube (0.3721432685852051, 8.055603343887878e-19)
square (0.5164250731468201, 1.181204650302863e-16)
sqrt (0.9359350800514221, 0.00010134831973118708)
cuberoot (0.973645806312561, 0.04057560861110687)
log (0.9695112705230713, 0.019343357533216476)
invnegsq (0.6864817142486572, 2.2394713579225456e-13)
recip (0.3706700801849365, 7.691381761507943e-19)

Figure 10-52 *Shapiro–Wilk Test for Normality*

The goal with this test is to have a W statistic (first entry) near 1.0 and a p -value (second entry) that is greater than 0.05. In this example, you do not have that 0.05, but you have something close with the cube root at 0.04. You can use that cube root transformation to see how the analysis progresses. You can come back later and get more creative with transformations if necessary. One benefit to being an SME is that you can build models and validate your findings using both data science and your SME knowledge. You know that the values you are using are borderline acceptable from a data science perspective, so you need to make sure the SME side is extra careful about evaluating the results.

A quantile–quantile (Q–Q) plot is another mechanism for examining the normality of the distribution. In Figure 10-53 notice what the scaled-up rate variable looks like in this plot. Be sure to import the required libraries first by using the following:

- `import statsmodels.api as sm`
- `import pylab`

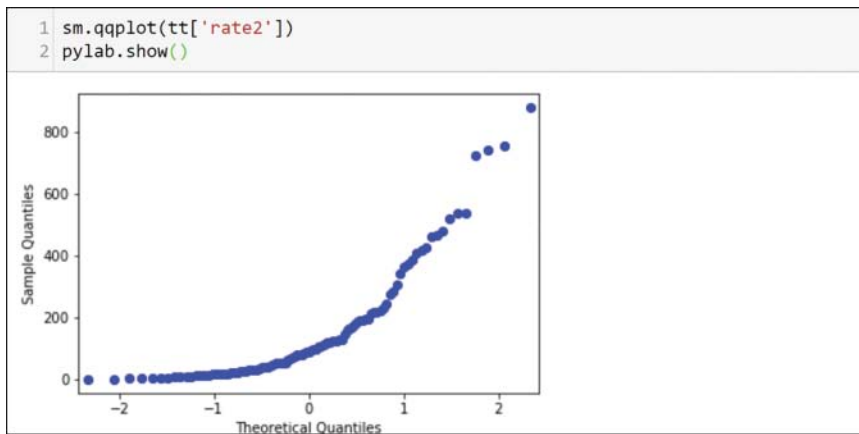


Figure 10-53 *Q–Q Plot of a Non-normal Value*

Q–Q plots for normal data show the data generally in a straight line on the diagonal of the plot. You can clearly see in Figure 10-53 that the untransformed **rate2** variable is not straight. After the cube root transformation, things look much better, as shown in Figure 10-54.

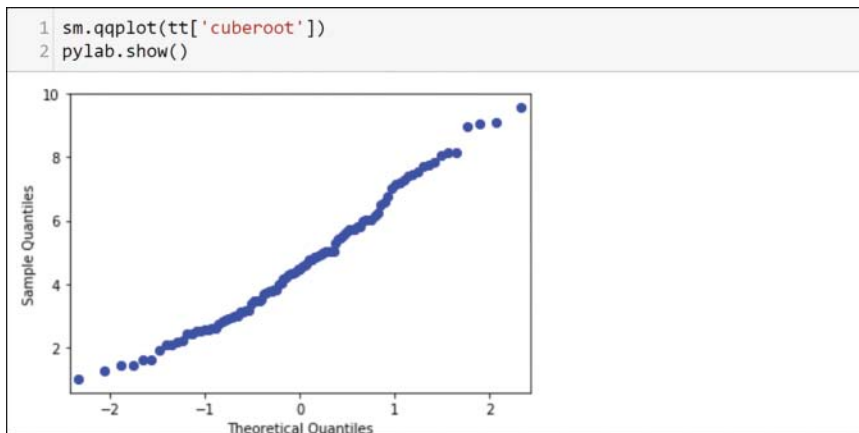


Figure 10-54 *Q–Q Plot Very Close to Normal*

Note If you have Jupyter Notebook set up and are working along as you read, try the other transformations in the Q–Q plot to see what you get.

Now that you have the transformations you want to go with, you can copy them back into the dataframe by adding new columns, using the methods shown in Figure 10-55.

```

1 adf5['rate2']=adf5.apply(lambda x: int(x['rate']*100), axis=1)
2 adf5['cuberoot']=adf5.apply(lambda x: np.cbrt(x['rate2']), axis=1)
3 groups = pd.unique(adf5.ver.values)
4 group_dict = {x:adf5['cuberoot'][adf5.ver == x] for x in groups}

```

Figure 10-55 *Applying Transformations and Adding Them to the Dataframe*

You also create some groups here so that you have lists of the values to use for the ANOVA work. **groups** is an array of the unique values of the version in your dataframe, and **group_dict** is a Python dictionary of all the cube roots for each of the platform families that comprise each group. This dictionary is a convenient way to have all of the grouped data points together so you can look at additional statistical tests.

Examining Variance

Another important assumption of ANOVA is homogeneity of variance within the groups. This is also called *homoscedasticity*. You can see the variance of each of the groups by selecting them from the dictionary grouping you just created, as shown in Figure 10-56, and using the numpy package (**np**) to get the variance.

```

1 print(np.var(group_dict["15_0"], ddof=1))
2 print(np.var(group_dict["15_1"], ddof=1))
3 print(np.var(group_dict["15_2"], ddof=1))
4 print(np.var(group_dict["15_3"], ddof=1))
5 print(np.var(group_dict["15_4"], ddof=1))
6 print(np.var(group_dict["15_5"], ddof=1))
7 print(np.var(group_dict["15_6"], ddof=1))

```

4.45736423134
2.22699069168
3.86833433381
4.4021137927
6.46296678415
1.98614307337
1.78938625963

Figure 10-56 *Checking the Variance of Groups*

As you see, these variances are clearly not the same. ANOVA sometimes works with up to a fourfold difference in variance, but you will not know the impact until you do some statistical examination. As you will learn, there is a test for almost every situation in statistics; there are multiple tests available to examine variance. Levene's test is used here. You can examine some the variances you already know to see if they are statistically different, as shown in Figure 10-57.

```

1 print(stats.levene(group_dict["15_4"],group_dict["15_6"]))
2 print(stats.levene(group_dict["15_0"],group_dict["15_3"]))
3 print(stats.levene(group_dict["15_4"],group_dict["15_5"]))

```

LeveneResult(statistic=1.2396849178442981, pvalue=0.28569566537497904)
 LeveneResult(statistic=0.0010369699862904005, pvalue=0.97448882445880869)
 LeveneResult(statistic=1.0844227481331332, pvalue=0.31823696677101121)

Figure 10-57 *Levene's Test for Equal Variance*

Here you check some variances that you know are close and some that you know are different. You want to find out if they are significantly different enough to impact the ANOVA analysis. The value of interest is the p -value. If p -value is greater than 0.05, you can assume that the variances are equal. You know from Figure 10-56 that variances of 15_0 and 15_3 are very close. Note that the other variance p -value results, even for the worst variance differences from Figure 10-56, are still higher than 0.05. This means you cannot reject that you have equal variance in the groups. You can assume that you do have statistically equal variance. You should be able to rely on your results of ANOVA because you have statistically met the assumption of equal variance. You can view the results of your first one-way ANOVA in Figure 10-58.

```

1 from scipy import stats
2 f, p = stats.f_oneway(group_dict["15_4"], group_dict["15_5"])
3 print("F is " + str(f) + " with a p-value of " + str(p))

```

F is 0.001033098679 with a p-value of 0.974887349769

```

1 from scipy import stats
2 f, p = stats.f_oneway(group_dict["15_1"], group_dict["15_6"])
3 print("F is " + str(f) + " with a p-value of " + str(p))

```

F is 12.397660866 with a p-value of 0.00192445781717

Figure 10-58 *One-Way ANOVA Results*

For both of the group pairs where you know the variance to be close, notice that there appears to be significant evidence to reject that 15_4 and 15_5 are any different from each other because the p -value is well over a threshold of 0.05. They are not statistically different. You seek a high F -statistic and a p -value under 0.05 here to find something that may be statistically different.

Conversely, you cannot reject that 15_1 and 15_6 are different because the p -value is well under the 0.05 threshold. It appears that 15_1 and 15_6 may be statistically different. You can create a loop to run through all combinations that include either of these, as shown in Figure 10-59.


```

1 from scipy import stats
2 myrecords=[]
3 done=[]
4 for k in group_dict.keys():
5     done.append(k)
6     for kk in group_dict.keys():
7         if ((kk not in done) & (k!=kk)):
8             f, p = stats.f_oneway(group_dict[k], group_dict[kk])
9             if k in ("15_1|15_6"):
10                 print(str(kk) + "<->" + str(k) + ". F is " \
11                     + str(f) + " with a p-value of " + str(p))
12                 myrecords.append((k,kk,f,p))

15_0<->15_1. F is 11.6196598365 with a p-value of 0.00140664887696
15_6<->15_1. F is 12.397660866 with a p-value of 0.00192445781717
15_5<->15_1. F is 1.2137441349 with a p-value of 0.28305916744
15_4<->15_1. F is 1.00958108066 with a p-value of 0.325457778895
15_5<->15_6. F is 3.99927026866 with a p-value of 0.0708260899879
15_4<->15_6. F is 1.9192348957 with a p-value of 0.189251632739

```

Figure 10-59 *Pairwise One-Way ANOVA in a Python Loop*

In this loop, you can run through every pairwise combination with ANOVA and identify the F -statistic and p -value for each of them. At the end, you gather them into a list of four value tuples. You want a high F -statistic and a low p -value, under 0.05, for significant findings. You can see that the 15_0 and 15_1 appear to have significantly different mean crash rates. You cannot reject that either of these are different. However, you can reject that many others are different. You can filter all the results to only those with p -values below the 0.05 threshold, as shown in Figure 10-60.

```

1 topn=sorted(myrecords, key=lambda x: x[3])
2 sig_topn=[match for match in topn if match[3]<0.05]
3 sig_topn

[('15_1', '15_0', 11.619659836489873, 0.0014066488769551085),
 ('15_1', '15_6', 12.397660865957176, 0.0019244578171699646),
 ('15_2', '15_1', 4.2703283982942661, 0.045299878433185101),
 ('15_3', '15_1', 4.2756461354347222, 0.049605465427939671)]

```

Figure 10-60 *Statistically Significant Differences from ANOVA*

Now you can sort your records on the third value in the tuple you collected and then select the records that have interesting values in a new `sig_topn` list. Of these four, two have very low p -values, and two others are statistically valid. Now it is time to do some model validation. The first and most common method is to use your box plots on the cube root data that you were using for the analysis. Figure 10-61 shows how to use that box plot to evaluate whether these findings are valid, based on visual inspection of scaled data.

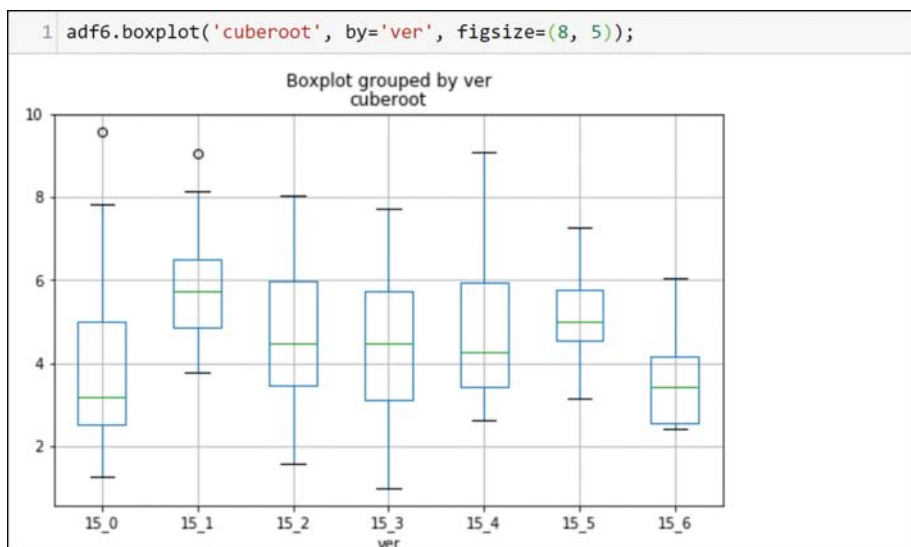


Figure 10-61 *Box Plot Comparison of All Versions*

Using this visual examination, do the box plot pairs of 15_0 versus 15_1, and 15_1 versus 15_6 look significantly different to you? They are clearly different, so you can trust that your analysis found that there is a statistically significant difference in crash rates.

You may be wondering why a visual examination wasn't performed in the first place. When you are building models, you can use many of the visuals that were used in this chapter to examine results along the way to validate that the models are working. You would even build these visuals into tools that you can periodically check. However, the real goal is to build something fully automated that you can put into production. You can use statistics and thresholds to evaluate the data in automated tools. You can use visuals as much as you want, and how much you use them will depend on where you sit on the analytics maturity curve. Full preemptive capability means that you do not have to view the visuals to have your system take action. You can develop an analysis that makes programmatic decisions based on the statistical findings of your solution.

There are a few more validations to do before you are finished. You saw earlier that there is statistical significance between a few of these versions, and you validated this visually by using box plots. You can take this same data from `adf6` and put it into Excel and run ANOVA (see Figure 10-62).

	A	B	C	D	E	F	G
2							
3	SUMMARY						
4	Groups	Count	Sum	Average	Variance		
5	15_0	29	113.814179	3.92462686	4.457364231		
6	15_1	17	100.51957	5.91291587	2.226990692		
7	15_2	25	118.71741	4.74869641	3.868334334		
8	15_3	9	40.0353492	4.44837213	4.402113793		
9	15_4	8	40.8408654	5.10510817	6.462966784		
10	15_5	6	30.8539714	5.14232857	1.986143073		
11	15_6	7	25.3210471	3.61729245	1.78938626		
12							
13							
14	ANOVA						
15	Source of Variation	SS	df	MS	F	P-value	F crit
16	Between Groups	53.55716873	6	8.92619479	2.367538708	0.0356778	2.19660151
17	Within Groups	354.4027843	94	3.77024239			
18							
19	Total	407.959953	100				

Figure 10-62 Example of ANOVA in Excel

The p -value here is less than 0.05, so you can reject the null hypothesis that the groups have the same crash rate. However, you saw actual pairwise differences when working with the data and doing *pairwise* ANOVA. Excel looks across *all* groups here. When you added your SME evaluation of the numbers and the box plots, you noted that there are some differences that can be meaningful. This is why you are uniquely suited to find use cases in this type of data. Generalized evaluations using packaged tools may not provide the level of granularity needed to uncover the true details. This analysis in Excel tells you that there is a difference, but it does not show any real standout when looking at all the groups compared together. It is up to you to split the data and analyze it differently. You could do this in Excel.

There is a final validation common for ANOVA, called *post-hoc testing*. There are many tests available. One such test, from Tukey, is used in Figure 10-63 to validate that the results you are seeing are statistically significant.

```
1 adf6=adf5[adf5.ver.str.contains("15_0|15_1|15_6|15_2")].copy()
```

```
1 from statsmodels.stats.multicomp import pairwise_tukeyhsd
2 print pairwise_tukeyhsd(adf6["cuberoor"], adf6["ver"])
3
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

```
=====
group1 group2 meandiff  lower  upper  reject
-----
15_0    15_1    1.9883    0.4717 3.5048  True
15_0    15_2    0.8241   -0.5309 2.179  False
15_0    15_6   -0.3073   -2.3981 1.7834  False
15_1    15_2   -1.1642   -2.725  0.3965  False
15_1    15_6   -2.2956   -4.5253 -0.066  True
15_2    15_6   -1.1314   -3.2544 0.9916  False
=====
```

Figure 10-63 Tukey HSD Post-Hoc test for ANOVA

Here you filter down to the top four version groups that show up in your results. Then you run the test to see if the result is significant enough to reject that the groups are the same. You have now validated through statistics, visualization, and post-hoc testing that there are differences in these versions. You have seen from visualization (you can visually or programmatically compare means) that version 15_0 and 15_6 are both exhibiting lower crash rates than version 15_1, given this data.

Where do you go from here? Consider how you could operationalize this statistics solution into a use case. If you package up your analysis into an ongoing system of automated application on real data from your environment, you can collect any of the values over time and examine trends over time. You can have constant evaluation of any numeric parameter that you need to use to evaluate the choice of one group over another. This example just happens to involve crashes in software. You can use these methods on other data.

Remember now to practice inverse thinking and challenge your own findings. What are you *not* looking at? Here are just a few of many possible trains of thought:

- These are the latest deployed routers, captured as a snapshot in time, and you do not have any historical data. You do not have performance or configuration data either.
- On reload of a router, you lose crash information from the past unless you have historical snapshots to draw from.
- You would not show crashes for routers that you upgraded or manually reloaded. For such routers, you might see the last reset as reload, unknown, or power-on.
- The dominant crashes at the top of your data could be attempts to fix bad configurations or bad platform choices with software upgrades.
- There may or may not be significant load on devices, and the hope may be that a software upgrade will help them perform better.
- There may be improperly configured devices.
- There are generally more features in newer versions, which increases risk.
- You may have mislabeled some data, as in the case of the 5900 routers.

There are many directions you could take from here to determine *why* you see what you see. Remember that correlation is not causation. Running newer software like 15_1 over 15_0 does not cause devices to crash. Use your SME skills to find out what the crashes are all about.

Statistical Anomaly Detection

This chapter closes with some quick anomaly detection on the data you have. Figure 10-64 shows some additional columns in the dataframe to define outlier thresholds. You could examine the mean or median values here, but in this case, let's choose the mean value.

```

1 adf6['std']=np.std(adf6['cuberoot'])
2 adf6['mean']=np.mean(adf6['cuberoot'])

1 adf6['highside']=adf6.apply(lambda x: x['mean'] + (2*x['std']),axis=1)
2 adf6['lowside']=adf6.apply(lambda x: x['mean'] - (2*x['std']),axis=1)

```

Figure 10-64 *Creating Outlier Boundaries*

Because you have your cube root data close to a normal distribution, it is valid to use that to identify statistical anomalies that are on the low or high side of the distribution. In a normal distribution, 95% of values fall within two standard deviations of the mean. Therefore, you generate those values for the entire cuberoot column and add them as the threshold. You can use grouping again and look at each version and platform family as well. Then you can generate them and use them to create high and low thresholds to use for analysis.

Note the following about what you will get from this analysis:

- You previously removed any platforms that show a zero crash rate. Those platforms were not interesting for what you wanted to explore. Keeping those in the analysis would skew your data a lot toward the “good” side outliers—versions that show no crashes at all.
- You already removed a few very high outliers that would have skewed your data. Do not forget to count them in your final list of findings.

In order to apply this analysis, you create a new function to compare the cuberoot column against the thresholds, as shown in Figure 10-65.

```

1 def outliers(x):
2     if x['cuberoot'] > x['highside']:
3         x['outlier']="yes-high"
4     if x['cuberoot'] < x['lowside']:
5         x['outlier']="yes-low"
6     return x

1 adf6['outlier']="no"
2 adf7=adf6.apply(outliers, axis=1)

```

Figure 10-65 *Identifying Outliers in the Dataframe*

Now you can create a column in your dataframe to identify these outliers and set it to `no`. Then you can apply the function and create a new dataframe with your outliers. Figure 10-66 shows how you filter this dataframe to find only outliers.

1	adf7[adf7.outlier.str.contains("yes")]\				
2	[['ver', 'productFamily', 'rate', 'outlier']]				
	ver	productFamily	rate	outlier	
18	15_0	Cisco_7600_Series_Routers	8.78	yes-high	
116	15_1	Cisco_Catalyst_6500_Series_Switches	7.39	yes-high	
269	15_4	Cisco_7600_Series_Routers	7.54	yes-high	
276	15_4	Cisco_Catalyst_6500_Series_Switches	7.22	yes-high	

Figure 10-66 *Statistical Outliers for Crash Rates*

Based on the data you have and your SME knowledge, you can tell that these are older switches and routers (except for the mis-labeled 5900) that are using newer software versions. They are correlated with higher crash rates. You do not have the data to determine the causation. Outlier analysis such as this guides the activity prioritization toward understanding the causation. It can be assumed that these end-of-life devices were problematic with the older software versions in the earlier trains, and they are in a cycle of trying to find newer software versions that will alleviate those problems. Those problems could be with the software, the device hardware, the configuration, the deployment method, or many other factors. You cannot know until you gather more data and use more SME analysis and analytics.

If your role is software analysis, you just found 4 areas to focus on next, out of 800,000 records:

- 15_3 on Aironet 1140 wireless devices
- 15_5 on Cisco 5900 routers
- 15_0 and 15_4 on 7600 routers
- 15_1 and 15_4 on 6500 switches

This analysis was on entire trains of software. You could choose to go deeper into each version or perform very granular analysis on the versions that comprise each of these families. You now have the tools to do that part on your own.

Note Do not be overly alarmed or concerned with device crashes. Software often resets to self-repair failure conditions, and well-designed network environments continue to operate normally as they recover from software resets. A crash in a network device is not equal to a business-impacting outage in a well-designed network. Since you do not have the data, the SME validation of your statistical results is provided here.

- The Aironet 1140 series showed a large number of crashes for a single deployment, using a single version of 15_3_3_JAB, which skewed the results. It is a very old platform, which is no longer supported by Cisco.

- There is a non-traditional reset reason, and no real crashes for the 5900 routers, so the data is misleading for this platform. This finding should be disregarded.
- There were a number of crashes in the earlier version of 15.0.1S for 7600, and these releases are no longer available for download on the Cisco website.
- There were a number of crashes observed in the earlier versions of 15.4.3S for 7600. Recommended version for 7600 are later 15.4 or 15.5 releases.
- There were a number of crashes in the early releases of 15.1 for 6500. The majority of these crashes were on the software that has since been deferred by Cisco.
- For Catalyst 6500 with 15.4, there were only 97 devices represented in the data, with 7 crashes from a single deployment from 2017. No crashes with 6500 and 15.4 have been observed in 2018 data. The unknown problems with this one deployment and a small overall population skewed the data.

After statistical analysis, and SME validation, you can use this kind of analysis to make informed decisions for deploying software. Since every network environment is unique in some way, your results using your own data may turn up entirely different findings.

Summary

This chapter has spent a lot of time on dataframes. A dataframe is a heavily used data construct that you should understand in detail as you learn data science techniques to support your use cases. Quite a bit of time was spent in this chapter on how to programmatically and systematically step through data manipulation, visualization, analysis, and statistical testing and model building.

While this chapter is primarily about the analytics process when starting from the data, you also gained a few statistical solutions to use in your use cases. The atomic components you developed in this chapter are about uncovering true base rates from your data and comparing those base rates in statistically valid ways. You learned that you can use your outputs to uncover anomalies in your data.

If you want to operationalize this system, you can do it in a batch manner by building your solution into an automated system that takes daily or weekly batches of data from your environment and run this analysis as a Python program. You can find libraries to export the data from variables at any point during the program. Providing an always-on, real-time list of the findings from each of these sections in one notification email or dashboard allows you and your stakeholders to use this information as context for making maintenance activity decisions. Your decision making then comes down to a decision about whether you want to upgrade the high-count devices or the high crash rate devices in the next maintenance window. Now you can identify which devices have high counts of crashes, and which devices have a high rate of crashes.

The next chapter uses the infrastructure data again to move into unsupervised learning techniques you can use as part of your growing collection of components for use cases.

This page intentionally left blank

Developing Real Use Cases: Network Infrastructure Analytics

This chapter looks at methods for exploring your network infrastructure. The inspiration for what you will build here came from industry cases focused on the *find people like me* paradigm. For example, Netflix looks at your movie preferences and associates you and people like you with common movies. As another example, Amazon uses *people who bought this also bought that*, giving you options to purchase additional things that may be of interest for you, based on purchases of other customers. These are well-known and popular use cases. Targeted advertising is a gazillion-dollar industry (I made up that stat), and you experience this all the time. Do you have any loyalty cards from airlines or stores?

So how does this relate to network devices? We can translate *people like me* to *network devices like my devices*. From a technical sense, this is much easier than finding people because you know all the metadata about the devices. You cannot predict *exact* behavior based on similarity to some other group, but you can identify a tendency or look at consistency. The goal in this chapter is not to build an entire recommender system but to use unsupervised machine learning to identify similar groupings of devices. This chapter provides you with the skills to build a powerful machine learning–based information retrieval system that you can use in your own company.

What network infrastructure tendencies are of interest from a business standpoint? The easiest and most obvious is network devices that exhibit positive or negative behavior that can affect productivity or revenue. Cisco Services is in the business of optimizing network performance, predicting and preventing crashes, and identifying high-performing devices to emulate.

You can find devices around the world that have had an incident or crash or that have been shown to be extra resilient. Using machine learning, you can look at the world from the perspective of that device and see how similar other devices are to that one. You can also note the differences between positive- or negative-performing devices and understand what it takes to be like them. For Cisco, if a crash happens in any

network that is covered, devices are immediately identified in other networks that are extremely similar.

You now know both the problem you want to solve and what data you already have. So let's get started building a solution for your own environment. You will not have the broad comparison that Cisco can provide by looking at many customer environments, but you can build a comparison of devices within your own environment.

Human DNA and Fingerprinting

First, because the boss will want to know what you are working on, you need to come up with a name. Simply explaining that I was building an anonymized feature vector for every device to do similarity lookups fell a bit flat. My work needed some other naming so that the nontechnical folks could understand it, too. I needed to put on the innovation hat and do some associating to other industries to see what I could use as a method for similarity. In human genome research, it is generally known that some genes make you predisposed to certain conditions. If you can identify early enough that you have these genes, then you can be proactive about your health to ward off to some extent the growth of that predisposition into a disease.

I therefore came up with term *DNA mapping* for this type of exercise, which involves breaking down the devices to the atomic parts to identify predisposition to known events. My manager suggested *fingerprinting* as a name, and that had a nice fit with what we wanted to do, so we went with it. Because we would only be using device meta-data, this allowed for a distinction from a more detailed *full DNA* that is a longer-term goal, where we could include additional performance, state, policy, and operational characteristics of every device.

So how can you use fingerprinting in your networks to solve challenges? If you can find that one device crashed or had an issue, you can then look for other devices that have a fingerprint similar to that of the affected device. In Cisco, we bring our findings to the attention of the customer-facing engineer, who can then look at it for their customers. You cannot predict exactly what will happen with unsupervised learning. However, you can identify tendencies, or predispositions, and put that information in front of the folks who have built mental models of their customer environments. At Cisco Services, these are the primary customer engineers, and they provide a perfect combination of analytics and expertise for each Cisco customer.

Your starting point is modeled representations of millions of devices, including hardware, software, and configuration, as shown in Figure 11-1. You saw the data processing pipeline details for this in Chapter 9, “Building Analytics Use Cases,” and Chapter 10, “Developing Real Use Cases: The Power of Statistics.”

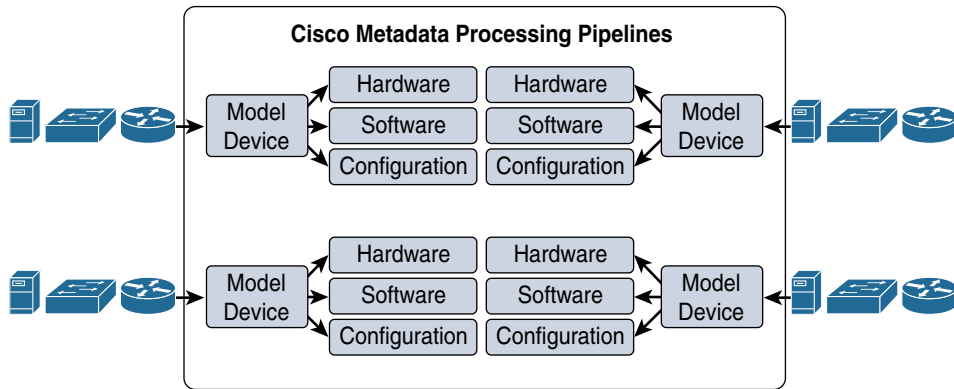


Figure 11-1 *Data Types for This Chapter*

Your goal is to determine which devices are similar to others. Even simpler, you also want to be able to match devices based on any given query for hardware, software, or configuration. This sounds a lot like the Internet, and it seems to be what Google and other search algorithms do. If Google indexes all the documents on the Internet and returns the best documents, based on some tiny document that you submit (your search query), why can't you use information retrieval techniques to match a device in the index to some other device that you submit as a query? It turns out that you can do this with network devices, and it works very well. This chapter starts with the search capability, moves through methods for grouping, and finishes by showing you how to visualize your findings in interesting ways.

Building Search Capability

In building this solution, note that the old adage that “most of the time is spent cleaning and preparing the data” is absolutely true. Many Cisco Services engineers built feature engineering and modeling layers over many years. Such a layer provides the ability to standardize and normalize the same feature on any device, anywhere. This is a more detailed set of the same data explored in Chapter 10. Let's get started.

Loading Data and Setting Up the Environment

First, you need to import a few packages you need and set locations from which you will load files and save indexes and artifacts, as shown in Figure 11-2. This chapter shows how to use pandas to load your data, nltk to tokenize it, and Gensim to create a search index.

```

1 import pandas as pd
2 import nltk
3 from gensim import corpora, models, matutils, similarities

1 routerdf_saved="/scratch/data/booktest/hw_sw_config_vectors_routers.csv"
2 ddf_saved="/scratch/data/booktest/detailed_devices_anonymized38.csv"
3 profile_dictionary_saved="/scratch/data/booktest/profile_dictionary"
4 profile_index_saved="/scratch/data/booktest/profile_index"
5 profile_id_saved="/scratch/data/booktest/profile_id_lookups.txt"

```

Figure 11-2 *Loading Data for Analysis*

You will be working with an anonymized data set of thousands of routers in this section. These are representations of actual routers seen by Cisco. Using the Gensim package, you can create the dictionary and index required to make your search functionality. First, however, you will do more of the work in Python pandas that you learned about in Chapter 10 to tackle a few more of the common data manipulations that you need to know. Figure 11-3 shows a new way to load large files. This is sometimes necessary when you try to load files that consume more memory than you have available on your system.

```

1 chunks = pd.read_csv(routerdf_saved, iterator=True, chunksize=10000)
2 df = pd.concat(list(chunks), ignore_index=True)
3 df[:2]

```

	id	profile	len
0	1541999266057 cisco2610 wic_2t c2600_i_m ios 12_2_6_acl_ip_...		2061
1	1541999212499 cisco2801 vic2_4fxo pvdmm2_32 c2801_spservicesk...		2242

Figure 11-3 *Loading Large Files*

In this example, the dataframe is read in as small chunks, and then the chunks are all assembled together to give you the full dataframe at the end. You can read data in chunks if you have large data files and limited memory capacity to load this data. This dataframe has some profile entries that are thousands of characters long, and in Figure 11-4 you sort them based on a column that contains the length of the profile.

```

1 dflen = df.sort_values(by='len')
2 dflen[120:122]

```

	id	profile	len
363535	426290635362 cisco2921_k9 hwic_16a pwr_2921_51_ac c2900_uni...		444
373129	523891169507 cisco2921_k9 hwic_16a pwr_2921_51_ac c2900_uni...		444

Figure 11-4 *Sorting a Dataframe*

Note that you can slice a few rows out of the dataframe at any location by using square brackets. If you grab one of these index values, you can see the data at any location in your dataframe by using pandas `loc` and the row index value, as shown in Figure 11-5. You can use Python `print` statements to print the entire cell, as Jupyter Notebook sometimes truncates the data.

```
1 print(df1en.loc[363535].profile)

cisco2921_k9 hwic_16a pwr_2921_51_ac c2900_universalk9_mz_spa ios 15_3_3_m6 cla
ssless_ip_routing_post_cscsa75635_ios enabling_mdns_gateway_functionality_ios p
arser_cache dhcp_server_and_relay_agent potential_smart_install_client_ios logg
ing_syslog_enabled ike_not_explicitly_disabled classless_ip_routing smart_insta
ll_client_devices_ios cisco_discovery_protocol_cdp sip_psirt ip_source_routing
device_sensor_with_dhcp_filter_not_configured_ios
```

Figure 11-5 *Fingerprint Example*

This small profile is an example of what you will use as the hardware, software, and configuration fingerprint for devices in this chapter. In this dataframe, you gathered every hardware and software component and the configuration model for a large group of routers. This provides a detailed record of the device as it is currently configured and operating.

How do you get these records? This data set was a combination of three other data sets that include millions of software records indicating every component of system software, firmware, software patches, and upgrade packages. Hardware records for every distinct hardware component down to the transceiver level come from another source. Configuration profiles for each device are yet another data source from Cisco expert systems. Note that it was important here to capture all instances of hardware, software, and configuration to give you a valid model of the complexity of each device. As you know, the same device can have many different hardware, software, and configuration options.

Note The word *distinct* and not *unique* is used in this book when discussing fingerprints. Unlike with human fingerprints, it is very possible to have more than one device with the same fingerprint. Having an identical fingerprint is actually desirable in many network designs. For example, when you deploy devices in resilient pairs in the core or distribution layers of large networks, identical configuration is required for successful failover. You can use the search engine and clustering that you build in your own environment to ensure consistency of these devices.

Once you have all devices as collections of fingerprints, how do you build a system to take your solution to the next level? Obviously, you want the ability to match and search, so some type of similarity measure is necessary to compare device fingerprints to other device fingerprints. A useful Python library is Gensim, (<https://radimrehurek.com/gensim/>).

Gensim provides the ability to collect and compare documents. Your profiles (fingerprints) are now documents. They are valid inputs to any text manipulation and analytics algorithms.

Encoding Data for Algorithmic Use

Before you get to building a search index, you should explore the search options that you have without using machine learning. You need to create a few different representations of the data to do this. In your data set, you already have a single long profile for each device. You also need a transformation of that profile to a tokenized form. You can use the nltk tokenizer to separate out the individual features into tokenized lists. This creates a bag of words implementation for each fingerprint in your collection, as shown in Figure 11-6. A bag of words implementation is useful when the order of the terms does not matter: All terms are just tossed into a big bag.

```
1 import nltk
2 df['tokens'] = df.apply(lambda row: \
3     nltk.word_tokenize(row['profile']), axis=1)
4 tokens_list=df['tokens']
5 profile_dictionary = corpora.Dictionary(tokens_list)
6 profile_dictionary[1]

u'cisco_discovery_protocol_cdp'

1 len(profile_dictionary)

14888
```

Figure 11-6 *Tokenizing and Dictionaries*

Immediately following the tokenization here, you can take all fingerprint tokens and create a dictionary of terms that you want to use in your analysis. You can use the newly created token forms of your fingerprint texts in order to do this. This dictionary will be the domain of possible terms that your system will recognize. You will explore this dictionary later to see how to use it for encoding queries for machine learning algorithms to use. For now, you are only using this dictionary representation to collect all possible terms across all devices. This is the full domain of your hardware, software, and configuration in your environment.

In the last line of Figure 11-6, notice that there are close to 15,000 possible features in this data set. Each term has a dictionary number to call upon it, as you see from the Cisco Discovery Protocol (CDP) example in the center of Figure 11-6. When you generate queries, every term in the query is looked up against this dictionary in order to build the numeric representation of the query. You will use this numeric representation later to find the similarity percentage. Terms not in this dictionary are simply not present in the query because the lookup to create the numeric representation returns nothing.

The behavior of dropping out terms not in the dictionary at query time is useful for refining your searches to interesting things. Just leave them out of the index creation, and they will not show up in any query. This form of *context-sensitive stop words* allows for noise and garbage term removal as part of everyday usage of your solution. Alternatively, you could add a few extra features of future interest in the dictionary if you made up some of your own features.

You now have a data set that you can search, as well as a dictionary of all possible search terms. For now, you will only use the dictionary to show term representations that you can use for search. Later you will use it with machine learning. Figure 11-7 shows how you can use Python to write a quick loop for partial matching to find interesting search terms from the dictionary.

```

1 checkme="2951"
2 for v in profile_dictionary.values():
3     if checkme in v:
4         print(v)

c2951_universalk9_m
c2951_universalk9_mz_spa
c2951_universalk9_npe_mz_spa
cisco2951_k9

1 df[df.profile.str.contains("cisco2951_k9")][:2]

```

Figure 11-7 *Using the Dictionary*

You can use any line from these results to identify a feature of interest that you want to search for in the profile dataframe that you loaded, as shown in Figure 11-8.

```

1 df[df.profile.str.contains("cisco2951_k9")][:2]

```

	id	profile	len	tokens
4	1541999303911	glc_sx_mmd cisco2951_k9 vwc3_2mft_t1_e1 vic2_...	4008	[glc_sx_mmd, cisco2951_k9, vwc3_2mft_t1_e1, v...
5	1541999303912	vwc3_2mft_t1_e1 vic2_4fxo pvdm3_64 sm_x_1t3_e...	3044	[vwc3_2mft_t1_e1, vic2_4fxo, pvdm3_64, sm_x_1...

Figure 11-8 *Profile Comparison*

You can find many 2951 routers by searching for the string contained in the profile column. Because the first two devices returned by the search are next to each other in the data when sorted by profile length, you can assume that they are from a similar environment. You can use them to do some additional searching. Figure 11-9 shows how you can load a more detailed dataframe to add some context to later searches and filter out single-entry dataframe views to your devices of interest. Notice that **devicedf** has only a single row when you select a specific device ID.



Figure 11-9 Creating Dataframe Views

Notice that this is a 2951 router. Examine the small profile dataframe view to select a partial fingerprint and get some ideas for search terms. You can examine only part of the thousands of characters in the fingerprint by selecting a single value as a string and then slicing that string, as shown in Figure 11-10. In pandas, `loc` chooses a row index from the dataframe and copies it to a string. Python also uses square brackets for string slicing, so in line 2 the square brackets choose the character locations. In this case you are choosing the first 210 characters.



Figure 11-10 Examining a Specific Cell in a Dataframe

You can filter for terms by using dataframe filtering to find similar devices. Each time you expand the query, you get fewer results that match all your terms. You can do this with Python loops, as shown in Figure 11-11.



Figure 11-11 Creating Filtered Dataframes

These loops do a few things. First, a loop makes a copy of your original dataframe and then loops through and whittles it down by searching for everything in the `split` string of query terms. The second loop runs three times, each time overwriting the working dataframe with the next filter. You end up with a whittled-down dataframe that contains all the search terms from your query.

Search Challenges and Solutions

As you add more and more search terms, the number of matches gets smaller. This happens because you eliminate everything that is not an exact match to the entire set of features of interest. In Figure 11-12, notice what happens when you try to match your entire feature set by submitting the entire profile as a search query.

```

1 query=devicepr.loc[4].profile
2 df2=df.copy()
3 for feature in query.split():
4     df2=df2[df2.profile.str.contains(feature)]
5 len(df2)

```

1

Figure 11-12 *Applying a Full Profile to a Dataframe Filter*

You get only one match, which is your own device. You are a snowflake. With feature vectors that range from 70 to 7000 characters, it is going to be hard to use filtering mechanisms alone for searches. What is the alternative? Because you already have the data in token format, you can use Gensim to create a search index to give you partial matches with a match percentage. Figure 11-13 shows the procedure you can use to do this.

```

1 profile_dictionary = corpora.Dictionary(tokens_list)
2 profile_ids=df["id"]
3 profile_corpus= [profile_dictionary.doc2bow(x) for x in tokens_list]
4 profile_index = similarities.Similarity(profile_index_saved, \
5     profile_corpus, num_features=len(profile_dictionary))
6 profile_index.save(profile_index_saved)
7 profile_dictionary.save(profile_dictionary_saved)
8 import pickle
9 with open(profile_id_saved, 'wb') as fp:
10     pickle.dump(profile_ids, fp)
11 fp.close()

```

Figure 11-13 *Creating a Search Index*

Although you already created the dictionary, you should do it here again to see how Gensim uses it in context. Recall that you used the feature tokens to create that dictionary. Using this dictionary representation, you can use the token list that you created previously to make a numerical vector representation of each device. You output all

these as a corpus, which is, by definition, a collection of documents (or collection of fingerprints in this case). The Gensim `doc2bow` (document-to-bag of words converter) does this. Next, you create a search index on disk to the `profile_index_saved` location that you defined in your variables at the start of the chapter. You can build this index from the corpus of all device vectors that you just created. The index will have all devices represented by all features from the dictionary that you created. Figure 11-14 provides a partial view of what your current test device looks like in the index of all corpus entries.

```
1 for feature in profile_corpus[4]:
2     print(feature),
(0, 1) (1, 1) (3, 1) (4, 1) (7, 1) (8, 1) (12, 1) (13, 1) (14, 1) (16, 1) (17,
1) (18, 1) (20, 1) (21, 1) (22, 1) (24, 1) (29, 1) (31, 1) (33, 1) (34, 1) (35,
1) (36, 1) (37, 1) (40, 1) (41, 1) (45, 1) (46, 1) (47, 1) (49, 1) (50, 1) (51,
1) (52, 1) (54, 1) (56, 1) (58, 1) (63, 1) (65, 1) (66, 1) (67, 1) (68, 1) (71,
1) (76, 1) (77, 1) (78, 1) (79, 1) (81, 1) (83, 1) (84, 1) (85, 1) (87, 1) (88,
1) (90, 1) (91, 1) (92, 1) (93, 1) (94, 1) (102, 1) (105, 1) (109, 1) (115, 1)
(116, 1) (117, 1) (118, 1) (119, 1) (122, 1) (127, 1) (128, 1) (129, 1) (130,
```

Figure 11-14 *Fingerprint Corpus Example*

Every one of these Python tuple objects represents a dictionary entry, and you see a count of how many of those entries the device has. Everything shows as count 1 in this book because the data set was deduplicated to simplify the examples. Cisco sometimes sees representations that have hundreds of entries, such as transceivers in a switch with high port counts.

You can find the `fxo vic` that was used in the earlier search example in the dictionary as shown in Figure 11-15.

```
1 profile_dictionary[91]
u'vic2_4fxo'
```

Figure 11-15 *Examining Dictionary Entries*

Now that you have an index, how do you search it with your terms of interest? First, you create a representation that matches the search index, using your search string and the dictionary. Figure 11-16 shows a function to take any string you send it and return the properly encoded representation to use your new search index.

```
1 def get_querystring(single_string, profile_dictionary):
2     testwordvec=nltk.word_tokenize(single_string)
3     string_profile = profile_dictionary.doc2bow(testwordvec)
4     return string_profile
```

Figure 11-16 *Function for Generating Queries*

Note that the process for encoding a single set of terms is the same process that you followed previously, except that you need to encode only a single string rather than thousands of them. Figure 11-17 shows how to use the device profile from your device and apply your function to get the proper representation.

```
1 query_string = get_querystring(deviceprofile, profile_dictionary)
2 for feature in query_string:
3     print(feature),
```

(0, 1) (1, 1) (3, 1) (4, 1) (7, 1) (8, 1) (12, 1) (13, 1) (14, 1) (16, 1) (17, 1) (18, 1) (20, 1) (21, 1) (22, 1) (24, 1) (29, 1) (31, 1) (33, 1) (34, 1) (35, 1) (36, 1) (37, 1) (40, 1) (41, 1) (45, 1) (46, 1) (47, 1) (49, 1) (50, 1) (51, 1) (52, 1) (54, 1) (56, 1) (58, 1) (63, 1) (65, 1) (66, 1) (67, 1) (68, 1) (71, 1) (76, 1) (77, 1) (78, 1) (79, 1) (81, 1) (83, 1) (84, 1) (85, 1) (87, 1) (88, 1) (90, 1) (91, 1) (92, 1) (93, 1) (94, 1) (102, 1) (105, 1) (109, 1) (115, 1) (116, 1) (117, 1) (118, 1) (119, 1) (122, 1) (127, 1) (128, 1) (129, 1) (130,

Figure 11-17 *Corpus Representation of a Test Device*

Notice when you expand your new `query_string` that it is a match to the representation shown in the corpus. Recall from the discussion in Chapter 2, “Approaches for Analytics and Data Science”, that building a model and implementing the model in production are two separate parts of analytics. So far in this chapter, you have built something cool, but you still have not implemented anything to solve your search challenge. Let’s look at how you can use your new search functionality in practice. Figure 11-18 shows the results of the first lookup for your test device.

```
1 profile_index.num_best=1000
2 similarity_matches=profile_index[query_string]
3 print(similarity_matches[0:10])
```

[(4, 0.99999934434890747), (10887, 0.98734188079833984), (5460, 0.96252012252807617), (3933, 0.95597422122955322), (2323, 0.95001983642578125), (4007, 0.94671422243118286), (11908, 0.94275224208831787), (823, 0.93890774250030518), (2801, 0.93843364715576172), (359, 0.92881762981414795)]

Figure 11-18 *Similarity Index Search Results*

This example sets the number of records to return to 1000 and runs the query on the index using the encoded query string that was just created. If you print the first 10 matches, notice your own device at corpus row 4 is a perfect match (ignoring the floating-point error). There are 3 other devices that are at least 95% similar to yours. Because you only have single entries in each tuple, only the first value that indicates the feature is unique, so you can do a simple set compare Python operation. Figure 11-19 shows how to use this compare to find the differences between your device and the closest neighbor with a 98.7% match.

```
1 diff1=set(profile_corpus[4])-set(profile_corpus[10887])
2 diff1

{(40, 1), (203, 1), (204, 1), (216, 1)}

1 print(profile_dictionary[40])
2 print(profile_dictionary[203])
3 print(profile_dictionary[204])
4 print(profile_dictionary[216])

ppp
mlppp_bundling_dsl_interfaces
multilink_ppp
ip_accounting_ios
```

Figure 11-19 Differences Between Two Devices Where First Device Has Unique Entries

By using `set` for corpus features that show in your device but not in the second device, you can get the differences and then use your dictionary to look them up. It appears that you have 4 features on your device that do not exist on that second device. If you check the other way by changing the order of the inputs, you see that the device does not have any features that you do not have already, as shown with the empty set in Figure 11-20. The hardware and software are identical because no differences appear here.

```
1 diff2=set(profile_corpus[10887])-set(profile_corpus[4])
2 diff2

set()
```

Figure 11-20 Differences Between Two Devices Where First Device Has Nothing Unique

You can do a final sanity check by checking the rows of the original dataframe using a combined dataframe search for both. Notice that the lengths of the profiles are 66 characters different in Figure 11-21. The 4 features above represent 62 characters. You can therefore add 4 spaces between, and you have an exact match.

1	df[((df.id==df.loc[4].id) (df.id==df.loc[10887].id))]			
	id	profile	len	tokens
4	1541999303911	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	4008	[glc_sx_mmd, cisco2951_k9, vwic3_2mft_t1_e1, v...
10887	1541999301844	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	3942	[glc_sx_mmd, cisco2951_k9, vwic3_2mft_t1_e1, v...

Figure 11-21 Profile Length Comparison

Cisco often sees 100% matches, as well as matches that are very close but not quite 100%. With the thousands of features and an almost infinite number of combinations of features, it is rare to see things 99% or closer that are not part of the same network.

These tight groupings help identify groups of interest just as Netflix and Amazon do. You can add to this simple search capability with additional analysis using algorithms such as latent semantic indexing (LSI) or latent Dirichlet allocation (LDA), random forest, and additional expert systems engagement. Those processes can get quite complex, so let's take a break from building the search capability and discuss a few of the ways to use it so you can get more ideas about building your own internal solution.

Here are some ways that this type of capability is used in Cisco Services:

- If a Cisco support service request shows a negative issue on a device that is known to our internal indexes, Cisco tools can proactively notify engineers from other companies that have very similar devices. This notification allows them to check their similar customer devices to make sure that they are not going to experience the same issue.
- This is used for software, hardware, and feature intelligence for many purposes. If a customer needs to replace a device with a like device, you can pull the topmost similar devices. You can summarize the hardware and software on these similar devices to provide replacement options that most closely match the existing features.
- When there is a known issue, you can collect that issue as a labeled case for supervised learning. Then you can pull the most similar devices that have not experienced the issue to add to the predictive analytics work.
- A user interface for full queries is available to engineers for ad hoc queries of millions of anonymized devices. Engineers can use this functionality for any purpose where they need comparison.

Figure 11-22 is an example of this functionality in action in the Cisco Advanced Services Business Critical Insights (BCI) platform. Cisco engineers use this functionality as needed to evaluate their own customer data or to gain insights from an anonymized copy of the global installed base.

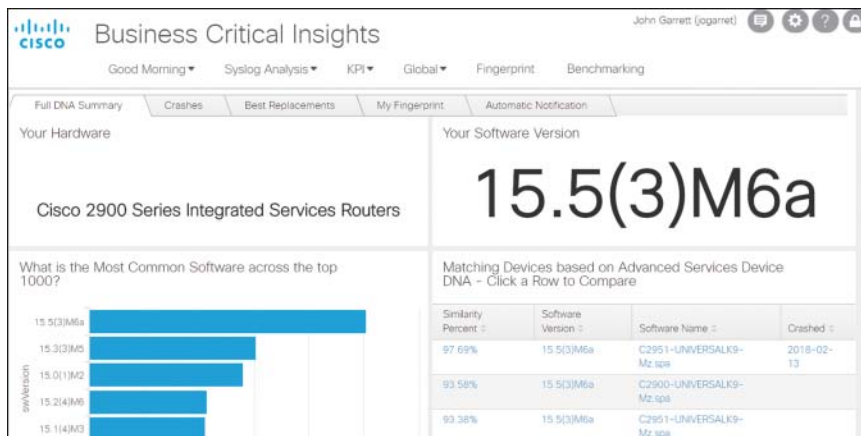


Figure 11-22 Cisco Services Business Critical Insights

Having a search index for comparison provides immediate benefits. Even without the ability to compare across millions of devices as at Cisco, these types of consistency checks and rapid search capability are very useful and are valid cases for building such capabilities in your own environment. If you believe that you have configured your devices in a very consistent way, you can build this index and use machine learning to prove it.

Other Uses of Encoded Data

What else can you do with fingerprints? You just used them as a basis for a similarity matching solution, realizing all the benefits of finding devices based on sets of features or devices like them. With the volume of data that Cisco Services has, this is powerful information for consultants. However, you can do much more with the fingerprints. Can you visually compare these fingerprints? It would be very hard to do so in the current form. However, you can use machine learning and encode them, and then you can apply dimensionality reduction techniques to develop useful visualizations. Let's do that.

First, you encode your fingerprints into vectors. As the name suggests, *encoding* is a mathematical formula to use when transforming the counts in a matrix to vectors for machine learning. Let's take a few minutes here to talk about the available transformations so that you understand the choices you can make when building these solutions. First, let's discuss some standard encodings used for documents.

With one-hot encoding, all possible terms have a column heading, and any term that is in the document gets a one in the row represented by the document. Your documents are rows, and your features are columns. Every other column entry that is not in the document gets a zero, and you have a full vector representation of each document when the encoding is complete, as shown in Figure 11-23. This is called a *term document matrix*, and it is the transposed form of document term matrix.

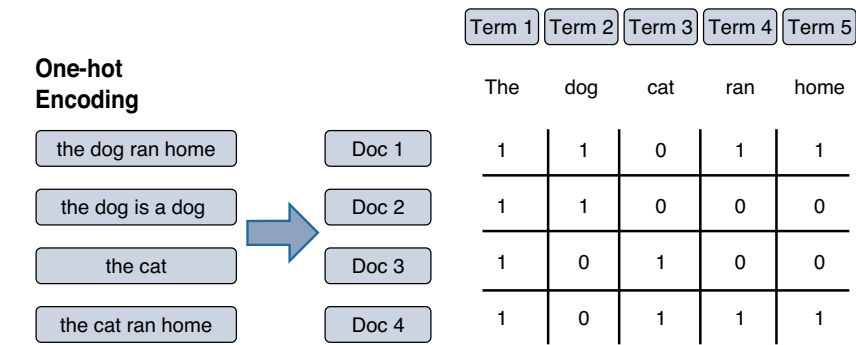


Figure 11-23 One-Hot Encoding

Encodings of sets of documents are stored in matrix form. Another method is count representation. In a count representation, the raw counts are used. With one-hot encoding you are simply concerned that there is at least one term, but with a count matrix you are interested in *how many* of something are in each document, as shown in Figure 11-24.

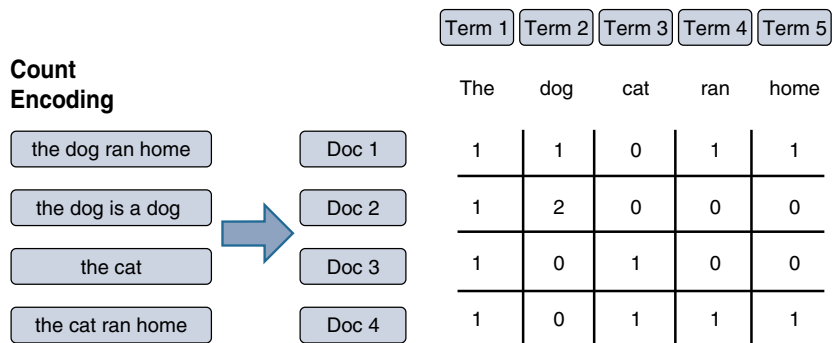


Figure 11-24 Count Encoded Matrix

Where this representation gets interesting is when you want to represent things that are rarer with more emphasis over things that are very common in the same matrix. This is where the term frequency/inverse document frequency (TF/IDF) encoding works best. The values in the encodings are not simple ones or counts but the frequency of each term divided by the inverse document frequency. Because you are not using TF/IDF here, it isn't covered, but if you intend to generate an index with counts that vary widely and have some very common terms (such as transceiver counts), keep in mind that TF/IDF provides better results for searching.

Dimensionality Reduction

In this section you will do some encoding and analysis using unsupervised learning and dimensionality reduction techniques. The purpose of dimensionality reduction in this context is to reduce/summarize the vast number of features into two or three dimensions for visualization.

For this example, suppose you are interested in learning more about the 2951 routers that are using the fxo and T1 modules used in the earlier filtering example. You can filter the routers to only devices that match those terms, as shown in Figure 11-25. Filtering is useful in combination with machine learning.

```
1 query="cisco2951_k9 vwic3_2mft_t1_e1 vic2_4fxo"
2 df2=df.copy()
3 for feature in query.split():
4     df2=df2[df2.profile.str.contains(feature)].copy()
5 len(df2)
```

3856

Figure 11-25 Filtered Data Set for Clustering

Notice that 3856 devices were found that have this fxo with a T1 in the same 2951 chassis. Now encode these by using one of the methods discussed previously, as shown in Figure 11-26. Because you have deduplicated features, many encoding methods will work for your purpose. Count encoding and one-hot encoding are equivalent in this case.

```

1 from sklearn.feature_extraction.text import CountVectorizer
2 vectorizer = CountVectorizer()
3 word_transform = vectorizer.fit_transform(df2['profile'])
4 X_matrix = word_transform.todense()
5 print(X_matrix.shape)

```

(3856, 1155)

Figure 11-26 *Creating the Count-Encoded Matrix*

Using the Scikit-learn `CountVectorizer`, you can create a vectorizer object that contains all terms found across all profiles of this filtered data set and fit it to your data. You can then convert it to a dense matrix so you have the count encoding with both ones and zeros, as you expect to see it. Note that you have a row for each of the entries in your data and more than 1100 unique features across that group, as shown by counting the length of the feature list in Figure 11-27.

```

1 feature_list=vectorizer.get_feature_names()
2 print("Feature list length is " + str(len(feature_list)))
3 for f in feature_list:
4     if "fxo" in f:
5         print(f)

```

Feature list length is 1155
em_hda_3fxs_4fxo
em_hda_6fxo
fxo_answer_and_disconnect_supervision
specify_guardout_duration_of_foreign_exchange_office_fxo_voice_port_ios
vic2_2fxo
vic2_4fxo

Figure 11-27 *Finding Vectorized Features*

When you extract the list of features found by the vectorizer, notice the `fxo` module you expect, as well as a few other entries related to `fxo`. The list contains all known features from your new filtered data set only, so you can use a quick loop for searching substrings of interest. Figure 11-28 shows a count-encoded matrix representation.

```

1 X_matrix

```

matrix([[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 1, 1, 1],
...,
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0]])

Figure 11-28 *Count-Encoded Matrix Example*

You have already done the filtering and searching, and you have examined text differences. For this example, you want to visualize the components. It is not possible for your stakeholders to visualize differences across a matrix of 1100 columns and 3800 rows. They will quickly tune out.

You can use dimensionality reduction to get the dimensionality of an 1155×3856 matrix down to 2 or 3 dimensions that you can visualize. In this case, you need machine learning dimensionality reduction.

Principal component analysis (PCA) is used here. Recall from Chapter 8, “Analytics Algorithms and the Intuition Behind Them,” that PCA attempts to summarize the most variance in the dimensions into component-level factors. As it turns out, you can see the amount of variance by simply trying out your data with the PCA algorithm and some random number of components, as shown in Figure 11-29.

```
1 from sklearn.decomposition import PCA
2 pca = PCA(n_components=8)
3 pca.fit(X_matrix)
4 pca.explained_variance_ratio_

array([ 0.40207146,  0.10423278,  0.05828202,  0.05291481,  0.04332381,
        0.03639082,  0.03002522,  0.02587077])
```

Figure 11-29 *PCA Explained Variance by Component*

Notice that when you evaluate splitting to eight components, the value diminishes to less than 10% explained variance after the second component, which means you can generalize about 50% of the variation in just two components. This is just what you need for a 2D visualization for your stakeholders. Figure 11-30 shows how PCA is applied to the data.

```
1 pca_data = PCA(n_components=2).fit_transform(X_matrix)
2 pca1=[]
3 pca2=[]
4 for index, instance in enumerate(pca_data):
5     pca_1, pca_2 = pca_data[index]
6     pca1.append(pca_1)
7     pca2.append(pca_2)
8 print(len(pca1))
9 print(len(pca2))

3856
3856
```

Figure 11-30 *Generating PCA Components*

You can gather all the component transformations into a few lists. Note that the length of each of the component lists matches your dataframe length. The matrix is an encoded representation of your data, in order. Because the PCA components are a representation of the data, you can add them directly to the dataframe, as shown in Figure 11-31.

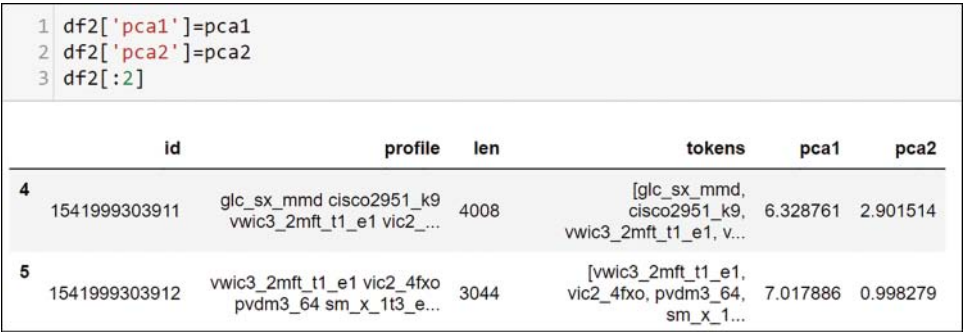


Figure 11-31 Adding PCA to the Dataframe

Data Visualization

The primary purpose of the dimensionality reduction you used in the previous section is to bring the data set down to a limited set of components to allow for human evaluation. Now you can use the PCA components to generate a visualization by using matplotlib, as shown in Figure 11-32.

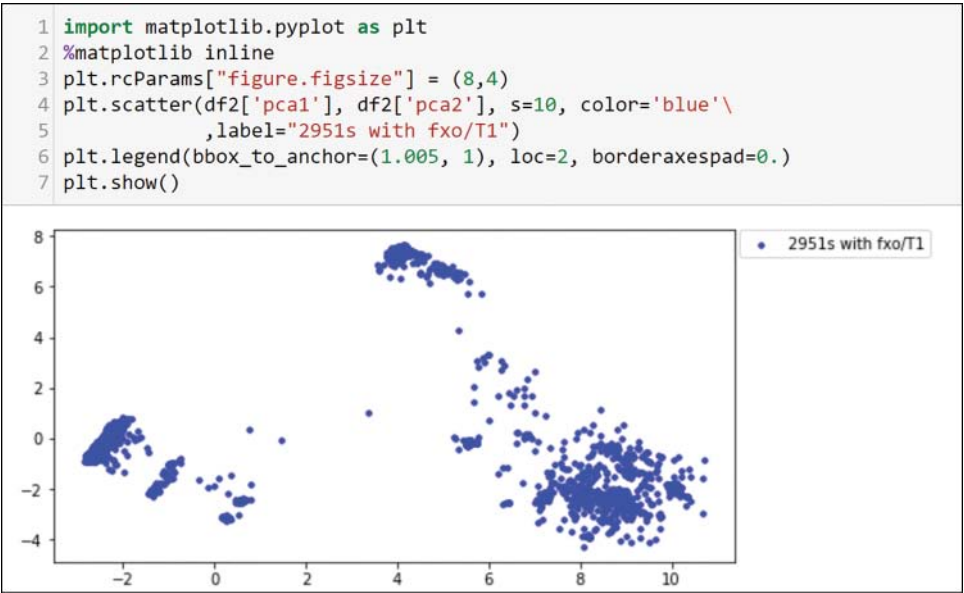


Figure 11-32 Visualizing PCA Components

In this example, you use matplotlib to generate a scatter plot using the PCA components directly from your dataframe. By importing the full matplotlib library, you get much more flexibility with plots than you did in Chapter 10. In this case, you choose an overall

plot size you like and add size, color, and a label for the entry. You also add a legend to call out what is in the plot. You only have one data set for now, but you will change that by identifying your devices of interest and overlaying them onto subsequent plots.

Recall your interesting device from Figure 11-21 and the device that was most similar to it. You can now create a visualization on this chart to show where those devices stand relative to all the other devices by filtering out a new dataframe or view, as shown in Figure 11-33.

1	df3=df2[((df2.id==1541999303911) (df2.id==1541999301844))]						
2	df3						
	id	profile	len	tokens	fingerprint	pca1	pca2
4	1541999303911	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	4008	[glc_sx_mmd, cisco2951_k9, vwic3_2mft_t1_e1, v...	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	6.328761	2.901514
10887	1541999301844	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	3942	[glc_sx_mmd, cisco2951_k9, vwic3_2mft_t1_e1, v...	glc_sx_mmd cisco2951_k9 vwic3_2mft_t1_e1 vic2_...	6.271231	2.693747

Figure 11-33 Creating Small Dataframes to Add to the Visualization

df3 in this case only has two entries, but they are interesting entries that you can plot in the context of other entries, as shown in Figure 11-34.

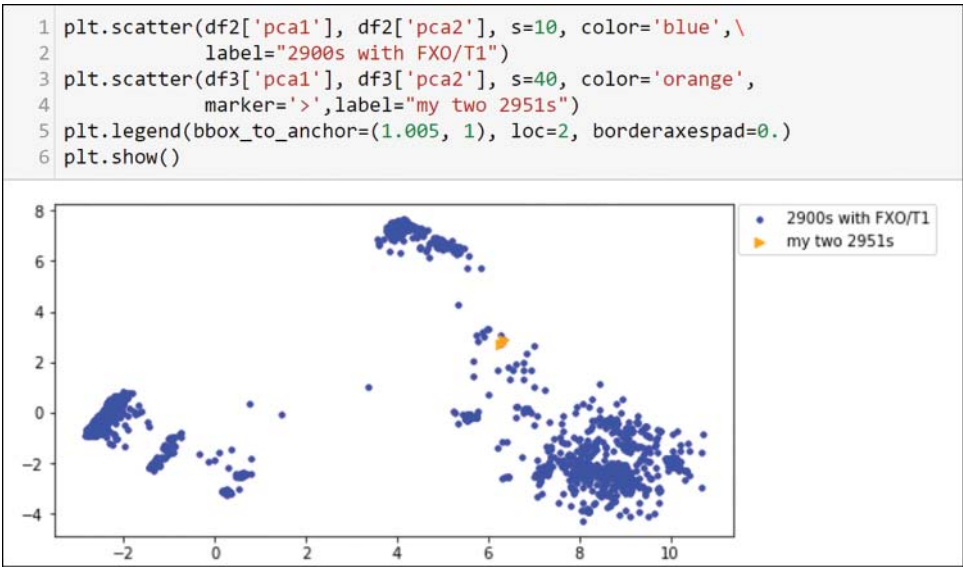


Figure 11-34 Two Devices in the Context of All Devices

What can you get from this? First, notice that the similarity index and the PCA are aligned in that the devices are very close to each other. You have not lost much information in the dimensionality reduction. Second, realize that with 2D modeling, you can easily represent 3800 devices in a single plot. Third, notice that your devices are not in a densely clustered area. How can you know whether this is good or bad?

One thing to do is to overlay the known crashes on this same plot. Recalling the crash matching logic from Chapter 10, you can identify the devices with a historical crash in this data set and add this information to your data. You can identify those crashes and build a new dataframe by using the procedure shown in Figure 11-35, where you use the data that has the resetReason column available to identify device IDs that showed a previous crash.

```

1 mylist=list(df2.id)
2 ddf1=ddf[ddf.id.isin(mylist)]
3 crashes=["error","watchdog","kernel","0x","abort","crash","ailure__",\
4          "generic_failure","s_w_reset","fault","reload_at_0",\
5          "reload_at@", "reload_at$"]
6 ddf2=ddf1[ddf1.resetReason.str.contains('|'.join(crashes))].copy()
7 ddf2['crash1']=1
8 ddf3=ddf2[['id', 'crash1']].copy()
9 len(ddf3)

```

42

Figure 11-35 *Generating Crash Data for Visualization*

Of the 3800 devices in your data, 42 showed a crash in the past. You know from Chapter 10 that this is a not a bad rate. You can identify the crashes and do some dataframe manipulation to add them to your working dataframe, as shown in Figure 11-36.

```

1 df2['crashed']=0.0
2 df4 = pd.merge(ddf3, df2, on='id', how='outer', indicator=False)
3 df4.fillna(0.0, inplace=True)
4 df4['crashed']=df4.apply(lambda x: x['crash1'] + x['crashed'], axis=1)
5 del df4['crash1']
6 df4.crashed.value_counts()

```

0.0	3814
1.0	42

Figure 11-36 *Adding Crash Data to a Dataframe*

What is happening here? You need a crash identifier to identify the crashes, so you add a column to your data set and initialize it to zero. In the previous section, you used crash1 as a column name. In this section, you create a new column called crashed in your previous dataframe and merge the dataframes so that you have both columns in the new

dataframe. This is necessary to allow the id field to align. For the dataframe of crashes with only 42 entries, all other entries in the new combined dataframe will have empty values, so you use the pandas `fillna` functionality to make them zero. Then you just add the crash1 and crashed columns together so that, if there is a crash, information about the crash makes it to the new crashed column. Recall that the initial crashed value is zero, so adding a noncrash leaves it at zero, and adding a crash moves it to one. Notice that Figure 11-36 correctly identified 42 of the entries as crashed.

Now you can copy your crash data from the new dataframe into a crash-only dataframe, as shown in Figure 11-37.

```
1 df5=df4[df4.crashed==1.0]
2 df3=df4[(df4.id==1541999303911) | (df4.id==1541999301844)]
```

Figure 11-37 *Generating Dataframes to Use for Visualization Overlays*

In case any of your manipulation changed any rows (it shouldn't have), you can also generate your interesting devices dataframe again here. You can plot your new data with crashes included as shown in Figure 11-38.

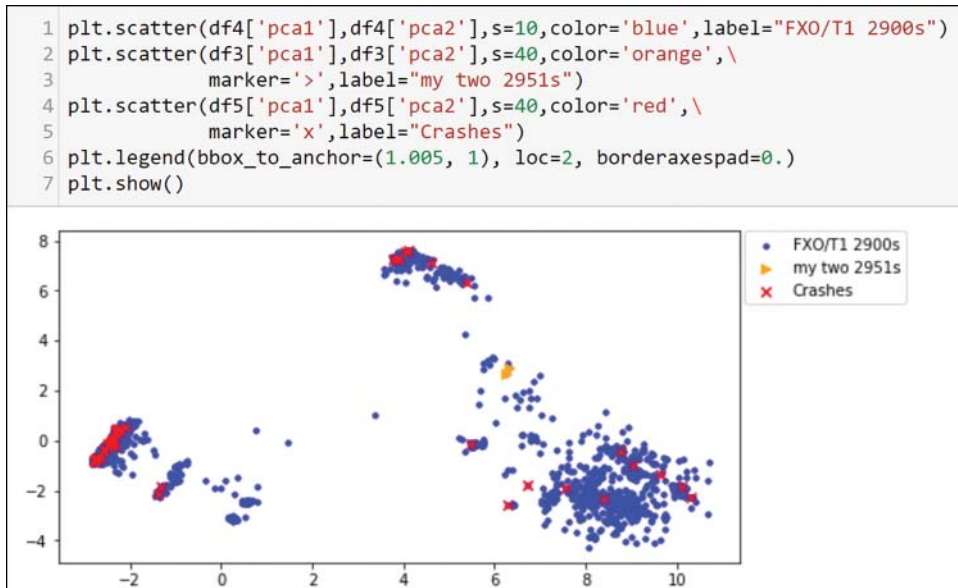


Figure 11-38 *Visualizing Known Crashes*

Entries that you add later will overlay the earlier entries in the scatterplot definition. Because this chart is only 2D, it is impossible to see anything behind the markers on the chart. Matching devices have the same marker location in the plot. Top-down order mat-

ters as you determine what you show on the plot. What you see is that your two devices and devices like them appear to be in a place that does not exhibit crashes. How can you know that? What data can you use to evaluate *how* safe this is?

K-Means Clustering

Unsupervised learning and clustering can help you see if you fall into a cluster that is associated to higher or lower crash rates. Figure 11-39 shows how to create a matrix representation of the data you can use to see this in action.

```
1 vectorizer = CountVectorizer()
2 word_transform = vectorizer.fit_transform(df4['profile'])
3 X_matrix = word_transform.todense()
```

Figure 11-39 *Generating Data for Clustering*

Instead of applying the PCA reduction, this time you will perform clustering using the popular K-means algorithm. Recall the following from the Chapter 8:

- K-means is very scalable for large data sets.
- You need to choose the number of clusters.
- Cluster centers are interesting because new entries can be added to the best cluster by using the closest cluster center.
- K-means works best with globular clusters.

Because you have a lot of data and large clusters that appear to be globular, using K-means seems like a good choice. However, you have to determine the number of clusters. A popular way to do this is to evaluate a bunch of possible cluster options, which is done with a loop in Figure 11-40.

```
1 import numpy as np
2 from scipy.spatial.distance import cdist
3 from sklearn.cluster import KMeans
4 tightness = []
5 possibleKs = range(1,10)
6 for k in possibleKs:
7     km = KMeans(n_clusters=k).fit(X_matrix)
8     km.fit(X_matrix)
9     tightness.append(sum(np.min(cdist(X_matrix, \
10         km.cluster_centers_, 'euclidean'), axis=1)) / X_matrix.shape[0])
```

Figure 11-40 *K-means Clustering Evaluation of Clusters*

Using this method, you run through a range of possible cluster-K values in your data and determine the relative tightness (or distortion) of each cluster. You can collect the tightness values into a list and plot those values as shown in Figure 11-41.

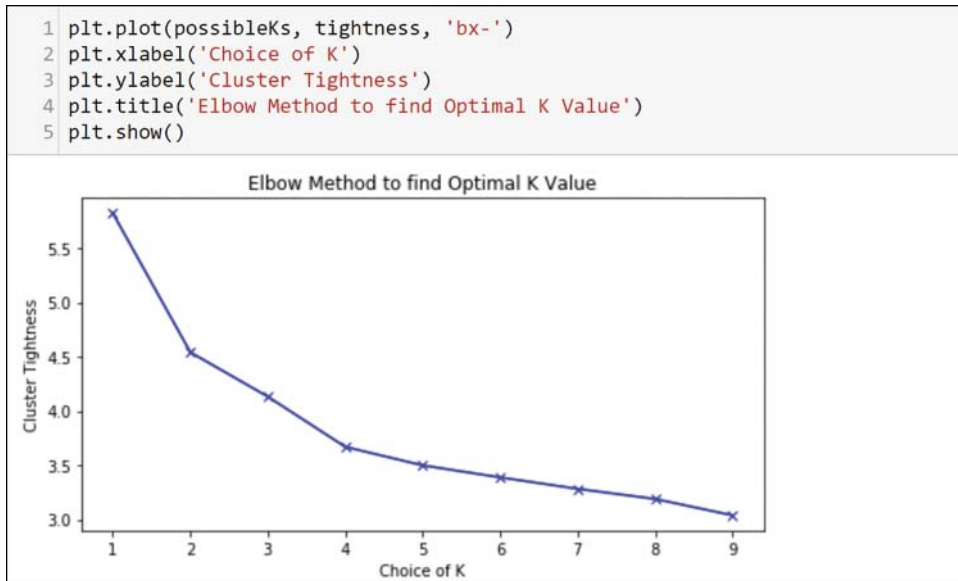


Figure 11-41 *Elbow Method for K-means Cluster Evaluation*

The elbow method shown in Figure 11-41 is useful for visually choosing the best change in cluster tightness covered by each choice of cluster number. You are seeking the cutoff where it appears to have an elbow which shows that the next choice of K does not maintain a strong trend downward. Notice these elbows at K=2 and K=4 here. Two clusters would not be very interesting, so let's explore four clusters for this data. Different choices of data and features for your analysis can result in different-looking plots, and you should include this evaluation as part of your clustering process. Choosing four clusters, you run your encoded matrix through the algorithm as shown in Figure 11-42.

```
1 kclusters=4
2 km = KMeans(n_clusters=kclusters,n_init=100,random_state=0)
3 km.fit_predict(X_matrix)
4 len(km.labels_.tolist())
```

3856

Figure 11-42 *Generating K-means Clusters*

In this case, after you run the K-means algorithm, you see that there are labels for the entire data set, which you can add as a new column, as shown in Figure 11-43.

```

1 df4['kcluster']=km.labels_.tolist()
2 df4[['id','kcluster']][:2]

```

	id	kcluster
0	1541999304184	2
1	1541999304424	2

Figure 11-43 *Adding Clusters to the Dataframe*

Look at the first two entries of the dataframe and notice that you added a column for clusters back to the dataframe. You can look at crashes per cluster by using the **groupby** method that you learned about in Chapter 10, as shown in Figure 11-44.

```

1 dfgroup1=df4.groupby(['kcluster','crashed'])
2 df6=dfgroup1.size().reset_index(name='count')
3 print(df6)

```

	kcluster	crashed	count
0	0	0.0	1435
1	0	1.0	11
2	1	0.0	671
3	1	1.0	10
4	2	0.0	1432
5	2	1.0	14
6	3	0.0	276
7	3	1.0	7

Figure 11-44 *Clusters and Crashes per Cluster*

It appears that there are crashes in every cluster, but the sizes of the clusters are different, so the crash rates should be different as well. Figure 11-45 shows how to use the **totals** function defined in Chapter 10 to get the group totals.

```

1 def myfun(x):
2     x['totals'] = x['count'].agg('sum')
3     return x
4 dfgroup2=df6.groupby(['kcluster'])
5 df7 = dfgroup2.apply(myfun)

```

Figure 11-45 *Totals Function to Use with pandas apply*

Next, you can calculate the rate and add it back to the dataframe. You are interested in the rate, and you divide the count by the total to get that. Multiplying by 100 and rounding to two places provides a number. Noncrash counts provide an uptime rate, and crash counts provide a crash rate. You could leave in the uptime rate if you wanted, but in this case, you are interested in the crash rates per cluster only, so you can filter out a new dataframe with that information, as shown in Figure 11-46.


```
1 df7['rate'] = df7.apply\
2 (lambda x: round(float(x['count'])/float(x['totals']) * 100.0,2), axis=1)
3 df8=df7[df7.crashed==1.0]
4 df8
```

	kcluster	crashed	count	totals	rate
1	0	1.0	11	1446	0.76
3	1	1.0	10	681	1.47
5	2	1.0	14	1446	0.97
7	3	1.0	7	283	2.47

Figure 11-46 *Generating Crash Rate per Cluster*

Now that you have a rate for each of the clusters, you can use it separately or add it back as data to your growing dataframe. Figure 11-47 shows how to add it back to the dataframe you are using for clustering.

1	df4['crashrate']=0.0	
2	for c in list(df8.kcluster):	
3	df4.loc[df4.kcluster==c, 'crashrate']=df8[df8.kcluster==c].rate.max()	
4	df4.crashrate.value_counts()	
0.97	1446	
0.76	1446	
1.47	681	
2.47	283	
Name: crashrate, dtype: int64		

Figure 11-47 *Adding Crash Rate to the Dataframe*

You need to ensure that you have a crash rate column and set an initial value. Then you can loop through the **kcluster** values in your small dataframe and apply them to the columns by matching the right cluster. Something new for you here is that you appear to be assigning a full series on the right to a single dataframe cell location on the left in line 3. By using the **max** method, you are taking the maximum value of the filtered column only. There is only one value, so the max will be that value. At the end, notice that the crash rate numbers in your dataframe match up to the grouped objects that you generated previously.

Now that you have all this information in your dataframe, you can plot it. There are many ways to do this, but it is suggested that you pull out individual dataframe views per group, as shown in Figure 11-48. You can overlay these onto the same plot.

1	df9=df4[df4.kcluster==0]	
2	df10=df4[df4.kcluster==1]	
3	df11=df4[df4.kcluster==2]	
4	df12=df4[df4.kcluster==3]	

Figure 11-48 *Creating Dataframe Views per Cluster*

Figure 11-49 shows how to create some dynamic labels to use for these groups on the plots. This ensures that, as you try other data using this same method, the labels will reflect the true values from that new data.

```
1 c0="C10, Crashrate=" + str(df4[df4.kcluster==0].crashrate.max())
2 c1="C11, Crashrate=" + str(df4[df4.kcluster==1].crashrate.max())
3 c2="C12, Crashrate=" + str(df4[df4.kcluster==2].crashrate.max())
4 c3="C13, Crashrate=" + str(df4[df4.kcluster==3].crashrate.max())
```

Figure 11-49 *Generating Dynamic Labels for Visualization*

Figure 11-50 shows how to add all the dataframes to the same plot definition used previously to see everything in a single plot.

```
1 plt.scatter(df9['pca1'],df9['pca2'],s=20,color='blue',label=c0)
2 plt.scatter(df10['pca1'],df10['pca2'],s=20,marker='^',label=c1)
3 plt.scatter(df11['pca1'],df11['pca2'],s=20,marker='*',label=c2)
4 plt.scatter(df12['pca1'], df12['pca2'],s=20,marker='.',label=c3)
5 plt.scatter(df3['pca1'],df3['pca2'],s=40,color='orange',\
6           marker='>',label="my two 2951s")
7 plt.scatter(df5['pca1'],df5['pca2'],s=40,color='red',\
8           marker='x',label="Crashes")
9 plt.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
10 plt.title("2900 Cluster Crash Rates", fontsize=12)
11 plt.show()
```

Figure 11-50 *Combined Cluster and Crash for Plotting*

Because all the data is rooted from the same data set that you continue to add to, you can slice out any perspective that you want to put on your plot. You can see the resulting plot from this collection in Figure 11-51.

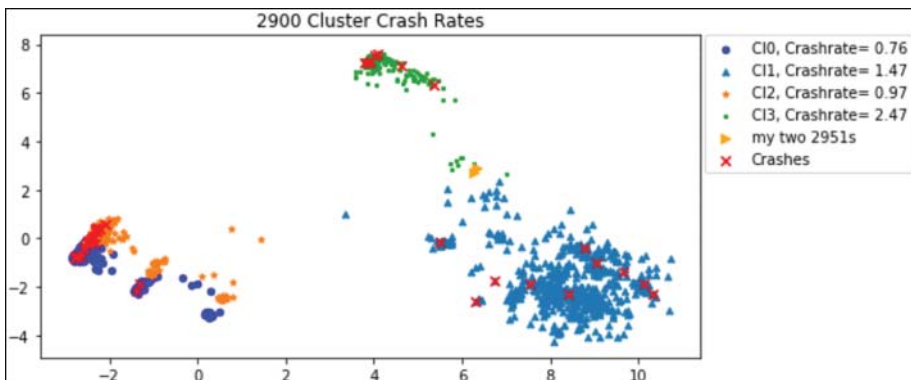


Figure 11-51 *Final Plot with Test Devices, Clusters, and Crashes*

The first thing that jumps out in this plot is the unexpected split of the items to the left. It is possible that there are better clustering algorithms that could further segment this area, but I leave it to you to further explore this possibility. If you check the base rates as you learned to do, you will find that this area to the left may appear to be small, but it actually represents 75% of your data. You can identify this area of the plot by filtering the PCA component values, as shown in Figure 11-52.

```
1 leftside=len(df4[df4['pca1']<2])
2 everything=len(df4)
3 float(leftside)/float(everything)
```

0.75

Figure 11-52 *Evaluating the Visualized Data*

So where did your interesting devices end up? They appear to be between two clusters. You can check the cluster mapping as shown in Figure 11-53.

```
1 df3=df4[((df4.id==1541999303911) | (df4.id==1541999301844))]
2 df3[['id', 'kcluster']]
```

	id	kcluster
42	1541999303911	3
2498	1541999301844	3

Figure 11-53 *Cluster Assignment for Test Devices*

It turns out that these devices are in a cluster that shows the highest crash rate. What can you do now? First, you can make a few observations, based on the figures you have seen in the last few pages:

- The majority of the devices are in tight clusters on the left, with low crash rates.
- Correlation is not causation, and being in a high crash rate cluster does not cause a crash on a device.
- While you are in this cluster with a higher crash rate, you are on the edge that is most distant from the edge that shows the most crashes.

Given these observations, it would be interesting to see the differences between your devices and the devices in your cluster that show the most crashes. This chapter closes by looking at a way to do that. Examining the differences between devices is a common troubleshooting task. A machine learning solution can help.

Machine Learning Guided Troubleshooting

Now that you have all your data in dataframes, search indexes, and visualizations, you have many tools at your disposal for troubleshooting. This section explores how to compare dataframes to guide troubleshooting. There are many ways to compare dataframes, but this section shows how to write a function to do comparisons across any two dataframes from this set (see Figure 11-54). Those could be the cluster dataframes or any dataframes that you choose to make.

This function normalizes the rate of deployment of individual features within each of the clusters and returns the rates that are higher than the threshold value. The threshold is 80% by default, but you can use other values. You can use the function to compare clusters or individual slices of your dataframe. Step through the function line by line, and you will recognize that you have learned most of it already. As you gain more practice, you can create combinations of activities like this to aid in your analysis.

```

1 def get_cluster_diffs(highdf,lowdf,threshold=80):
2     """
3     Returns where highdf has significant difference over the lowdf features
4     """
5     count1=highdf.profile.str.split(expand=True).stack().value_counts()
6     c1=count1.to_frame()
7     c1 = c1.rename(columns= {0: 'count1'})
8     c1['max1']=c1['count1'].max()
9     c1['rate1'] = c1.apply(lambda x: \
10         round(float(x['count1'])/float(x['max1']) * 100.0,4), axis=1)
11     count2=lowdf.profile.str.split(expand=True).stack().value_counts()
12     c2=count2.to_frame()
13     c2 = c2.rename(columns= {0: 'count2'})
14     c2['max2']=c2['count2'].max()
15     c2['rate2'] = c2.apply(lambda x: \
16         round(float(x['count2'])/float(x['max2']) * 100.0,4), axis=1)
17     c3=c1.join(c2)
18     c3.fillna(0,inplace=True)
19     c3['difference']=c3.apply(lambda x: x['rate1']-x['rate2'], axis=1)
20     highrates=c3[((c3.rate1>threshold) & (c3.rate2<threshold) \
21         & (c3.difference>threshold))].difference.sort_values(ascending=False)
22     return highrates

```

Figure 11-54 *Function to Evaluate Dataframe Profile Differences*

Note Be sure to go online and research anything you do not fully understand about working with dataframes. They are a foundational component that you will need.

Figure 11-55 shows how to carve out the items in your own cluster that showed crashes, as well as the items that did not. Now you can seek a comparison of what is more likely to appear on crashed devices.

```

1 df12_crashed=df12[df12.crashed==1.0]
2 df12_nocrash=df12[df12.crashed==0.0]

```

Figure 11-55 *Splitting Crash and Noncrash Entries in a Cluster*

Using these items and your new function, you can determine what is most different in your cluster between the devices that showed failures and the devices that did not (see Figure 11-56).

```

1 get_cluster_diffs(df12_crashed,df12_nocrash,40)

```

cp_7937g	58.1781
cp_7925g_ex_k9	43.1677
clear_channel_t3_e3_with_integrated_csu_dsu	40.3209
nm_1t3_e3	40.3209

Figure 11-56 *Differences in Crashes Versus Noncrashes in the Cluster*

Notice that there are four features in your cluster that show up with 40% higher frequency on the crashed devices. Some of these are IP phones, which indicates that the routers are also performing voice functionality. This is not a surprise. Recall that you chose your first device using an fxo port, which is common for voice communications in networks.

Because this is only within your cluster, make sure that you are not basing your analysis on outliers by checking the entire set that you were using. For these top four features, you can zoom out to look at all your devices in the dataframe to see if there are any higher associations to crashes by using a Python loop (see Figure 11-57).

```

1 querylist=['cp_7937g','cp_7925g_ex_k9','nm_1t3_e3',\
2           'clear_channel_t3_e3_with_integrated_csu_dsu']
3 for feature in querylist:
4     df_filter=df4.copy()
5     df_filter=df_filter[df_filter.profile.str.contains(feature)].copy()
6     dcheck=dict(df_filter.crashed.value_counts())
7     print("Feature " + feature + " : " + \
8           str(round(float(dcheck[1.0])/(float(dcheck[0.0]))*100)))

```

```

Feature cp_7937g : 8.0
Feature cp_7925g_ex_k9 : 5.0
Feature nm_1t3_e3 : 20.0
Feature clear_channel_t3_e3_with_integrated_csu_dsu : 17.0

```

Figure 11-57 *Crash Rate per Component*

You can clearly see that the highest incidence of crashes in routers with fxo ports is associated with the T3 network module. For the sake of due diligence, check the clusters where this module shows up. Figure 11-58 illustrates where you determine that the module appears in both clusters 1 and 3.

```

1 feat="nm_1t3_e3"
2 df_t3=df4[df4.profile.str.contains(feat)].copy()
3 df_t3.kcluster.value_counts()

3    10
1     8

```

Figure 11-58 Cluster Segmentation of a Single Feature

In Figure 11-59, however, notice that only the devices in cluster 3 show crashes with this module. Cluster 1 does not show any crashes, although Cluster 1 does have routers that are using this module. This module alone may not be the cause.

```

1 df_t3_1=df_t3[df_t3.kcluster==1].copy()
2 df_t3_3=df_t3[df_t3.kcluster==3].copy()
3 print(df_t3_1.crashed.value_counts())
4 print(df_t3_3.crashed.value_counts())

0.0    8
Name: crashed, dtype: int64
0.0    7
1.0    3
Name: crashed, dtype: int64

```

Figure 11-59 Crash Rate per Cluster by Feature

This means you can further narrow your focus to devices that have this module and fall in cluster 3 rather than in cluster 1. You can use the `diffs` function one more time to determine the major differences between cluster 3 and cluster 1. Figure 11-60 shows how to look for items in cluster 3 that are significantly different than in cluster 1.

```

1 get_cluster_diffs(df12,df10,95)

ospf_area_authentication          98.5316
ospf_authentication_ios           98.5316
ospf_route_redistribution          97.6505
tracking_the_line_protocol_state_of_an_interface_ios  96.7695
vxml_version_2                    95.6726
15_3_3_m5                         95.6497
command_scheduler                  95.5257
vxml_audioerror                    95.5257
modem_relay_support_on_voip_platforms 95.4064
modem_relay                        95.4064

```

Figure 11-60 Cluster-to-Cluster Differences

This is where you stop the data science part and put on your SME hat again. You used machine learning to find an area of your network that is showing a higher propensity to crash, and you have details about the differences in hardware, software, and configuration

between those devices. You can visually show these differences by using dimensionality reduction. You can get a detailed evaluation of the differences within and between clusters by examining the data from the clusters. For your next steps, you can go many directions:

- Because the software version has shown up as a major difference, you could look for software defects that cause crash in that version.
- You could continue to filter and manipulate the data to find more information about these crashes.
- You could continue to filter and manipulate the data to find more information about other device hotspots.
- You could build automation and service assurance systems to bring significant cluster differences and known crash rates to your attention automatically.

Note In case you are wondering, the most likely cause of these crashes is related to two of the cluster differences that you uncovered in Figure 11-60—in particular, the 15.3(3)M5 software version with the vXML capabilities. There are multiple known bugs in that older release for vXML. Cisco TAC can help with the exact bug matching, using additional device details and the decoding tools built by Cisco Services engineers. Validation of your machine learning findings using SME skills from you, combined with Cisco Services, should be part of your use-case evaluation process.

When you complete your SME evaluation, you can come back to the search tools that you created here and find more issues like the one you researched in this chapter. As you use these methods more and more, you will see the value of building an automated system with user interfaces that you can share with your peers to make their jobs easier as well. The example in this chapter involves network device data, but this method can uncover things for you with any data.

Summary

It may be evident to you already, but remember that much of the work for network infrastructure use cases is about preparing and manipulating data. You may have already noted that many of the algorithms and visualizations are very easy to apply on prepared data. Once you have prepared data, you can try multiple algorithms. Your goal is not to find the perfect algorithmic match but to uncover insights to help yourself and your company.

In this chapter, you have learned how to use modeled network device data to build a detailed search interface. You can use this search and filtering interface for exact match searches or machine learning–based similarity matches in your own environment. These search capabilities are explained here with network devices, but the concepts apply to anything in your environment that you can model with a descriptive text.

You have also learned how to develop clustered representations of devices to explore them visually. You can share these representations with stakeholders who are not skilled in analytics so that they can see the same insights that you are finding in the data. You know how to slice, dice, dig in, and compare the features of anything in the visualizations. You can turn your knowledge so far into a full analytics use case by building a system that allows your users to select their own data to appear in your visualizations; to do so, you need to build your analysis components to be dynamic enough to draw labels from the data.

This is the last chapter that focuses on infrastructure metadata only. Two chapters of examining static information—Chapter 10 and this chapter—should give you plenty of ideas about what you can build from the data that you can access right now. Chapter 12, “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry,” moves into the network operations area, examining event-based telemetry. In that chapter, you will look at what you can do with syslog telemetry from a control plane protocol.

Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry

This chapter moves away from working with static metadata and instead focuses on working with telemetry data sent to you by devices. *Telemetry data* is data sent by devices on regular, time-based intervals. You can use this type of data to analyze what is happening on the control plane. Depending on the interval and the device activity, you will find that the data from telemetry can be very high volume. Telemetry data is your network or environment telling you what is happening rather than you having to poll for specific things.

There are many forms of telemetry from networks. For example, you can have memory, central processing unit (CPU), and interface data sent to you every five seconds. Telemetry as a data source is growing in popularity, but the information from telemetry may or may not be very interesting. Rather than use this point-in-time counter-based telemetry, this chapter uses a very popular telemetry example: syslog.

By definition, *syslog* is telemetry data sent by components in timestamped formats, one message at a time. Syslog is common, and it is used here to show event analysis techniques. As the industry is moving to software-centric environments (such as software-defined networking), analyzing event log telemetry is becoming more critical than ever before.

You can do syslog analysis with a multitude of standard packages today. This chapter does not use canned packages but instead explores some raw data so that you can learn additional ways to manipulate and work with event telemetry data. Many of the common packages work with filtering and data extraction, as you already saw in Chapter 10, “Developing Real Use Cases: The Power of Statistics,” and Chapter 11, “Developing Real Use Cases: Network Infrastructure Analytics”—and you probably already use a package or two daily. This chapter goes a step further than that.

Data for This Chapter

Getting from raw log messages to the data here involves the Cisco pipeline process, which is described in Chapter 9, “Building Analytics Uses Cases.” There are many steps and different options for ingesting, collecting, cleaning, and parsing.

Depending on the types of logs and collection mechanisms you use, your data may be ready to go, or you may have to do some cleaning yourself. This chapter does not spend time on those tasks. The data for this chapter was preprocessed, anonymized, and saved as a file to load into Jupyter Notebook.

With the preprocessing done for this chapter, syslog messages are typically some variation of the following format:

HOST - TIMESTAMP - MESSAGE_TYPE: MESSAGE_DETAIL

For example, a log from a router might look like this:

Router1 Jan 2 14:55:42.395: %SSH-5-ENABLED: SSH 2.0 has been enabled

In preparation for analysis, you need to use common parsing and cleaning to split out the data as you want to analyze it. Many syslog parsers do this for you. For the analysis in this chapter, the message is split as follows:

HOST - TIMESTAMP - SEVERITY - MESSAGE_TYPE - MESSAGE

So that you can use your own data to follow along with the analysis in this chapter, a data set was prepared in the following way:

1. I collected logs to represent 21 independent locations of a fictitious company. These logs are from real networks’ historical data.
2. I filtered these logs to Open Shortest Path First (OSPF), so you can analyze a single control plane routing protocol.
3. I anonymized the logs to make them easier to follow in the examples.
4. I replaced any device-specific parts of the logs into a new column in order to identify common logs, regardless of location.
5. I provided the following data for each log message:
 - a. The original host that produced the log
 - b. The business, which is a numerical representation for 1 of the 21 locations
 - c. The time, to the second, of when the host produced the log
 - d. The log, split into type, severity, and log message parts
 - e. The log message, cleaned down to the actual structure with no details
6. I put all the data into a pandas dataframe that has a time-based index to load for analysis in this chapter.

Log analysis is critically important to operating networks, and Cisco has hundreds of thousands of human hours invested in building log analysis. Some of the types of analysis that you can do with Python is covered in this chapter.

OSPF Routing Protocols

OSPF is a routing protocol used to set up paths for data plane traffic to flow over networks. OSPF is very common, and the telemetry instrumentation for producing and sending syslog is very mature, so you can perform a detailed analysis from telemetry alone. OSPF is an interior gateway protocol (IGP), which means it is meant to be run on bounded locations and not the entire Internet at once (as Border Gateway Protocol [BGP] is meant to do). You can assume that each of your 21 locations is independent of the others.

Any full analysis in your environment also includes reviewing the device-level configuration and operation. This is the natural next step in addressing any problem areas that you find doing the analysis in this chapter. Telemetry data tells you what is happening but does not always provide reasons *why* it is happening. So let's get started looking at syslog telemetry for OSPF across your locations to see where to make improvements.

Remember that your goal is to learn to build atomic parts that you can assemble over time into a growing collection of analysis techniques. You can start building this knowledge base for your company. Cisco has thousands of rules that have been developed over the years by thousands of engineers. You can use the same analysis themes to look at any type of event log data. If you have access to log data, try to follow along to gain some deliberate practice.

Non-Machine Learning Log Analysis Using pandas

Let's start this section with some analysis typically done by syslog SMEs, without using machine learning techniques. The first thing you need to do is load the data. In Chapters 10 and 11 you learned how to load data from files, so in this chapter we can get right to examining what has been loaded in Figure 12-1. (The loading command is shown later in this chapter.)

1	<code>df.columns</code>
<pre>Index([u'host', u'business', u'severity', u'cleaned_message', u'zip_code', u'city', u'state', u'message_type', u'message'], dtype='object')</pre>	

Figure 12-1 Columns in the Syslog Dataframe

Do you see the columns you expect to see? The first thing that you may notice is that there isn't a timestamp column. Without time awareness, you are limited in what you can do. Do not worry: It is there, but it is not a column; rather, it is the index of the dataframe, which you can set when you load the dataframe, as shown in Figure 12-2.

```

1 df=pd.read_csv(filename, parse_dates=["datetime"], index_col=["datetime"])
2 df.index

DatetimeIndex(['2018-04-12 11:59:56', '2018-04-12 11:59:51',
               '2018-04-12 11:59:46', '2018-04-12 11:59:41',
               '2018-04-12 11:59:38', '2018-04-12 11:59:36',
               '2018-04-12 11:59:31', '2018-04-12 11:59:29',
               '2018-04-12 11:59:26', '2018-04-12 11:59:21',
               ...
               '2018-04-06 13:02:42', '2018-04-06 13:02:42',
               '2018-04-06 13:02:37', '2018-04-06 13:02:32',
               '2018-04-06 13:01:51', '2018-04-06 13:01:51',
               '2018-04-06 13:01:04', '2018-04-06 13:00:35',
               '2018-04-06 13:00:11', '2018-04-06 13:00:06'],
              dtype='datetime64[ns]', name=u'datetime', length=1582816, freq=None)

```

Figure 12-2 *Creating a Time Index from a Timestamp in Data*

The Python pandas library that you used in the Chapters 10 and 11 also provides the capability to parse dates into a very useful index with time awareness. You have the syslog timestamp in your data file as a datetime column, and when you load the data for analysis, you tell pandas to use that column as the index. You can also see that your data is from one week—from Friday, April 6, to Thursday, April 12—and you have more than 1.5 million logs for that time span. Because you have an index based on time, you can easily plot the count of log messages that you have over the time that you are analyzing, as shown in Figure 12-3.

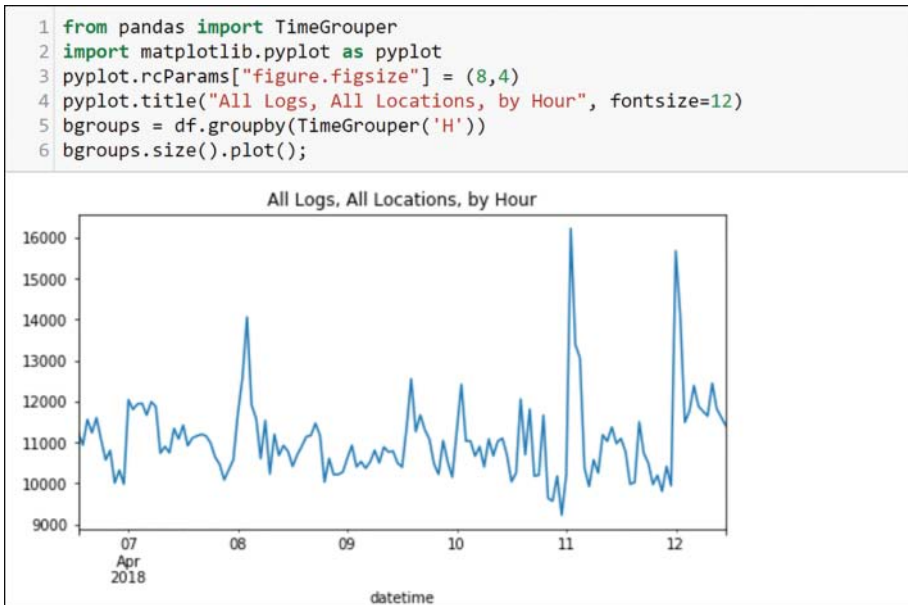


Figure 12-3 *Plot of Syslog Message Counts by Hour*

pandas **TimeGrouper** allows you to segment by time periods and plot the counts of events that fall within each one by using the size of each of those groups. In this case, H was used to represent hourly. Notice that significant occurrences happened on April 8, 11, and 12. In Figure 12-4, look at the severity of the messages in your data to see *how* significant events across the entire week were. Severity is commonly the first metric examined in log analysis.

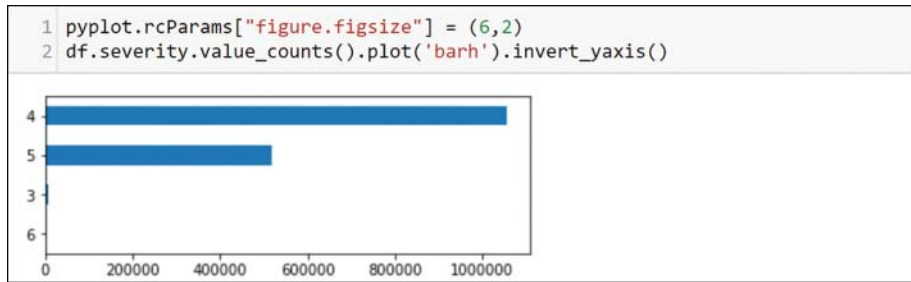


Figure 12-4 Message Severity Counts

Here you use **value_counts** to get the severity counts and add plotting of the data to get the bar chart. The default plotting behavior is bottom to top—or least to most—and you can use **invert_axis** to reverse the plot. When you plot all values of severity from your OSPF data, notice that all the messages have severity between 3 and 6. This means there aren't any catastrophic issues right now. You can see from the standard syslog severities in Table 12-1 that there are a few errors and lots of warnings and notices, but nothing is critical.

Table 12-1 Standard Syslog Severities

Message Level	Severity
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

The lack of emergency, alert, or critical does not mean that you do not have problems in your network. It just means that nothing in the OSPF software on the devices is severely broken anywhere. Do not forget that you filtered to OSPF data only. You may still find issues if you focus your analysis on CPU, memory, or hardware components. You can perform that analysis with the techniques you learn in this chapter.

At this point, you should be proficient enough with pandas to identify how many hosts are sending these messages or how many hosts there are per location. If you want to know those stats about your own log, you can use `filter` with the square brackets and then choose the host column to show `value_counts()`.

Noise Reduction

A very common use case for log analysis is to try to reduce the high volume of data by eliminating logs that do not have value for the analysis you want to do. That was already done to some degree by just filtering the data down to OSPF. However, even within OSPF data, there may be further noise that you can reduce. Let's check.

In Figure 12-5, look at the simple counts by message type.

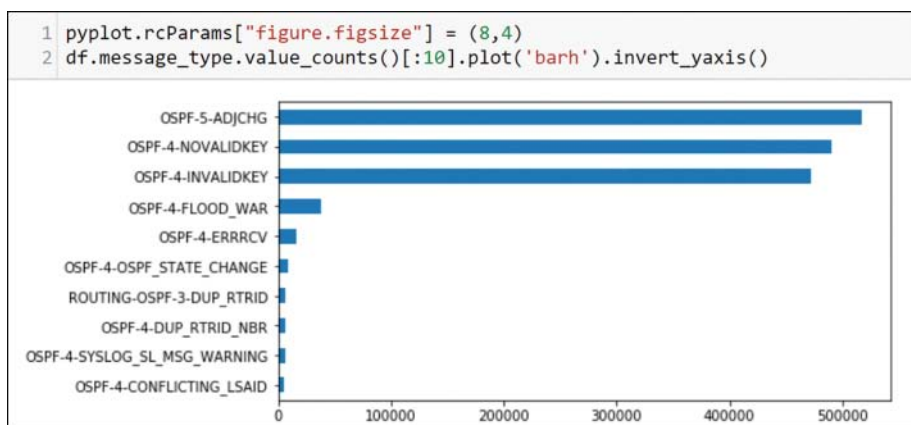


Figure 12-5 Syslog Message Type Counts

You immediately see a large number of three different message types. Because you can see a clear visual correlation between the top three, you may be using availability bias to write a story that some problem with keys is causing changes in OSPF adjacencies. Remember that correlation is not causation. Look at what you can prove. If you look at the two of three that seem to be related by common keyword, notice from the filter in Figure 12-6 that they are the only message types that contain the keyword `key` in the `message_type` column.

```

1 df[df.message_type.str.contains("KEY")].message_type.value_counts()
OSPF-4-NOVALIDKEY      490421
OSPF-4-INVALIDKEY      472434
Name: message_type, dtype: int64

```

Figure 12-6 *Regex Filtered key Messages*

If you put on your SME hat and consider what you know, you realize that you know that keys are used to form authenticated OSPF adjacencies. These top three message types may indeed be related. If you take the same filter and change the values on the right of the filter, as shown in Figure 12-7, you can plot which of your locations is exhibiting a problem with OSPF keys.

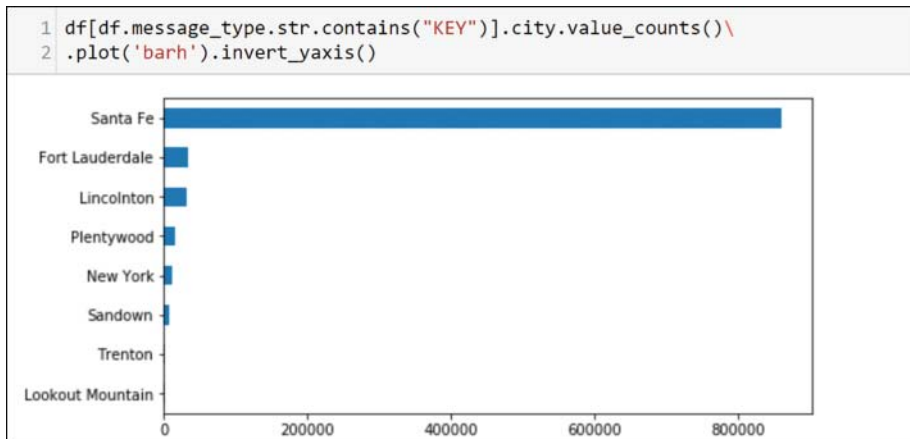


Figure 12-7 *Key Messages by Location*

Notice that the Santa Fe location is significantly higher than the other locations for this message type. Figure 12-7 shows the results filtered down to only this message type and a plot of the value counts for the city that had these messages. It seems like something is going on in Santa Fe because well over half of the 1.58 million messages are coming from there. Overall, this warning level problem is showing up in 8 of the 21 locations. Figure 12-8 shows how to look at Santa Fe to see what is happening there with OSPF.

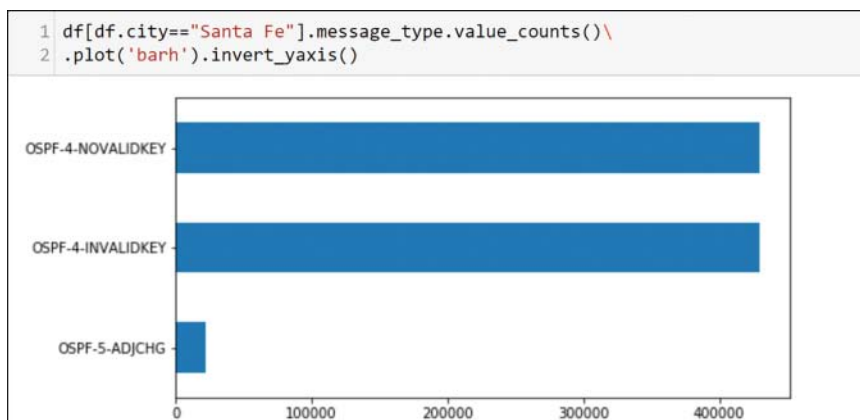


Figure 12-8 *Message Types in the Santa Fe Location*

You have already found that most of your data set is coming from this one location. Do you notice anything else here? A high number of adjacency changes is not correlating with the **key** messages. There are a few, but there are not nearly enough to show direct correlation. There are two paths to take now:

1. Learn more about these **key** messages and what is happening in Santa Fe.
2. Find out where the high number of adjacency changes is happening.

If you start with the **key** messages, a little research informs you that this is a misconfiguration of OSPF MD5 authentication in routers. In some cases, adjacencies will still form, but the routers have a security flaw that should be corrected. For a detailed explanation and to learn why adjacencies may still form, see the Cisco forum at <https://supportforums.cisco.com/t5/wan-routing-and-switching/asr900-ospf-4-novalidkey-no-valid-authentication-send-key-is/td-p/2625879>.

Note These locations and the required work have been added to Table 12-2 at the end of the chapter, where you will gather follow-up items from your analysis. Don't forget to address these findings while you go off to chase more butterflies.

Using your knowledge of filtering, you may decide to determine which of the **key** messages do not result in adjacency changes and filter your data set down to half. You know the cause, and you can find where the messages are not related to anything else. Now they are just noise. Distilling data down in this way is a very common technique in event log analysis. You find problems, create some task from them, and then whittle down the data set to find more.

Finding the Hotspots

Recall that the second problem is to find out where the high number of adjacency changes is happening. Because you have hundreds of thousands of adjacency messages, they might be associated to a single location, as the keys were. Figure 12-9 shows how to

examine any location that has generated more than 100,000 messages this week and plot them in the context of each other, using a loop.

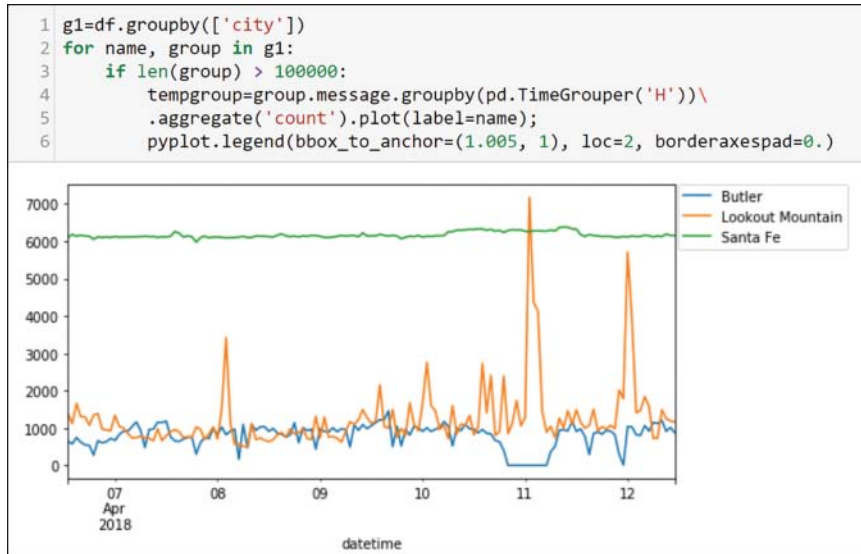


Figure 12-9 *Syslog High-Volume Producers*

pandas provides the capability to group by time periods, using **TimeGrouper**. In this case, you are double grouping. First, you are grouping by city so that you have one group for each city in the data. For each of those cities, you run through a loop and group the time by hour, aggregate the count of messages per hour, and plot the results of each of them.

You can clearly see the Santa Fe messages at a steady rate of over 6000 per hour. Those were already investigated, and you know the problem there is with **key** messages. However, there are two other locations that are showing high counts of messages: Lookout Mountain and Butler. Given what you have learned in the previous chapters, you should easily see how to apply anomaly detection to the daily run rate here. These spikes show up as anomalies. The method is the same as the method used at the end of Chapter 10, and you can set up systems to identify anomalies like this hour by hour or day by day. Those systems feed your activity prioritization work pipelines with these anomalies, and you do not have to do these steps and visual examination again.

You can also see something else of note that you want to add to your task list for later investigation: You appear to have a period in Butler, around the 11th, during which you were completely blind for log messages. Was that a period with no messages? Were there messages but the messages were not getting to your collection servers? Is it possible that the loss of messages correlates to the spike at Lookout Mountain around the same time? Only deeper investigation will tell. At a minimum, you need to ensure consistent flow of telemetry from your environment, or you could miss critical event notifications. This action item goes on your list.

Now let's look at the Lookout Mountain and Butler locations. Figure 12-10 shows the Lookout Mountain information.

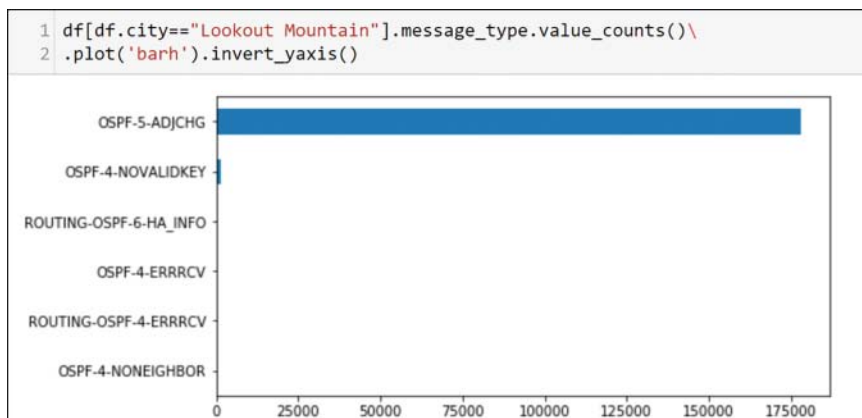


Figure 12-10 Lookout Mountain Message Types

You clearly have a problem with adjacencies at Lookout Mountain. You need to dig deeper to see why there are so many of these changes at this site. The spikes shown in Figure 12-9 clearly indicate that something happened three times during the week. You can add this investigation to your task list. There seem to be a few error warnings, but nothing else stands out here. There are no smoking guns. Sometimes OSPF adjacency changes are part of normal operations when items at the edge attach and detach intentionally. You need to review the intended design and the location before you make a determination.

Figure 12-11 shows how to finish your look at the top three producers by looking at Butler.

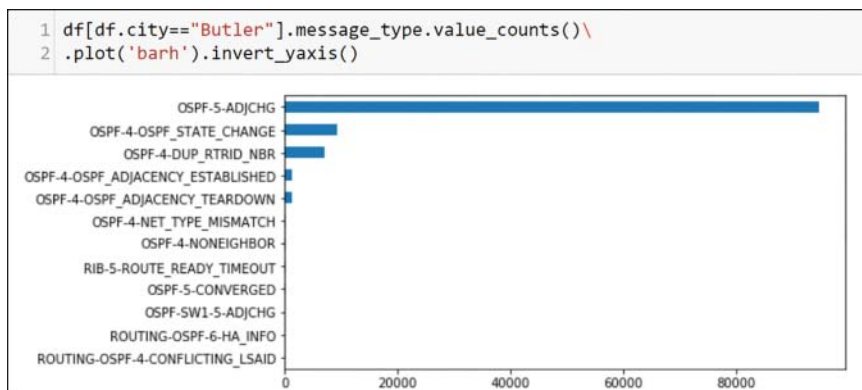


Figure 12-11 Butler Message Types

Now you can see something interesting. Butler also has many of the adjacency changes, but in this case, many other indicators raise flags for network SMEs. If you are a network SME, you know the following:

<https://t.me/learningnets>

- OSPF router IDs must be unique (line 3).
- OSPF network types must match (line 6).
- OSPF routes are stored in a routing information base (RIB; line 8)
- OSPF link-state advertisements (LSAs) should be unique in the domain (line 12).

There appear to be some issues in Butler, so you need to add this to the task list. Recall that this event telemetry is about the network telling you that there is a problem, and it has done that. You may or may not be able to diagnose the problem based on the telemetry data. In most cases, you will need to visit the devices in the environment to investigate the issue.

Ultimately, you may have enough data in your findings to create labels for sets of conditions, much like the crash labels used previously. Then you can use labeled sets of conditions to build inline models to predict behavior, using supervised learning classifier models.

There is much more that you can do here to continue to investigate individual messages, hotspots, and problems that you find in the data. You know how to sort, filter, plot, and dig into the log messages to get much of the same type of analysis that you get from the log analysis packages available today. You have already uncovered some action items.

This section ends with a simple example of something that network engineers commonly investigate: route flapping. Adjacencies go up, and they go down. You get the **ADJCHG** message when adjacencies change state between up and down. Getting many adjacency messages indicates many up-downs, or flaps. You need to evaluate these messages in context because sometimes connect/disconnect may be normal operation. Software-defined networking (SDN) and network functions virtualization (NFV) environments may have OSPF neighbors that come and go as the software components attach and detach. You need to evaluate this problem in context. Figure 12-12 shows how to quickly find the top flapping devices.

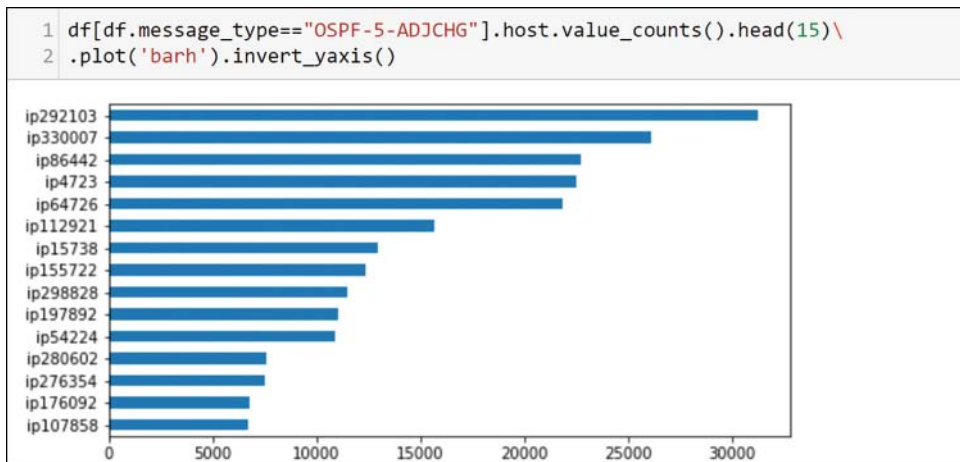


Figure 12-12 OSPF Adjacency Change, Top N

If you have a list of the hosts that should or should not be normally going up/down, you can identify problem areas by using dataframe filtering with the `isin` keyword and a list of those hosts.

For now we will stop looking at the sorting and filtering that SMEs commonly use and move on to some machine learning techniques to use for analyzing log-based telemetry.

Machine Learning–Based Log Evaluation

The preceding section spends a lot of time on message type. You will typically review the detailed parts of log messages only after the message types lead you there. With the compute power and software available today, this does not have to be the case. This section shows how to use machine learning to analyze syslog. It moves away from the message type and uses the more detailed full message so you can get more granular. Figure 12-13 shows how you change the filter to show the possible types of messages in your data that relate to the single message type of adjacency change.

```
1 df[df.message_type=="OSPF-5-ADJCHG"].cleaned_message.value_counts()\n2 .count()\n54
```

Figure 12-13 *Variations of Adjacency Change Messages*

`cleaned_message` is a column in the data that was stripped of specific data, and you can see 54 variations. Notice the top 4 counts and the format of `cleaned_message` in Figure 12-14.

```
1 df[df.message_type=="OSPF-5-ADJCHG"].cleaned_message.value_counts()\n2 .head(4)\n\nProcess PROC Nbr HOST on INT from LOADING to FULL Loading Done\n      212923\nProcess PROC Nbr HOST on INT from FULL to DOWN Neighbor Down: Dead timer expire\n      76394\nd\nProcess PROC Nbr HOST on INT from FULL to DOWN Neighbor Down: BFD node down\n      38373\nProcess PROC Nbr HOST on INT from DOWN to DOWN Neighbor Down: Ignore timer expi\n      27058\nred\nName: cleaned_message, dtype: int64
```

Figure 12-14 *Top Variations of OSPF Adjacency Change Types*

With 54 cleaned variations, you can see why machine learning is required for analysis. This section looks at some creative things you can do with log telemetry, using machine learning techniques combined with some creative scripting.

First, Figure 12-15 shows a fun way for you to give stakeholders a quick visual summary of a filtered set of telemetry.

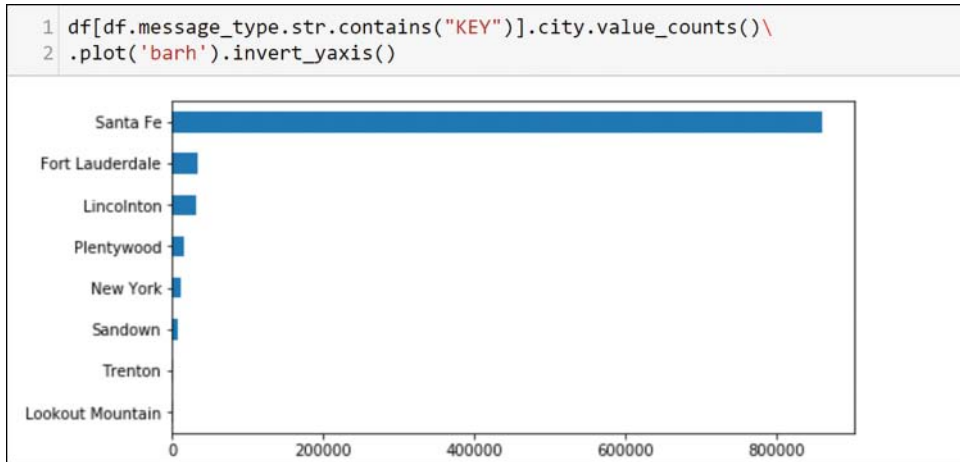


Figure 12-15 *Santa Fe key Message Counts*

As an SME, you know that this is related to the 800,000 **key** messages at this site. You can show this diagram to your stakeholders and tell them that these are **key** messages. Alternatively, you could get creative and start showing visualizations, as described in the following section.

Data Visualization

Let's take a small detour and see how to make a word cloud for Santa Fe to show your stakeholders something visually interesting. First, you need to get counts of the things that are happening in Santa Fe. In order to get a normalized view across devices, you can use the `cleaned_message` column. How you build the code to do this depends on the types of logs you have. Here is a before-and-after example that shows the transformation of the detailed part of the log message as transformed for this chapter:

Raw log format:

```
'2018 04 13 06:32:12 somedevic OSPF-4-FLOOD_WAR 4 Process 111 flushes
LSA ID 1.1.1.1 type-2 adv-rtr 2.2.2.2 in area 3.3.3.3'
```

Cleaned message portion:

'Process PROC flushes LSA ID HOST type-2 adv-rtr HOST in area AREA'

To set up some data for visualizing, Figure 12-16 shows a function that generates an interesting set of terms across all the cleaned messages in a dataframe that you pass to it.

```

1 def termdict(df, droplist, cutoff):
2     term_counts=dict(df.cleaned_message.str.split(expand=True)\
3                     .stack().value_counts())
4     keepthese={}
5     for k,v in term_counts.items():
6         if v < (len(df)*(1-cutoff)): # Low counts
7             if v > (len(df)*cutoff): # high counts
8                 if k not in droplist:
9                     keepthese.setdefault(k,v)
10    return keepthese

```

Figure 12-16 *Python Function to Generate Word Counts*

This function is set up to make a dictionary of terms from the messages and count the number of terms seen across all messages in the `cleaned_message` column of the dataframe. The `split` function splits each message into individual terms so you can count them. Because there are many common words, as well as many singular messages that provide rare words, the function provides a cutoff option to drop the very common and very rare words relative to the length of the dataframe that you pass to the function. There is also a drop list capability to drop out uninteresting words. You are just looking to generalize for a visualization here, so some loss of fidelity is acceptable.

You have a lot of flexibility in whittling down the words that you want to see in your word cloud. Figure 12-17 shows how to set up this list, provide a cutoff of 5%, and generate a dictionary of the remaining terms and a counts of those terms.

```

1 droplist=['INT', 'PROC', 'HOST', 'on', 'from', 'interface', \
2          'Nbr', 'Process', 'to', '0']
3 cutoff=.05
4 df2=df[df.city=="Santa Fe"]
5 print("Messages: " + str(len(df2)))
6 wordcounts=termdict(df2, droplist, cutoff)
7 print("Words in Dictionary: " + str(len(wordcounts)))

```

Messages: 881880
Words in Dictionary: 10

Figure 12-17 *Generating a Word Count for a Location*

Now you can filter to a dataframe and generate a dictionary of word count. The dictionary returned here is only 10 words. You can visualize this data by using the Python `wordcloud` package, as shown in Figure 12-18.



Figure 12-18 *Word Cloud Visual Summary of Santa Fe*

Now you have a way to see visually what is happening within any filtered set of messages. In this case, you looked at a particular location and summed up more than 800,000 messages in a simple visualization. Such visualizations can be messy, with lots of words from data that is widely varied, but they can appear quite clean when there is an issue that is repeating, as in this case. Recall that much analytics work is about generalizing the current state, and this is a way to do so visually. This is clearly a case of a dominant message in the logs, and you may use this visual output to determine that you need to reduce the noise in this data by removing the messages that you already know how to address.

Cleaning and Encoding Data

Word clouds may not have high value for your analysis, but they can be powerful for showing stakeholders what you see. We will discuss word clouds further later in this chapter, but for now, let's move to unsupervised machine learning techniques you can use on your logs.

You need to encode data to make it easier to do machine learning analysis. Figure 12-19 shows how to begin this process by making all your data lowercase so that you can recognize the same data, regardless of case. (Note that the word cloud in Figure 12-18 shows key and Key as different terms.)

```

1 df['cleaned_message2']=df['cleaned_message2'].\
2   apply(lambda x: str(x).lower())
3
4 fix1={'metric \d+':'metric n', 'area \d+':'area n'}
5 df['cleaned_message2']=df.cleaned_message2.replace(fix1, regex=True)
6 df=df[~(df.cleaned_message2.str.contains('host/32'))]

```

Figure 12-19 *Manipulating and Preparing Message Data for Analysis*

Something new for you here is the ability to replace terms in the strings by using a Python dictionary with regular expressions. In line 4, you create a dictionary of things you want to replace and things you want to use as replacements. The key/value pairs in the dictionary are separated by commas. You can add more pairs and run your data through the code in lines 4 and 5 as much as you need in order to clean out any data in the messages. Be careful not to be too general on the regular expressions, or you will remove more than you expected.

Do you recall the tilde character and its use? In this example, you have a few messages that have the forward slash in the data. Line 6 is filtering to the few messages that have that data, inverting the logic with a tilde to get all messages that do *not* have that data, and providing that as your new dataframe. You already know that you can create new dataframes with each of these steps if you desire. You made copies of the dataframes in previous chapters. In this chapter, you can keep the same dataframe and alter it.

Note Using the same dataframe can be risky because once you change it, you cannot recall it from a specific point. You have to run your code from the beginning to fix any mistakes.

Figure 12-20 shows how to make a copy for a specific analysis and generate a new dataframe with the authentication messages removed. In this case, you want to have both a filtered dataframe and your original data available.

```
1 tdf=df[~(df.cleaned_message2.str.contains\
2     ('no valid authentication send key'))].copy()
3 tdf=tdf[~(tdf.cleaned_message2.str.contains\
4     ('key id 0 received on interface int'))]
5 len(tdf)

619688
```

Figure 12-20 *Filtering a Dataframe with a Tilde*

Because there was so much noise related to the authentication **key** messages, you now have less than half of the original dataframe. You can use this information to see what is happening in the other cities, but first you need to summarize by city. Figure 12-21 shows how to group the city and the newly cleaned messages by city to come up with a complete summary of what is happening in each city.

```
1 tdf['logprofile']=tdf[['city','cleaned_message2']]\
2 .groupby(['city'])['cleaned_message2'].transform(lambda x: ' '.join(x))
3 tdf.drop_duplicates('city', inplace=True)
4 len(tdf)

21
```

Figure 12-21 *Creating a Log Profile for a City*

In this code, you use the Python `join` function to join all the messages together into a big string separated by a space. You can ensure that you have only your 21 cities by dropping any duplicates in line 3; notice that the length of the resulting dataframe is now only 21 cities long. A single city profile can now be millions of characters long, as shown in Figure 12-22.

```
1 len(tdf.iloc[2].logprofile)
12572075
```

Figure 12-22 *Character Length of a Log Profile*

As the Santa Fe word cloud example showed a unique signature, you hope to find out something that uniquely identifies the locations so you can compare them to each other by using machine learning or visualizations. You can do this by using text analysis. Figure 12-23 shows how to tokenize the full log profiles into individual terms.

```
1 import nltk
2 tdf['tokens'] = tdf.apply(lambda row: nltk.word_tokenize\
3                          (row['logprofile']), axis=1)
```

Figure 12-23 *Tokenizing a Log Profile*

Once you tokenize a log profile, you have lists of tokens that describe all the messages. Having high numbers of the same terms is useful for developing word clouds and examining repeating messages, but it is not very useful for determining a unique profile for an individual site. You can fix that by removing the repeating words and generating a unique signature for each site, as shown in Figure 12-24.

```
1 tdf['unique_tokens']=tdf['tokens'].apply(lambda x: list(set(x)))
2 tdf['token_string']=tdf['unique_tokens'].apply(lambda x: ' '.join(x))
3 tdf.iloc[2].token_string

'available bad process nbr received dead down done nsr check forced exstart loa
ding from bfd unknown strict-mode invalid id to init too lls backbone length tr
ansitioning : proc retransmissions node full description exchange adjacency hos
t packet interface expired reset on tlv database many area timer ignore int nei
ghbor checksum detached or mismatched'
```

Figure 12-24 *Unique Log Signature for a Location*

Python sets show only unique values. In line 1, you reduce each token list to a set and then return a list of unique tokens only. In line 3, you join these back to a string, which you can use as a unique profile for a site. You can see that this looks surprisingly like a fingerprint from Chapter 11—and you can use it as such. Figure 12-25 shows how to use `CountVectorizer` to encode these profiles.


```

1 from sklearn.feature_extraction.text import CountVectorizer
2 vectorizer = CountVectorizer()
3 word_transform = vectorizer.fit_transform(tdf['token_string'])
4 X_matrix = word_transform.todense()

```

Figure 12-25 *Encoding Logs to Numerical Vectors*

Just as in Chapter 11, you transform the token strings into an encoded matrix to use for machine learning. Figure 12-26 shows how to evaluate the principal components to see how much you should expect to maintain for each of the components.

```

1 from sklearn.decomposition import PCA
2 pca = PCA(n_components=8)
3 pca.fit(X_matrix)
4 pca.explained_variance_ratio_

```

array([0.1898529 , 0.134998 , 0.09161542, 0.08782205, 0.07726412,
0.07374336, 0.06247707, 0.05877654])

Figure 12-26 *Evaluating PCA Component Options*

Unlike in Chapter 11, there is no clear cutoff here. You can choose three dimensions so that you can still get a visual representation, but with the understanding that it will only provide about 40% coverage for the variance. This is acceptable because you are only looking to get a general idea of any major differences that require your attention. Figure 12-27 shows how to generate the components from your matrix.

```

1 pca_data = PCA(n_components=3).fit_transform(X_matrix)
2 pca1=[]
3 pca2=[]
4 pca3=[]
5 for index, instance in enumerate(pca_data):
6     pca_1, pca_2, pca_3 = pca_data[index]
7     pca1.append(pca_1)
8     pca2.append(pca_2)
9     pca3.append(pca_3)
10 print(len(pca1))
11 print(len(pca2))
12 print(len(pca3))

```

```

21
21
21

```

Figure 12-27 *Performing PCA Dimensionality Reduction*

Note that you added a third component here beyond what was used in Chapter 11. Your visualization is now three dimensional. Figure 12-28 shows how to add these components to the dataframe.

```

1 tdf['pca1']=pca1
2 tdf['pca2']=pca2
3 tdf['pca3']=pca3

```

Figure 12-28 *Adding PCA Components to the Dataframe*

Now that you have the components, you can visualize them. You already know how to plot this entire group, but you don't know how to do it in three dimensions. You can still plot the first two components only. Before you build the visualization, you can perform clustering to provide some context.

Clustering

Because you want to find differences in the full site log profiles, which translate to distances in machine learning, you need to apply a clustering method to the data. You can use the K-means algorithm to do this. The elbow method for choosing clusters was inconclusive here, so you can just randomly choose some number of clusters in order to generate a visualization. You may have picked up in Figure 12-26 that there was no clear distinction in the PCA component cutoffs. Because PCA and default K-means clustering use similar evaluation methods, the elbow plot is also a steady slope downward, with no clear elbows. You can iterate through different numbers of clusters to find a visualization that tells you something. You should seek to find major differences here that would allow you to prioritize paying attention to the sites where you will spend your time. Figure 12-29 shows how to choose three clusters and run through the K-means generation.

```

1 kclusters=3
2 km = KMeans(n_clusters=kclusters,n_init=200,random_state=0)
3 km.fit_predict(X_matrix)
4 tdf['kcluster']=km.labels_.tolist()

```

Figure 12-29 *Generating K-means Clusters and Adding to the Dataframe*

You can copy the data back to the dataframe as a kclusters column, and, as shown in Figure 12-30, slice out three views of just these cluster assignments for visualization.

```

1 df0=tdf[tdf.kcluster==0]
2 df1=tdf[tdf.kcluster==1]
3 df2=tdf[tdf.kcluster==2]

```

Figure 12-30 *Creating Dataframe Views of K-means Clusters*

Now you are ready to see what you have. Because you are generating three dimensions, you need to add additional libraries and plot a little differently, as shown in the plot definition in Figure 12-31.

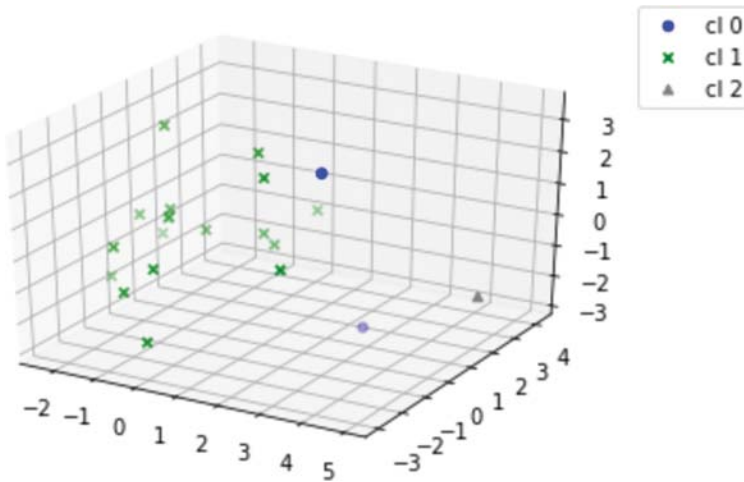
```

1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 fig = plt.figure()
4 ax = fig.add_subplot(111, projection='3d')
5 ax.scatter(df0['pca1'], df0['pca2'], df0['pca3'], s=20, \
6           color='blue', label="cl0")
7 ax.scatter(df1['pca1'], df1['pca2'], df1['pca3'], s=20, \
8           marker="x", color='green', label="cl1")
9 ax.scatter(df2['pca1'], df2['pca2'], df2['pca3'], s=20, \
10          marker="^", color='grey', label="cl2")
11 plt.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
12 plt.show()

```

Figure 12-31 *Scatterplot Definition*

In this definition, you bring in three-dimensional capability by defining the plot a little differently. You plot each of the cluster views using a different marker and increase the size for better visibility. Figure 12-32 shows the resulting plot.

**Figure 12-32** *3D Scatterplot of City Log Profiles*

The three-dimensional scatterplot looks interesting, but you may wonder how much value this has over just using two dimensions. You can generate a two-dimensional definition by using just the first two components, as shown in Figure 12-33.

```

1 plt.scatter(df0['pca1'], df0['pca2'], s=20, \
2           color='blue', label="cl 0")
3 plt.scatter(df1['pca1'], df1['pca2'], s=20, \
4           marker="x", color='green', label="cl 1")
5 plt.scatter(df2['pca1'], df2['pca2'], s=20, \
6           marker="^", color='grey', label="cl 2")
7 plt.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
8 plt.show()

```

Figure 12-33 *2D Scatterplot Definition*

Using your original scatter method from previous chapters, you can choose only the first two components from the dataframe and generate a plot like the one shown in Figure 12-34.

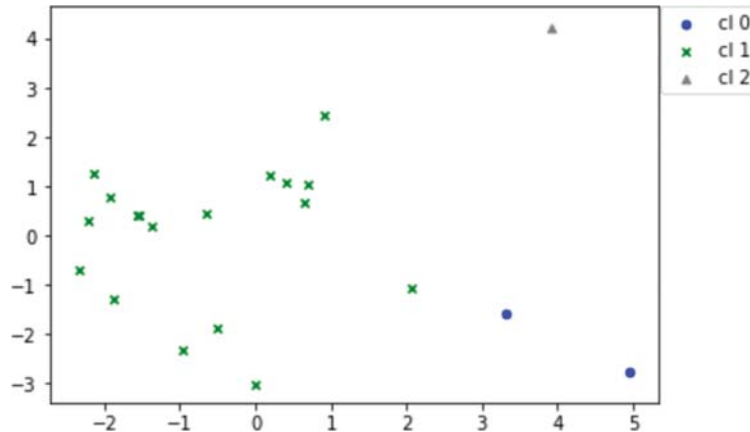


Figure 12-34 2D Scatterplot of City Log Profiles

Notice here that two dimensions appears to be enough in this case to identify major differences in the logs from location to location. It is interesting how the K-means algorithm decided to split the data: You have a cluster of 1 location, another cluster of 2 locations, and a cluster of 18 locations.

More Data Visualization

Just as you did earlier with a single location, you can visualize your locations now to see if anything stands out. You know as an SME that you can just go look at the log files. However, recall that you are building components that you can use again just by applying different data to them. You may be using this data to create visualizations for people who are not skilled in your area of expertise. Figure 12-35 shows how to build a new function for generating term counts per cluster so that you can create word cloud representations.

```

1 def termdict(df, droplist):
2     term_counts=dict(df.logprofile.str.split(expand=True)\
3                     .stack().value_counts())
4     keepthese={}
5     for k,v in term_counts.items():
6         if k not in droplist:
7             keepthese.setdefault(k,v)
8     sorted_x = sorted(keepthese.items(), key=operator.itemgetter(1),\
9                     reverse=True)
10    wordcounts=dict(sorted_x[:30])
11    return wordcounts

```

Figure 12-35 Dictionary to Generate Top 30 Word Counts

This function is very similar to the one used to visualize a single location, but instead of cutting off both the top and bottom percentages, you are filtering to return the top 30 term counts found in each cluster. You still use **droplist** to remove any very common words that may dominate the visualizations. This function allows you to see the major differences so you can follow the data to see where you need to focus your SME attention. Figure 12-36 shows how to use **droplist** and ensure that you have the visualization capability with the word cloud library.

```
1 droplist=['on','from','interface','is','proc','host','int',\
2          'ospf','process','to','nbr']
3 from wordcloud import WordCloud
4 import matplotlib.pyplot as pyplot
```

Figure 12-36 *Using droplist and Importing Visualization Libraries*

You do not have to know the **droplist** items up front. You can iteratively run through some word cloud visualizations and add to this list until you get what you want. Recall that you are trying to get a general sense of what is going on. Nothing needs to be precise in this type of analysis. Figure 12-37 shows how to build the required code to generate the word clouds. You can reuse this code by just passing a dataframe view in the first line.

```
1 whichdf=df0
2 wordcounts=termdict(whichdf,droplist)
3 wordcloud = WordCloud(relative_scaling=1,background_color='white',scale=3,
4                       max_words=400,max_font_size=40, \
5                       normalize_plurals=False)\
6                       .generate_from_frequencies(wordcounts)
7 pyplot.figure(figsize=(8,4))
8 pyplot.imshow(wordcloud)
9 pyplot.axis("off");
```

Figure 12-37 *Generating a Word Cloud Visualization*

Now you can run each of the dataframes through this code to see what a visual representation of each cluster looks like. Cluster 2 is up and to the right on your plot, and it is a single location. Look at that one first. Figure 12-38 shows how to use **value_counts** with the dataframe view to see the locations in that cluster.

```
1 df2.city.value_counts()

Plainville    1
Name: city, dtype: int64
```

Figure 12-38 *Cities in the Dataframe*

This is not a location that surfaced when you examined the high volume messages in your data. However, from a machine learning perspective, this location was singled out into a separate cluster. See the word cloud for this cluster in Figure 12-39.

There are two locations here. You have already learned from previous analysis that Butler had problems, but this is the first time you see Gibson. According to your machine learning approach, Gibson is showing something different from the other clusters, but you know from the previous scatterplot that it is not exactly the same as Butler, though it's close. You can go back to your saved work from the previous non-machine learning analysis that you did to check out Gibson, as shown in Figure 12-42.

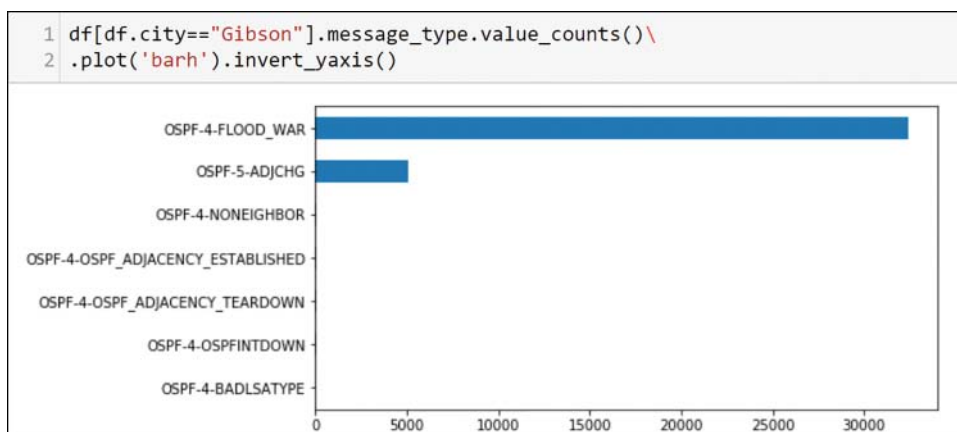


Figure 12-42 Gibson Message Types Top N

Sure enough, Gibson is showing more than 30,000 flood warnings. Due to the noise in your non-machine learning analysis, you did not catch it. As an SME, you know that flooding can adversely affect OSPF environments, so you need to add Gibson to the task list.

Your final cluster is all the remaining 18 locations that showed up on the left side of the plot in cluster 1 (see Figure 12-43).

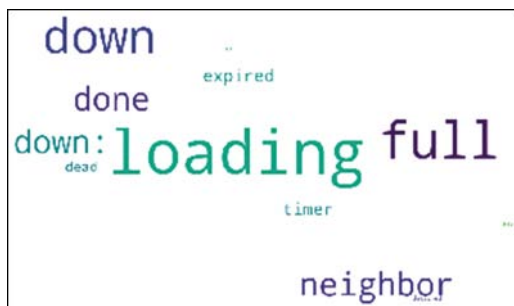


Figure 12-43 Word Cloud for 18 Locations in Cluster 1

Nothing stands out here aside from the standard neighbors coming and going. If you have stable relationships that should not change, then this is interesting. Because you have 18 locations with these standard messages coupled with the loss of information from the

dimensionality reduction, you may not find much more by using this method. You have found two more problem locations and added them to your list. Now you can move on to another machine learning approach to see if you find anything else.

Transaction Analysis

So far, you have analyzed by looking for high volumes and using machine learning cluster analysis of various locations. You have plenty of work to do to clean up these sites. As a final approach in this chapter, you will see how to use transaction analysis techniques and the apriori algorithm to analyze your messages per host to see if you can find anything else. There is significant encoding here to make the process easier to implement and more scalable. This encoding may get confusing at times, so follow closely. Remember that you are building atomic components that you will use over and over again with new data, so taking the time to build these is worth it.

Using market basket intuition, you want to turn every generalized syslog message into an item for that device, just as if it were an item in a shopping basket. Then you can analyze the per-device profiles just like retailers examine per-shopper profiles. Using the same dataframe you used in the previous section, you can add two new columns to help with this, as shown in Figure 12-44.

```
1 fix1={' ': '_'}
2 df['cleaned_message3']=df.cleaned_message2.replace(fix1,regex=True)
3 df['cleaned_message4']=df.cleaned_message3.apply(lambda x: [str(x)])
```

Figure 12-44 *Preparing Message Data for Analysis*

You have learned in this chapter how to replace items in the data by using a Python dictionary. In this case, you replace all spaces in the cleaned messages with underscores so that the entire message looks like a single term, and you create a new column for this. As shown in line 3 in Figure 12-45, you create a list representation of that string to use for encoding into the array used for the Gensim dictionary creation.

```
1 df['logbaskets']=df[['host','message_code']]\
2 .groupby(['host'])['message_code'].transform(lambda x: ' '.join(x))
3 df=df.drop_duplicates('host')
4 len(df.host.unique())
```

14007

Figure 12-45 *Count of Unique Cleaned Messages Encoded in the Dictionary*

Recall that this dictionary creates entries that are indexed with the *(number: item)* format. You can use this as an encoder for the analysis you want to do. Each individual cleaned message type gets its own number. When you apply this to your cleaned message array, notice that you have only 133 types of cleaned messages from your data of 1.5 million records. You will find that you also have a finite number of message types for each area that you chose to analyze.

Using your newly created dictionary, you can now create encodings for each of your message types by defining a function, as shown in Figure 12-46.

```
1 def get_key(x):
2     y=str(log_dictionary.keys()[log_dictionary.values().index(x)])
3     return y
4 df['message_code']=df['cleaned_message3'].apply(get_key)
```

Figure 12-46 Python Function to Look Up Message in the Dictionary

This function looks up the message string in the dictionary and returns the dictionary key. The dictionary key is a number, as you learned in Chapter 11, but you need a string because you want to combine all the keys per device into a single string representation of a *basket* of messages per device. You should now be very familiar with using the **groupby** method to gather messages per device, and it is used again in Figure 12-47.

```
1 df['logbaskets']=df[['host','message_code']]\
2 .groupby(['host'])['message_code'].transform(lambda x: ' '.join(x))
3 df=df.drop_duplicates('host')
4 len(df.host.unique())
```

14007

Figure 12-47 Generating Baskets of Messages per Host

In the last section, you grouped by your locations. In this section, you group by any host that sent you messages. You need to gather all the message codes into a single string in a new column called logbaskets. This column has a code for each log message produced by the host, as shown in Figure 12-48. You have more than 14,000 (See Figure 12-47) devices when you look for unique hosts in the dataframe host column.

```
1 df.iloc[1000].logbaskets
```

```
'3 3 9 9 3 10 3 9 3 26 9 3 9 3 26 26 26 28 26 26 28 26 26 26 9 29 3 29 29 30 30
30 29 29 29 30 30 30 29 29 29 30 31 30 29 29 30 30 29 30 29 30 29 30 29 30 29 3
2 29 30 10 10 3 3 3 29 30 29 30 29 30 29 30 29 30 29 30 29 30 3 29 3 30 2
9 31 3 29 31 3 9 3 26 26 26 26 26 9 3 26 9 3 26 9 3 9 3 9 3 3 3 3 3 9 10
3 3 3 3 3 3 3 10 10 18 20 29 32 29 31 10 10 3 9 9 3 3 9 9 3 9 3 26 26 26
9 3 9 3 9 3 9 3 9 3 3 9 3 3 26 26 26 9 3 28 9 3 3 9 3 3 9 3 10 3 9 3 3
9 3 9 3 9 3 9 3 26 9 3 9 3 26 9 3 26 9 3 9 3 9 3 26 26 9 3 26 9 3 26 26
9 3 9 3 9 3 9 3 9 3 9 3 9 3 9 3 9 3 26 26 28 9 3 26 9 3 9 3 9 3 9 3
9 3 9 3 9 3 9 3 26 9 3 9 3 9 3 9 3 9 3 3 10 3 10 3 3 10 3 10 10'
```

Figure 12-48 Encoded Message Basket for One Host

This large string represents every message received from the device during the entire week. Because you are using market basket intuition, this is the device “shopping basket.”

Figure 12-49 shows how you can see what each number represents by viewing the dictionary for that entry.

```
1 log_dictionary[3]
u'process_proc_nbr_host_on_int_from_loading_to_full_loading_done_'
```

Figure 12-49 *Lookup for Dictionary-Encoded Message*

Because you are only looking for unique combinations of messages, the order and repeating of messages are not of interest. The analysis would be different if you were looking for sequential patterns in the log messages. You are only looking at unique items per host, so you can tokenize, remove duplicates, and create a unique log string per device, as shown in Figure 12-50. You could also choose to keep all tokens and use term frequency/inverse document frequency (TF/IDF) encoding here and leave the duplicates in the data. In this case, you will deduplicate to work with a unique signature for each device by using the python set in line four.

```
1 import nltk
2 df['tokens'] = df.apply(lambda row: nltk.word_tokenize\
3                        (row['logbaskets']), axis=1)
4 df['unique_tokens']=df['tokens'].apply(lambda x: list(set(x)))
5 df['token_string']=df['unique_tokens'].apply(lambda x: ' '.join(x))
```

Figure 12-50 *Creating a Unique Signature of Encoded Dictionary Representation*

Now you have a token string that represents the unique set of messages that each device generated. We will not go down the search and similarity path again in this chapter, but it is now possible to find other devices that have the same log signature by using the techniques from Chapter 11.

For this analysis, you can create the transaction encoding by using the unique string to create a tokenized basket for each device, as shown in Figure 12-51.

```
1 import nltk
2 df['hostbasket'] = df.apply(lambda row: nltk.word_tokenize\
3                            (row['token_string']), axis=1)
```

Figure 12-51 *Tokenizing the Unique Encoded Log Signature*

With this unique tokenized representation of your baskets, you can use a package that has the apriori function you want, as shown in Figure 12-52. You have now experienced the excessive time it takes to prepare data for analysis, and you are finally ready to do some analysis.

```

1 from mlxtend.preprocessing import TransactionEncoder
2 from mlxtend.frequent_patterns import apriori
3 trans_enc = TransactionEncoder()
4 te_encoded = trans_enc.fit(df['hostbasket']).transform(df['hostbasket'])
5 tedf = pd.DataFrame(te_encoded, columns=trans_enc.columns_)
6 tedf.columns

Index([u'0', u'1', u'10', u'100', u'101', u'102', u'103', u'104', u'105',
      u'106',
      ...,
      u'90', u'91', u'92', u'93', u'94', u'95', u'96', u'97', u'98', u'99'],
      dtype='object', length=133)

```

Figure 12-52 Encoding Market Basket Transactions with the Apriori Algorithm

After loading the packages, you can create an instance of the transaction encoder and fit this to the data. You can create a new dataframe called `tedf` with this information. If you examine the output, you should recognize the length of the columns as the number of unique items in your log dictionary. This is very similar to the encoding that you already did in Chapter 11. There is a column for each value, and each row has a device with an indicator of whether the device in that row has the message in its host basket.

Now that you have all the messages encoded, you can generate frequent item sets by applying the apriori algorithm to the encoded dataframe that you created and return only messages that have a minimum support level, as shown in Figure 12-53. Details for how the apriori algorithm does this are available in Chapter 8, “Analytics Algorithms and the Intuition Behind Them.”

```

1 frequent_itemsets = apriori(tedf, min_support=0.3, use_colnames=True)
2 frequent_itemsets['length'] = \
3     frequent_itemsets['itemsets'].apply(lambda x: len(x))

```

Figure 12-53 Identifying Frequent Groups of Log Messages

When you look at all of your data, you see that you do not have many common sets of messages across all hosts. Figure 12-54 shows that only five individual messages or sets of messages show up together more than 30% of the time.

1 frequent_itemsets			
	support	itemsets	length
0	0.368816	[10]	1
1	0.526665	[11]	1
2	0.960734	[3]	1
3	0.365817	[10, 3]	2
4	0.525880	[11, 3]	2

Figure 12-54 Frequent Log Message Groups

Recall the message about a neighbor relationship being established. This message appears at least once on 96% of your devices. So how do you use this for analysis? Recall that you built this code with the entire data set. Many things are going to be generalized if you look across the entire data set. Now that you have set up the code to do market basket analysis, you can go back to the beginning of your analysis (just before Figure 12-19) and add a filter for each site that you want to analyze, as shown in Figure 12-55. Then you can run the filtered data set through the market basket code that you have built in this chapter.

```

1 #df=df[df.city=='Plainville']
2 #df=df[df.city=='Gibson']
3 #df=df[df.city=='Butler']
4 df=df[df.city=='Santa Fe']
5 #df=df[df.city=='Lookout Mountain']
6 #df=df[df.city=='Lincolnton']
7 #df=df[df.city=='Augusta']
8 #df=df[df.city=='Raleigh']
9 len(df)

```

881880

Figure 12-55 *Filtering the Entire Analysis by Location*

In this case, you did not remove the noise, and you filtered down to the Santa Fe location, as shown in Figure 12-56. Based on what you have learned, you should already know what you are going to see as the most common baskets at Santa Fe.

1 frequent_itemsets			
	support	itemsets	length
0	0.796875	[0]	1
1	0.796875	[1]	1
2	0.968750	[3]	1
3	0.812500	[6]	1
4	0.828125	[7]	1
5	0.796875	[0, 1]	2
6	0.812500	[3, 6]	2
7	0.828125	[3, 7]	2

Figure 12-56 *Frequent Message Groups for Santa Fe*

Figure 12-57 shows how to look up the items in the transactions in the log dictionary. On the first two lines, notice the **key** messages that you expected. It is interesting that they are only on about 80% of the logs, so not all devices are exposed to this key issue, but the ones that are exposed are dominating the logs from the site. You can find the bracketed item sets within the log dictionary to examine the transactions.

```

1 print("1: " + log_dictionary[0])
2 print("2: " + log_dictionary[1])
3 print("3: " + log_dictionary[3])
4 print("4: " + log_dictionary[6])
5 print("5: " + log_dictionary[7])

```

1: no_valid_authentication_send_key_is_available_on_interface_int_
2: key_id_0_received_on_interface_int_
3: process_proc_nbr_host_on_int_from_loading_to_full_loading_done_
4: process_proc_nbr_host_on_int_from_full_to_down_neighbor_down:_interface_down_or_detached_
5: process_proc_nbr_host_on_int_from_full_to_down_neighbor_down:_dead_timer_expired_

Figure 12-57 *Lookup Method for Encoded Messages*

One thing to note about Santa Fe and this type of analysis in general is the inherent noise reduction you get by using only unique transactions. In the other analyses to this point, the **key** messages have dominated the counts, or you have removed them to focus on other messages. Now you still represent these messages but do not overwhelm the analysis because you do not include the counts. This is a third perspective on the same data that allows you to uncover new insights.

If you look at your scatterplot again to find out what is unique about something that appeared to be on a cluster edge, you can find additional items of interest by using this method. Look at the closest point to the single node cluster in Figure 12-58, which is your Raleigh (RTP) location.

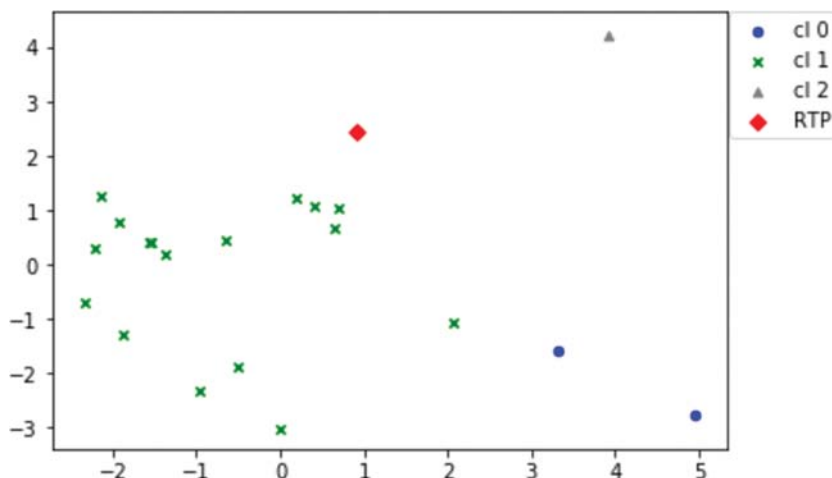


Figure 12-58 *Scatterplot of Relative Log Signature Differences with Clustering*

When you examine the data from Raleigh, you see some new frequent messages in Figure 12-59 that you didn't see before.

1	frequent_itemsets		
	support	itemsets	length
0	0.681416	[1]	1
1	0.632743	[2]	1
2	0.623894	[1, 2]	2
1	print("1: " + log_dictionary[1])		
2	print("2: " + log_dictionary[2])		
1:	interface_int_cost-fallback_not_active_metric_n__		
2:	interface_int_cost-fallback_active_metric_n__		

Figure 12-59 *Frequent Groups of Messages in Raleigh*

If you put on your SME hat, you can determine that this relates to link bundles adjusting their OSPF cost because link members are being added and dropped. These messages showing up here in a frequent transaction indicate that this pair is repeating across 60% of the devices. This tells you that there is churn in the routing metrics. Add it to the task list.

Finally, note that common sets of messages can be much longer than just two messages. The set in Figure 12-60 shows up in Plainville on 50% of the devices. This means that more than half of the routers in Plainville had new connections negotiated. Was this expected?

1	best=frequent_itemsets['length'].max()
2	print(df.iloc[0].city)
3	for entry in list(frequent_itemsets.itemsets):
4	if len(entry)>=best:
5	print("transaction length " + str(len(entry)))
6	for item in entry:
7	print(item,log_dictionary[int(item)])
Plainville	
transaction length 6	
('3', u'process_proc_nbr_host_on_int_from_loading_to_full_loading_done_')	
('4', u'process_proc_nbr_host_on_int_from_exchange_to_loading_exchange_done_')	
('5', u'process_proc_nbr_host_on_int_from_exstart_to_exchange_negotiation_done_')	
('6', u'process_proc_nbr_host_on_int_from_init_to_2way_2-way_received_')	
('7', u'process_proc_nbr_host_on_int_from_down_to_init_received_hello_')	
('8', u'process_proc_nbr_host_on_int_from_2way_to_exstart_adjok?_')	

Figure 12-60 *Plainville Longest Frequent Message Set*

You could choose to extend this method to count occurrences of the sets, or you could add ordered transaction awareness with variable time windows. Those advanced methods are natural next steps, and that is what Cisco Services does in the Network Early Warning (NEW) tool.

You now have 16 items to work on, and you can stop finding more. In this chapter you have learned new ways to use data visualization and data science techniques to do log analysis. You can now explore ways to build these into regular analysis engines that become part of your overall workflow and telemetry analysis. Remember that the goal is to build atomic components that you can add to your overall solution set.

You can very easily add a few additional methods here. In Chapter 11, you learned how to create a search index for device profiles based on hardware, software, and configuration. Now you know how to create syslog profiles. You have been learning how to take the intuition from one solution and use it to build another. Do you want another analysis method? If you can cluster device profiles, you can cluster log profiles. You can cluster anything after you encode it. Devices with common problems cluster together. If you find a device with a known problem during your normal troubleshooting, you could use the search index or clustering to find other devices most like it. They may also be experiencing the same problem.

Task List

Table 12-2 shows the task list that you have built throughout the chapter, using a combination of SME expert analysis and machine learning.

Table 12-2 Work Items Found in This Chapter

#	Category	Task
1	Security issue	Fix authentication keys at Santa Fe
2	Security issue	Fix authentication keys at Fort Lauderdale
3	Security issue	Fix authentication keys at Lincolnnton
4	Security issue	Fix authentication keys at Plentywood
5	Security issue	Fix authentication keys at New York
6	Security issue	Fix authentication keys at Sandown
7	Security issue	Fix authentication keys at Trenton
8	Security issue	Fix authentication keys at Lookout Mountain
9	Data loss	Investigate why no messages from Butler for a period on the 11th
10	Routing issue	Investigate adjacency changes at Lookout Mountain

#	Category	Task
11	Routing issue	Investigate OSPF message spikes at Lookout Mountain
12	Routing issue	Investigate OSPF problems at Butler
13	OSPF logs	Investigate OSPF duplicate problems at Plainville
14	OSPF logs	Investigate OSPF flooding in Gibson
15	OSPF logs	Investigate cost fallback in Raleigh
16	OSPF logs	Investigate neighbor relationships established in Plainville

Summary

In this chapter, you have learned many new ways to analyze log data. First, you learned how to slice, dice, and group data programmatically to mirror what common log packages provide. When you do this, you can include the same type of general evaluation of counts and message types in your workflows. Combined with what you have learned in Chapters 10 and 11, you now have some very powerful capabilities.

You have also seen how to perform data visualization on telemetry data by developing and using encoding methods to use with any type of data. You have seen how to represent the data in ways that open up many machine learning possibilities. Finally, you have seen how to use common analytics techniques such as market basket analysis to examine your own data in full or in batches (by location or by host, for example).

You could go deeper with any of the techniques you have learned in this chapter to find more tasks and apply your new techniques in many different ways. So far in this book, you have learned about management plane data analysis and analysis of a control plane protocol using telemetry reporting. In Chapter 13, “Developing Real Use Cases: Data Plane Analytics,” the final use-case chapter, you will perform analysis on data plane traffic captures.

This page intentionally left blank

Developing Real Use Cases: Data Plane Analytics

This chapter provides an introduction to data plane analysis using a data set of over 8 million packets loaded from a standard pcap file format. A publicly available data set is used to build the use case in this chapter. Much of the analysis here focuses on ports and addresses, which is very similar to the type of analysis you do with NetFlow data. It is straightforward to create a similar data set from native NetFlow data. The data inside the packet payloads is not examined in this chapter. A few common scenarios are covered:

- Discovering what you have on the network and learning what it is doing
- Combining your SME knowledge about network traffic with some machine learning and data visualization techniques
- Performing some cybersecurity investigation
- Using unsupervised learning to cluster affinity groups and bad actors

Security analysis of data plane traffic is very mature in the industry. Some rudimentary security checking is provided in this chapter, but these are rough cuts only. True data plane security occurs inline with traffic flows and is real time, correlating traffic with other contexts. These contexts could be time of day, day of week, and/or derived and defined standard behaviors of users and applications. The context is unavailable for this data set, so in this chapter we just explore how to look for interesting things in interesting ways. As when performing a log analysis without context, in this chapter you will simply create a short list of findings. This is a standard method you can use to prioritize findings after combining with context later. Then you can add useful methods that you develop to your network policies as expert systems rules or machine learning models. Let's get started.

The Data

The data for this chapter is traffic captured during collegiate cyber defense competitions, and there are some interesting patterns in it for you to explore. Due to the nature of this competition, this data set has many interesting scenarios for you to find. Not all of them are identified, but you will learn about some methods for finding the unknown unknowns.

The analytics infrastructure data pipeline is rather simple in this case, no capture mechanism was needed. The public packet data was downloaded from <http://www.netresec.com/?page=MACCDC>. The files are from standard packet capture methods that produce pcap-formatted files. You can get pcap file exports from most packet capture tools, including Wireshark (refer to Chapter 4, “Accessing Data from Network Components”). Alternatively, you can capture packets from your own environment by using Python scapy, which is the library used for analysis in this chapter. In this section, you will explore the downloaded data by using the Python packages scapy and pandas. You import these packages as shown in Figure 13-1.

```
1 from scapy.all import *
2 import datetime
3 import pandas as pd
```

Figure 13-1 *Importing Python Packages*

Loading the pcap files is generally easy, but it can take some time. For example, the import of the 8.5 million packets shown in Figure 13-2 took two hours to load the 2G file that contained the packet data. You are loading captured historical packet data here for data exploration and model *building*. *Deployment* of anything you build into a working solution would require that you can capture and analyze traffic near real time.

```
1 file0='/scratch/data/booktest/maccdc2012_00000.pcap'
2 packets = rdpcap(file0)
```

Figure 13-2 *Packet File Loading*

Only one of the many available MACCDC files was loaded this way, but 8.5 million packets will give you a good sample size to explore data plane activity.

Here we look again at some of the diagrams from Chapter 4 that can help you match up the details in the raw packets. The Ethernet frame format that you will see in the data here will match what you saw in Chapter 4 but will have an additional virtual local area network (VLAN) field, as shown in Figure 13-3.

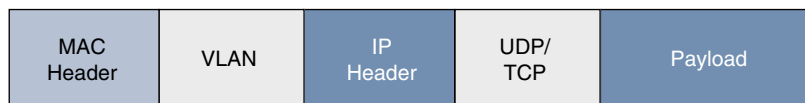


Figure 13-3 *IP Packet Format*

Compare the Ethernet frame in Figure 13-3 to the raw packet data in Figure 13-4 and notice the fields in the raw data. Note the end of the first row in the output in Figure 13-4, where you can see the Dot1Q VLAN header inserted between the MAC (Ether) and IP headers in this packet. Can you tell whether this is a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) packet?

```
1 packets[0]
<Ether dst=00:0c:29:41:4b:e7 src=00:16:47:9d:f2:c2 type=n_802_1Q |<Dot1Q prio
=0 id=0 vlan=120 type=IPv4 |<IP version=4 ihl=5 tos=0x0 len=99 id=36140 flags=
frag=0 ttl=254 proto=tcp checksum=0xfdc8 src=192.168.229.254 dst=192.168.202.79 o
ptions=[] |<TCP sport=https dport=46117 seq=2162570451 ack=4204467708 dataofs=
8 reserved=0 flags=PA window=32768 checksum=0x19da urgptr=0 options=[('NOP', Non
e), ('NOP', None), ('Timestamp', (319644338, 0))] |<Raw load='x14\x03\x01\x00
\x01\x01\x16\x03\x01\x00$\x95wk\xd4\xf3?\xae\xa1\xaa\xca\xf1\xfb\xe6\x02l&/\xc
c/\x8c\xd0\xf8(!m\xc4\xab\xa5\xbc\xc1\xa8\xe0;In\x82' |>>>>
```

Figure 13-4 Raw Packet Format from a pcap File

If you compare the raw data to the diagrams that follow, you can clearly match up the IP section to the IP packet in Figure 13-5 and the TCP data to the TCP packet format shown in Figure 13-6.

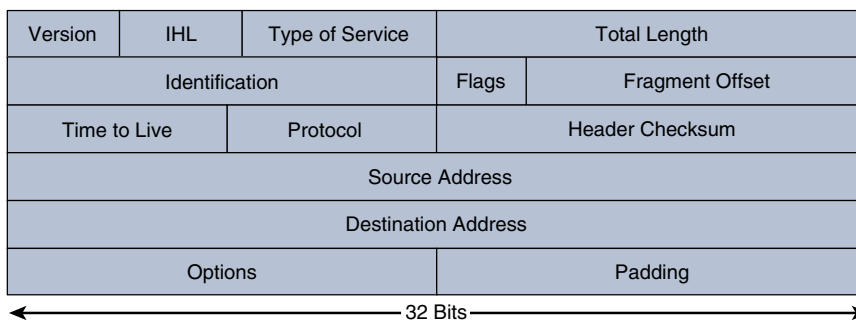


Figure 13-5 IP Packet Fields

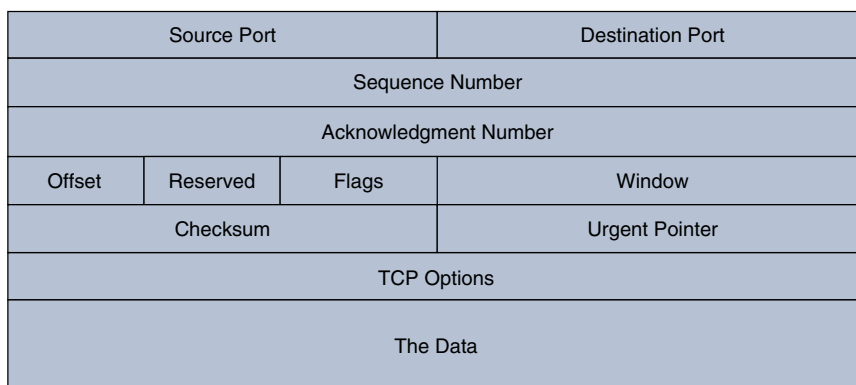


Figure 13-6 TCP Packet Fields

You could loop through this packet data and create Python data structures to work with, but the preferred method of exploration and model building is to structure your data so that you can work with it at scale. The dataframe construct is used again.

You can use a Python function to parse the interesting fields of the packet data into a dataframe. That full function is shared in Appendix A, “Function for Parsing Packets from pcap Files.” You can see the definitions for parsing in Table 13-1. If a packet does not have the data, then the field is blank. For example, a TCP packet does not have any UDP information because TCP and UDP are mutually exclusive. You can use the empty fields for filtering the data during your analysis.

Table 13-1 *Fields Parsed from Packet Capture into a Dataframe*

Packet Data Field	Parsed into the Dataframe as
None	id (unique ID was generated)
None	len (packet length was generated)
Ethernet source MAC address	esrc
Ethernet destination MAC address	edst
Ethernet type	etype
Dot1Q VLAN	vlan
IP source address	lsrc
IP destination address	ldst
IP length	iplen
IP protocol	ipproto
IP TTL	lpttl
UDP destination port	utdport
UDP source port	utsport
UDP length	ulen
TCP source port	tsport
TCP destination port	tdport
TCP window	twindow
ARP hardware source	arpsrc
ARP hardware destination	arpdst
ARP operation	arpop
ARP IP source	arppsrc
ARP IP destination	arppdst
NTP mode	ntpmode

Packet Data Field	Parsed into the Dataframe as
SNMP community	snmpcommunity
SNMP version	snmpversion
IP error destination	iperrordst
IP error source	iperrorproto
IP error protocol	iperrordst
UDP error destination	uerrordst
UDP error source	uerrorsrc
ICMP type	icmptype
ICMP code	icmpcode
DNS operation	dnsopcode
BootP operation	bootpop
BootP client hardware	bootpchaddr
BootP client IP address	bootpciaddr
BootP server IP address	bootpsiaddr
BootP client gateway	bootpgiaddr
BootP client assigned address	bootpyiaddr

This may seem like a lot of fields, but with 8.5 million packets over a single hour of user activity (see Figure 13-9), there is a lot going on. Not all the fields are used in the analysis in this chapter, but it is good to have them in your dataframe in case you want to drill down into something specific while you are doing your analysis. You can build some Python techniques that you can use to analyze files offline, or you can script them into systems that analyze file captures for you as part of automated systems.

Packets on networks typically follow some standard port assignments, as described at <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>. While these are standardized and commonly used, understand that it is possible to spoof ports and use them for purposes outside the standard. Standards exist so that entities can successfully interoperate. However, you can build your own applications using any ports, and you can define your own packets with any structure by using the scapy library that you used to parse the packets. For the purpose of this evaluation, assume that most packet ports are correct. If you do the analysis right, you will also pick up patterns of behavior that indicate use of nonstandard or unknown ports. Finally, having a port open does not necessarily mean the device is

running the standard service at that port. Determining the proper port and protocol usage is beyond the scope of this chapter but is something you should seek to learn if you are doing packet-level analysis on a regular basis.

SME Analysis

Let's start with some common SME analysis techniques for data plane traffic. To prepare for that, Figure 13-7 shows how to load some libraries that you will use for your SME exploration and data visualization.

```
1 import pandas as pd
2 import matplotlib as plt
3 from pandas import TimeGrouper
4 from wordcloud import WordCloud
5 import matplotlib.pyplot as pyplot
```

Figure 13-7 *Dataframe and Visualization Library Loading*

Here again you see **TimeGrouper**. You need this because you will want to see the packet flows over time, just as you saw telemetry over time in Chapter 12, “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry.” The packets have a time component, which you call as the index of the dataframe as you load it (see Figure 13-8), just as you did with syslog in Chapter 12.

```
1 filename="/scratch/data/booktest/packetdf_0000.csv"
2 df=pd.read_csv(filename, parse_dates=["timestamp"], \
3               dtype=object, index_col=["timestamp"])
4 print(df.columns)
5 print(df.shape)

Index([u'arpdst', u'arppop', u'arppdst', u'arppsrc', u'arpsrc', u'bootpchaddr',
      u'bootpciaddr', u'bootpgiaddr', u'bootpop', u'bootpsiaddr',
      u'bootpyiaddr', u'dhcpoptions', u'dnsopcode', u'edst', u'esrc',
      u'etype', u'icmpcode', u'icmptype', u'id', u'idst', u'iperrordst',
      u'iperrorproto', u'iperrorsrc', u'iplen', u'ipproto', u'ipttl', u'isrc',
      u'len', u'ntpmode', u'snmpcommunity', u'snmpversion', u'tdport',
      u'tsport', u'twindow', u'uerrordst', u'uerrorsrc', u'ulen', u'utdport',
      u'utsport', u'vlan'],
      dtype='object')
(8635943, 40)
```

Figure 13-8 *Loading a Packet Dataframe and Applying a Time Index*

In the output in Figure 13-8, notice that you have all the expected columns, as well as more than 8.5 million packets. Figure 13-9 shows how to check the dataframe index times to see the time period for this capture.

```

1 print(df.index.max())
2 print(df.index.min())

```

2012-03-16 13:30:07.780000
2012-03-16 12:30:00

Figure 13-9 *Minimum and Maximum Timestamps in the Data*

You came up with millions of packets in a single hour of capture. You will not be able to examine any long-term behaviors, but you can try to see what was happening during this very busy hour. The first thing you want to do is to get a look at the overall traffic pattern during this time window. You do that with **TimeGrouper**, as shown in Figure 13-10.

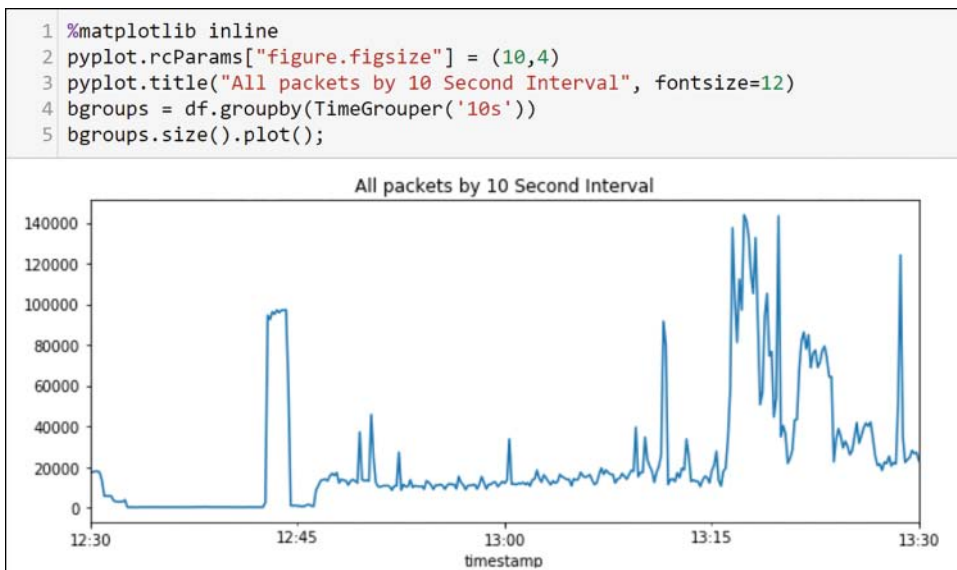


Figure 13-10 *Time Series Counts of Syslog Messages*

In this case, you are using the **pyplot** functionality to plot the time series. In line 4, you create the groups of packets, using 10-second intervals. In line 5, you get the size of each of those 10-second intervals and plot the sizes.

Now that you know the overall traffic profile, you can start digging into what is on the network. The first thing you want to know is how many hosts are sending and receiving traffic. This traffic is all IP version 4, so you only have to worry about the **isrc** and **idst** fields that you extracted from the packets, as shown in Figure 13-11.


```

1 print(df.isrc.value_counts().count())
2 print(df.idst.value_counts().count())

```

191
2709

Figure 13-11 *Counts of Source and Destination IP Addresses in the Packet Data*

If you use the `value_counts` function that you are very familiar with, you can see that 191 senders are sending to more than 2700 destinations. Figure 13-12 shows how to use `value_counts` again to see the top packet senders on the network.

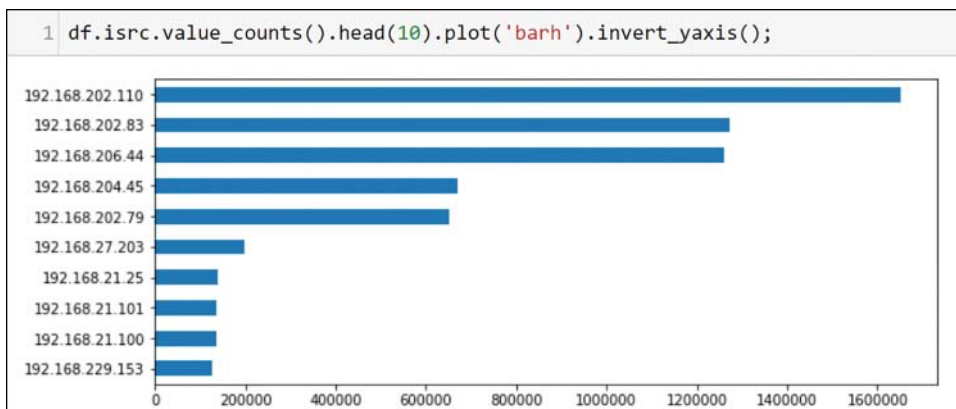


Figure 13-12 *Source IP Address Packet Counts*

Note that the source IP address value counts are limited to 10 here to make the chart readable. You are still exploring the top 10, and the `head` command is very useful for finding only the top entries. Figure 13-13 shows how to list the top packet destinations.

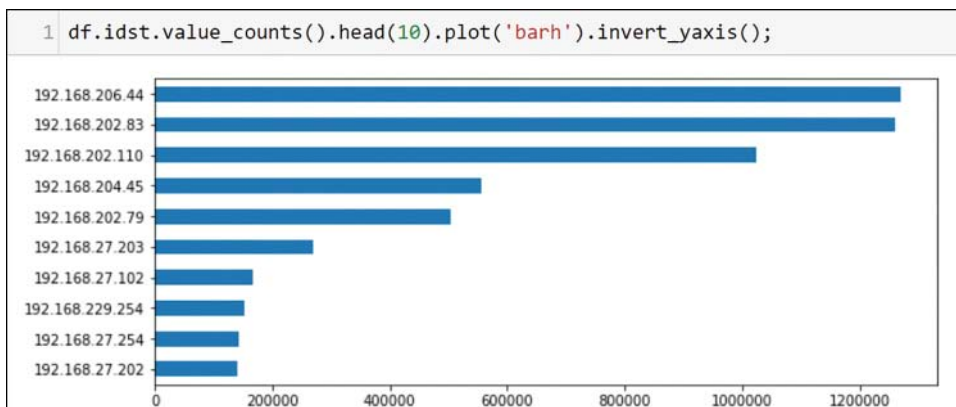


Figure 13-13 *Destinations IP Address Packet Counts*

In this case, you used the destination IP address to plot the top 10 destinations. You can already see a few interesting patterns. The hosts 192.168.202.83 and 192.168.202.110 appear at the top of each list. This is nothing to write home about (or write to your task list), but you will eventually want to understand the purpose of the high volumes for these two hosts. Before going there, however, you should examine a bit more about your environment. In Figure 13-14, look at the VLANs that appeared across the packets.

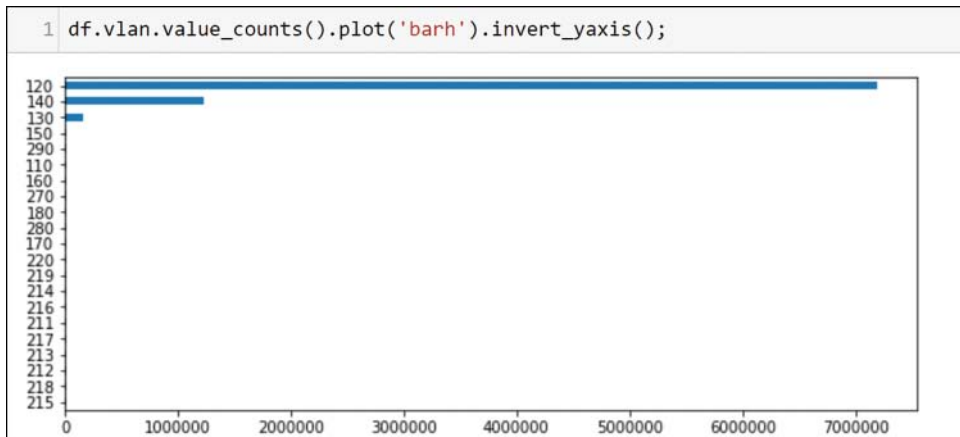


Figure 13-14 *Packet Counts per VLAN*

You can clearly see that the bulk of the traffic is from VLAN 120, and some also comes from VLANs 140 and 130. If a VLAN is in this chart, then it had traffic. If you check the IP protocols as shown in Figure 13-15, you can see the types of traffic on the network.

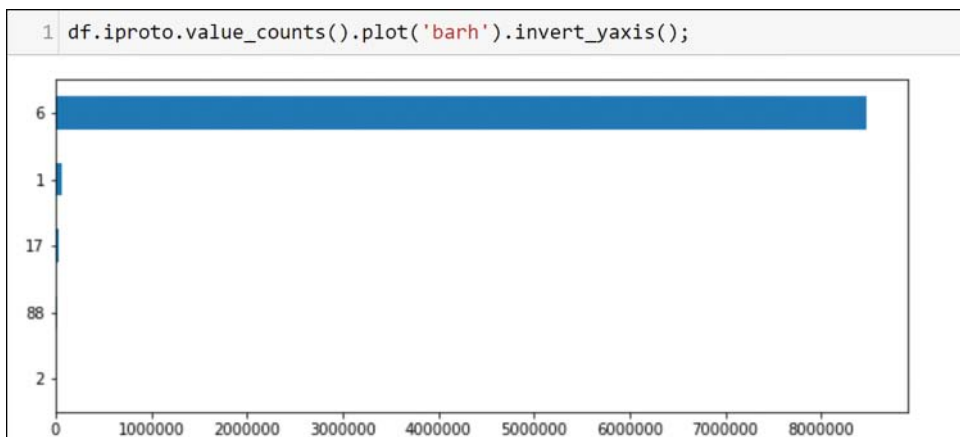


Figure 13-15 *IP Packet Protocols*

The bulk of the traffic is protocol 6, which is TCP. You have some Internet Control Message Protocol (ICMP) (ping and family), some UDP (17), and some Internet Group Management Protocol (IGMP). You may have some multicast on this network. The protocol 88 represents your first discovery. This protocol is the standard protocol for the Cisco Enhanced Interior Gateway Routing Protocol (EIGRP) routing protocol. EIGRP is a Cisco alternative to the standard Open Shortest Path First (OSPF) that you saw in Chapter 12. You can run a quick check for the well-known neighboring protocol address of EIGRP; notice in Figure 13-16 that there are at least 21 router interfaces active with EIGRP.

1	<code>df[df.idst=="224.0.0.10"].isrc.value_counts().count()</code>
21	

Figure 13-16 Possible EIGRP Router Counts

Twenty-one routers seems like a very large number of routers to be able to capture packets from in a single session. You need to dig a little deeper to understand more about the topology. You can see what is happening by checking the source Media Access Control (MAC) addresses with the same filter. Figure 13-17 shows that these devices are probably from the same physical device because all 21 sender MAC addresses (`esrc`) are nearly sequential and are very similar. (The figure shows only 3 of 21 devices for brevity.)

1	<code>df[df.idst=="224.0.0.10"].esrc.value_counts()</code>
00:16:47:9d:f2:d4	783
00:16:47:9d:f2:d3	783
00:16:47:9d:f2:d5	782

Figure 13-17 EIGRP Router MAC Addresses

Now that you know this is probably a single device using MAC addresses from an assigned pool, you can check for some topology mapping information by looking at all the things you checked together in a single group. You can use filters and the `groupby` command to bring this topology information together, as shown in Figure 13-18.

1	<code>routers=df[df.idst=="224.0.0.10"][['esrc','isrc','vlan','iproto']]</code>
2	<code>routers=routers.drop_duplicates(['isrc','esrc']).reset_index()</code>
3	<code>gr=routers.groupby(['isrc','vlan','esrc'])</code>
4	<code>gr.size()</code>

isrc	vlan	esrc	
192.168.201.1	110	00:16:47:9d:f2:c1	1
192.168.202.1	120	00:16:47:9d:f2:c2	1
192.168.203.1	130	00:16:47:9d:f2:c3	1
192.168.204.1	140	00:16:47:9d:f2:c4	1
192.168.205.1	150	00:16:47:9d:f2:c5	1
192.168.206.1	160	00:16:47:9d:f2:c6	1

Figure 13-18 Router Interface, MAC, and VLAN Mapping

This output shows that most of the traffic that you know to be on three VLANs is probably connected to a single device with multiple routed interfaces. MAC addresses are usually sequential in this case. You can add this to your table as a discovered asset. Then you can get off this router tangent and go back to the top senders and receivers to see what else is happening on the network.

Going back to the top talkers, Figure 13-19 uses host 192.168.201.110 to illustrate the time-consuming nature of exploring each host interaction, one at a time.

1	df[df.isrc=="192.168.202.110"].idst.value_counts().count()
2042	
1	df[df.isrc=="192.168.202.110"].iproto.value_counts()
6	1637017
1	10631
17	2531
Name: ipproto, dtype: int64	
1	df[df.isrc=="192.168.202.110"].tdport.value_counts().count()
65536	
1	df[df.isrc=="192.168.202.110"].tdport.value_counts().head(10)
80	170196
8080	104829

Figure 13-19 *Host Analysis Techniques*

Starting from the top, see that host 110 is talking to more than 2000 hosts, using mostly TCP, as shown in the second command, and it has touched 65,536 unique destination ports. The last two lines in Figure 13-19 show that the two largest packet counts to destination ports are probably web servers.

In the output of these commands, you can see the first potential issue. This host tried every possible TCP port. Consider that the TCP packet ports field is only 16 bits, and you know that you only get 64k (1k=1024) entries, or 65,536 ports. You have identified a host that is showing an unusual pattern of activity on the network. You should record this in your investigation task list so you can come back to it later.

With hundreds or thousands of hosts to examine, you need to find a better way. You have an understanding of the overall traffic profile and some idea of your network topology at this point. It looks as if you are using captured traffic from a single large switch environment with many VLAN interfaces. Examining host by host, parameter by parameter would be quite slow, but you can create some Python functions to help. Figure 13-20 shows the first function for this chapter.

```

1 sourceip="192.168.202.110"
2 def packet_info(sourceip):
3     a=df[df.isrc==sourceip].icmpcode.value_counts().sum()
4     print(sourceip + " ICMP Packets: " + str(a))
5     b=df[df.isrc==sourceip].arpshr.value_counts().sum()
6     print(sourceip + " ARP Packets: " + str(b))
7     c=df[df.isrc==sourceip].dnscode.value_counts().sum()
8     print(sourceip + " DNS Packets: " + str(c))
9     d=df[df.isrc==sourceip].bootp.value_counts().sum()
10    print(sourceip + " BOOTP Packets: " + str(d))
11    e=df[df.isrc==sourceip].tshost.value_counts().sum()
12    print(sourceip + " TCP Packets: " + str(e))
13    f=df[df.isrc==sourceip].tshost.value_counts().sum()
14    print(sourceip + " UDP Packets: " + str(f))
15    g=df[df.isrc==sourceip].snmpcommunity.value_counts().sum()
16    print(sourceip + " SNMP Packets: " + str(g))

```

Figure 13-20 *Smart Function to Automate per-Host Analysis*

With this function, you can send any source IP address as a variable, and you can use that to filter through the dataframe for the single IP host. Note the sum at the end of `value_counts`. You are not looking for individual `value_counts` but rather for a summary for the host. Just add `sum` to `value_counts` to do this. Figure 13-21 shows an example of the summary data you get.

```

1 sourceip="192.168.202.110"
2 packet_info(sourceip)

```

192.168.202.110 ICMP Packets: 10631
192.168.202.110 ARP Packets: 0
192.168.202.110 DNS Packets: 825
192.168.202.110 BOOTP Packets: 9
192.168.202.110 TCP Packets: 1637017
192.168.202.110 UDP Packets: 2531
192.168.202.110 SNMP Packets: 163

Figure 13-21 *Using the Smart Function for per-Host Analysis*

This host sent more than 1.6 million packets, most of them TCP, which matches what you saw previously. You add more information requests to this function, and you get it all back in a fraction of the time it takes to go run these commands individually. You also want to know the hosts at the other end of these communications, and you can create another function for that, as shown in Figure 13-22.

```

1 def i_talk_to(sourceip):
2     print("I talk to:")
3     print(df[df.isrc==sourceip].groupby(['isrc','idst'])\
4           .size().sort_values(ascending=False))

```

```

1 sourceip="192.168.202.110"
2 i_talk_to(sourceip)

```

I talk to:

isrc	idst	
192.168.202.110	192.168.27.203	266811
	192.168.27.102	163083
	192.168.27.254	140330

Figure 13-22 Function for per-Host Conversation Analysis

You already know that this sender is talking to more than 2000 hosts, and this output is truncated to the top 3. You can add a head to the function if you only want a top set in your outputs. Finally, you know that the TCP and UDP port counts already indicate scanning activity. You need to watch those as well. As shown in Figure 13-23, you can add them to another function.

```

1 def host_profile(df,sourceip):
2     dests=df[df.isrc==sourceip].idst.value_counts().count()
3     print(sourceip + " Destinations: " + str(dests))
4     tdests=df[df.isrc==sourceip].tdport.value_counts().count()
5     print(sourceip + " TCP Port Destinations: " + str(tdests))
6     tdests=df[df.isrc==sourceip].tsport.value_counts().count()
7     print(sourceip + " TCP Port Sources: " + str(tdests))
8     udests=df[df.isrc==sourceip].utdport.value_counts().count()
9     print(sourceip + " UDP Port Destinations: " + str(udests))
10    udests=df[df.isrc==sourceip].utsport.value_counts().count()
11    print(sourceip + " UDP Port Sources: " + str(udests))
12    packet_info(sourceip)
13    i_talk_to(sourceip)

```

Figure 13-23 Function for a Full Host Profile Analysis

Note that here you are using counts instead of sum. In this case, however, you want to see the count of possible values rather than the sum of the packets. You also want to add the other functions that you created at the bottom, so you can examine a single host in detail with a single command. As with your solution building, this involves creating atomic components that work in a standalone manner, as in Figure 13-21 and Figure 13-22, and become part of a larger system. Figure 13-24 shows the result of using your new function.


```
1 sourceip="192.168.202.110"
2 host_profile(df,sourceip)

192.168.202.110 Destinations: 2042
192.168.202.110 TCP Port Destinations: 65536
192.168.202.110 TCP Port Sources: 64515
192.168.202.110 UDP Port Destinations: 100
192.168.202.110 UDP Port Sources: 1355
192.168.202.110 ICMP Packets: 10631
192.168.202.110 ARP Packets: 0
192.168.202.110 DNS Packets: 825
192.168.202.110 BOOTP Packets: 9
192.168.202.110 TCP Packets: 1637017
192.168.202.110 UDP Packets: 2531
192.168.202.110 SNMP Packets: 163
I talk to:
isrc          idst
192.168.202.110 192.168.27.203    266811
                  192.168.27.102  163083
                  192.168.27.254  140330
```

Figure 13-24 Using the Full Host Profile Function on a Suspect Host

With this one command, you get a detailed look at any individual host in your capture. Figure 13-25 shows how to look at another of the top hosts you discovered previously.

```
1 sourceip="192.168.202.83"
2 host_profile(df,sourceip)

192.168.202.83 Destinations: 4
192.168.202.83 TCP Port Destinations: 1000
192.168.202.83 TCP Port Sources: 28232
192.168.202.83 UDP Port Destinations: 1
192.168.202.83 UDP Port Sources: 1307
192.168.202.83 ICMP Packets: 4
192.168.202.83 ARP Packets: 0
192.168.202.83 DNS Packets: 1336
192.168.202.83 BOOTP Packets: 0
192.168.202.83 TCP Packets: 1269780
192.168.202.83 UDP Packets: 1336
192.168.202.83 SNMP Packets: 0
I talk to:
isrc          idst
192.168.202.83 192.168.206.44    1269574
                  192.168.207.4    1336
                  192.168.201.2    198
                  110.209.6.25    12
```

Figure 13-25 Using the Full Host Profile Function on a Second Suspect Host

In this output, notice that this host is only talking to four other hosts and is not using all TCP ports. This host is primarily talking to one other host, so maybe this is normal. The very even number of 1000 ports seems odd for talking to only 4 hosts, and you need to make a way to check it out. Figure 13-26 shows how you create a new function to step through and print out the detailed profile of the port usage that the host is exhibiting in the packet data.

```

1 def port_info(df, sourceip):
2     print(sourceip + " Source Ports TCP -----")
3     print(df[df.isrc==sourceip].tsport.value_counts())
4     print(sourceip + " Source Ports UDP -----")
5     print(df[df.isrc==sourceip].utsport.value_counts())
6     print(sourceip + " Destination Ports TCP -----")
7     print(df[df.isrc==sourceip].tdport.value_counts())
8     print(sourceip + " Destination Ports UDP -----")
9     print(df[df.isrc==sourceip].utdport.value_counts())

```

Figure 13-26 Smart Function for per-Host Detailed Port Analysis

Here you are not using `sum` or `count`. Instead, you are providing the full `value_counts`. For the 192.168.201.110 host that was examined previously, this would provide 65,000 rows. Jupyter Notebook shortens it somewhat, but you still have to review long outputs. You should therefore keep this separate from the `host_profile` function and call it only when needed. Figure 13-27 shows how to do that for host 192.168.202.83 because you know it is only talking to 4 other hosts.

```

1 port_info(df, "192.168.202.83")

```

192.168.202.83 Destination Ports TCP -----	
80	7752
53	3771
111	3771
443	2514
5633	1257
10025	1257
3269	1257
3268	1257
6881	1257

Figure 13-27 Using the per-Host Detailed Port Analysis Function

This output is large, with 1000 TCP ports, so Figure 13-27 shows only some of the TCP destination port section here. It is clear that 192.168.202.83 is sending a large number of packets to the same host, and it is sending an equal number of packets to many ports on that host. It appears that 192.168.202.83 may be scanning or attacking host 192.168.206.44 (see Figure 13-25). You should add this to your list for investigation. Figure 13-28 shows a final check, looking at host 192.168.206.44.


```

1 host_profile(df,"192.168.206.44")

192.168.206.44 Destinations: 1
192.168.206.44 TCP Port Destinations: 28232
192.168.206.44 TCP Port Sources: 1000
192.168.206.44 UDP Port Destinations: 0
192.168.206.44 UDP Port Sources: 0
192.168.206.44 ICMP Packets: 4
192.168.206.44 ARP Packets: 0
192.168.206.44 DNS Packets: 0
192.168.206.44 BOOTP Packets: 0
192.168.206.44 TCP Packets: 1259514
192.168.206.44 UDP Packets: 0
192.168.206.44 SNMP Packets: 0
I talk to:
isrc      idst
192.168.206.44 192.168.202.83 1259518
dtype: int64

```

Figure 13-28 *Host Profile for the Host Being Attacked*

This profile clearly shows that this host is talking only to a single other host, which is the one that you already saw. You should add this one to your list for further investigation. As a final check for your SME side of the analysis, you should use your knowledge of common ports and the code in Figure 13-29 to identify possible servers in the environment. Start by making a list of ports you know to be interesting for your environment.

```

1 interesting_ports=['20','21','22','23','25','53','67','68','69',\
2                  '80','123','137','161','427','443','514','554',\
3                  '666','1900','3128','3306','3702','4444','5357',\
4                  '5432','8000','8080','8089']
5 for port in interesting_ports:
6     print("Top 5 UDP active on port: " + port)
7     print(df[df.utsport==port].isrc.value_counts().head(5))
8     print("Top 5 TCP active on port: " + port)
9     print(df[df.tsport==port].isrc.value_counts().head(5))

```

Figure 13-29 *Loop for Identifying Top Senders on Interesting Ports*

This is a very common process for many network SMEs: applying what you know to a problem. You know common server ports on networks, and you can use those ports to discover possible services. In the following output from this loop, can you identify possible servers? Look up the port numbers, and you will find many possible services running on these hosts. Some possible assets have been added to Table 13-2 at the end of the chapter, based on this output. This output is a collection of the top 5 source addresses with packet counts sourced by the interesting ports list you defined in Figure 13-29. Using the `head` command will only show up to the top 5 for each. If there are fewer than 5 in the data then the results will show fewer than 5 entries in the output.

Top 5 TCP active on port: 20

192.168.206.44	1257

Top 5 TCP active on port: 21

192.168.206.44	1257

192.168.27.101	455

192.168.21.101	411

192.168.27.152	273

192.168.26.101	270

Top 5 TCP active on port: 22

192.168.21.254	2949

192.168.22.253	1953

192.168.22.254	1266

192.168.206.44	1257

192.168.24.254	1137

Top 5 TCP active on port: 23

192.168.206.44	1257

192.168.21.100	18

Top 5 TCP active on port: 25

192.168.206.44	1257

192.168.27.102	95

Top 5 UDP active on port: 53

192.168.207.4	6330

Top 5 TCP active on port: 53

192.168.206.44	1257

192.168.202.110	243

Top 5 UDP active on port: 123

192.168.208.18	122

192.168.202.81	58

Top 5 UDP active on port: 137

192.168.202.76	987

```

192.168.202.102    718
192.168.202.89    654
192.168.202.97    633
192.168.202.77    245

Top 5 TCP active on port: 161
192.168.206.44    1257

Top 5 TCP active on port: 3128
192.168.27.102    21983
192.168.206.44    1257

Top 5 TCP active on port: 3306
192.168.206.44    1257
192.168.21.203    343

Top 5 TCP active on port: 5432
192.168.203.45    28828
192.168.206.44    1257

Top 5 TCP active on port: 8089
192.168.27.253    1302
192.168.206.44    1257

```

This is the longest output in this chapter, and it is here to illustrate a point about the possible permutations and combinations of hosts and ports on networks. Your brain will pick up patterns that lead you to find problems by browsing data using these functions. Although this process is sometimes necessary, it is tedious and time-consuming. Sometimes there are no problems in the data. You could spend hours examining packets and find nothing. Data science people are well versed in spending hours, days, or weeks on a data set, only to find that it is just not interesting and provides no insights.

This book is about finding new and innovative ways to do things. Let's look at what you can do with what you have learned so far about unsupervised learning. Discovering the unknown unknowns is a primary purpose of this method. In the following section, you will apply some of the things you saw in earlier chapters to yet another type of data: packets. This is very much like finding a solution from another industry and applying it to a new use case.

SME Port Clustering

Combining your knowledge of networks with what you have learned so far in this book, you can find better ways to do discovery in the environment. You can combine your SME knowledge and data science and go further with port analysis to try to find more servers. Most common servers operate on lower port numbers, from a port range that goes up to 65,536. This means hosts that source traffic from lower port numbers are potential servers. As discussed previously, servers can use any port, but this assumption of low ports helps in initial discovery. Figure 13-30 shows how to pull out all the port data from the packets into a new dataframe.

```
1 dfports=df[['isrc','idst','tsport','tdport','utSPORT','utdport']]\
2 .fillna("0").copy()
3 dfports['tsport']=dfports['tsport'].astype(int)
4 dfports['tdport']=dfports['tdport'].astype(int)
5 dfports['utSPORT']=dfports['utSPORT'].astype(int)
6 dfports['utdport']=dfports['utdport'].astype(int)
7 dfports['d']=dfports['tdport']+dfports['utdport']
8 dfports['s']=dfports['tsport']+dfports['utSPORT']
9 dfports=dfports.drop(['tsport','tdport','utSPORT','utdport'],axis=1)
```

Figure 13-30 *Defining a Port Profile per Host*

In this code, you make a new dataframe with just sources and destinations for all ports. You can convert each port to a number from a string that resulted from the data loading. In lines 7 and 8 in Figure 13-30, you add the source and destinations together for TCP and UDP because one set will be zeros (they are mutually exclusive), and you convert empty data to zero with `fillna` when you create the dataframe. Then you drop all port columns and keep only the IP address and a single perspective of port sources and destinations, as shown in Figure 13-31.

1	dfports[:2]				
	isrc	idst	d	s	
timestamp					
2012-03-16 12:30:00	192.168.229.254	192.168.202.79	46117	443	
2012-03-16 12:30:00	192.168.202.79	192.168.229.254	443	46117	

Figure 13-31 *Port Profile per-Host Dataframe Format*

Now you have a very simple dataframe with packets, sources and destinations from both UDP and TCP. Figure 13-32 shows how you create a list of hosts that have fewer than 1000 TCP and UDP packets.

```

1 cutoff=1000
2 countframe=dfports.groupby('isrc').size().reset_index(name='counts')
3 droplist=list(countframe[countframe.counts<cutoff].isrc.unique())
4 len(droplist)

```

68

Figure 13-32 *Filtering Port Profile Dataframe by Count*

Because you are just looking to create some profiles by using your expertise and simple math, you do not want any small numbers to skew your results. You can see that 68 hosts did not send significant traffic in your time window. You can define any cutoff you want. You will use this list for filtering later. To prepare the data for that filtering, you add the average source and destination ports for each host, as shown in Figure 13-33.

```

1 d=dfports.groupby('isrc')['d'].mean().to_frame()
2 s=dfports.groupby('isrc')['s'].mean().to_frame()
3 sd=d.merge(s, how='outer', left_index=True, right_index=True).reset_index()
4 sd=sd[~sd.isrc.isin(droplist)]
5 sd[:2]

```

	isrc	d	s
9	192.168.202.1	2.166882	2.135016
10	192.168.202.10	1138.659275	8080.000000

Figure 13-33 *Generating and Filtering Average Source and Destination Port Numbers by Host*

After you add the average port per host to both source and destination, you merge them back into a single dataframe and drop the items in the drop list. Now you have a source and destination port average for each host that sent any significant amount of traffic. Recall that you can use K-means clustering to help with grouping. First, you set up the data for the elbow method of evaluating clusters, as shown in Figure 13-34.

```

1 import numpy as np
2 from scipy.spatial.distance import cdist
3 from sklearn.cluster import KMeans
4 std=np.array(sd[['s','d']])
5 tightness = []
6 possibleKs = range(1,10)
7 for k in possibleKs:
8     km = KMeans(n_clusters=k).fit(std)
9     km.fit(std)
10    tightness.append(sum(np.min(cdist(std, \
11    km.cluster_centers_, 'euclidean'), axis=1)) / std.shape[0])

```

Figure 13-34 *Evaluating K-means Cluster Numbers*

Note that you do not do any transformation or encoding here. This is just numerical data in two dimensions, but these dimensions are meaningful to SMEs. You can plot this data right now, but you may not have any interesting boundaries to help your understand it. You can use the K-means clustering algorithm to see if it helps with discovering more things about the data. Figure 13-35 shows how to check the elbow method for possible boundary options.

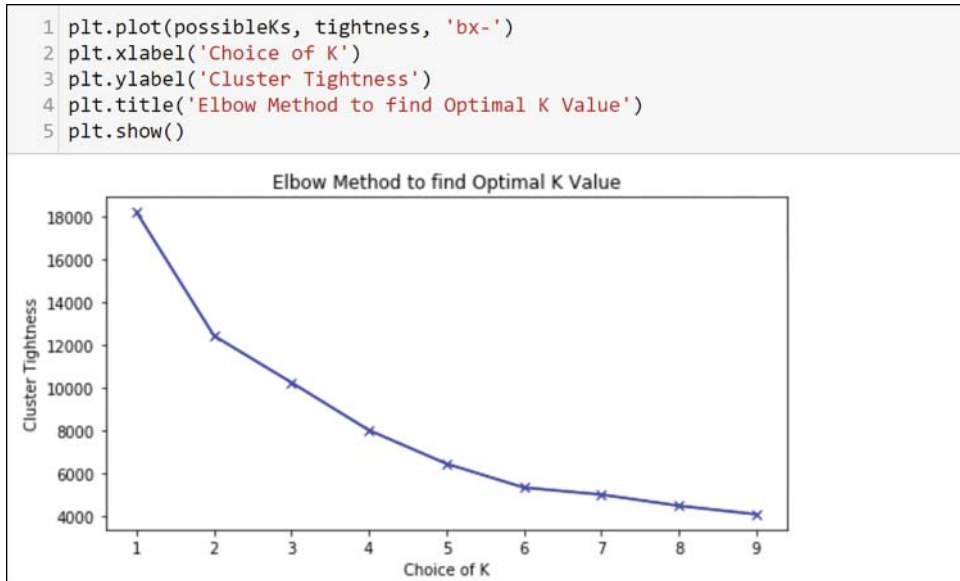


Figure 13-35 *Elbow Method for Choosing K-means Clusters*

The elbow method does not show any major cutoffs, but it does show possible elbows at 2 and 6. Because there are probably more than 2 profiles, you should choose 6 and run through the K-means algorithm to create the clusters, as shown in Figure 13-36.

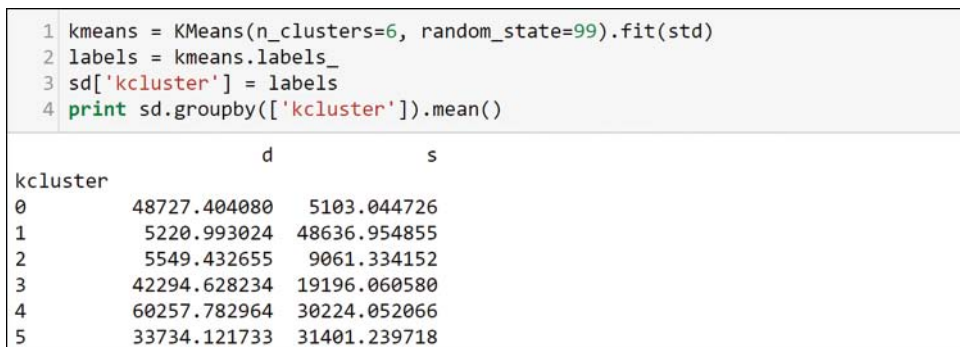


Figure 13-36 *Cluster Centroids for the K-means Clusters and Assigning Clusters to the Dataframe*

After running the algorithm, you copy the labels back to the dataframe. Unlike when clustering principal component analysis (PCA) and other computer dimension–reduced data, these numbers have meaning as is. You can see that cluster 0 has low average sources and high average destinations. Servers are on low ports, and hosts generally use high ports as the other end of the connection to servers. Cluster 0 is your best guess at possible servers. Cluster 1 looks like a place to find more clients. Other clusters are not conclusive, but you can examine a few later to see what you find. Figure 13-37 shows how to create individual dataframes to use as the overlays on your scatterplot.

```

1 kdf0=sd[sd.kcluster==0]
2 kdf1=sd[sd.kcluster==1]
3 kdf2=sd[sd.kcluster==2]
4 kdf3=sd[sd.kcluster==3]
5 kdf4=sd[sd.kcluster==4]
6 kdf5=sd[sd.kcluster==5]
7 sd.kcluster.value_counts()

3    43
0    27
5    22
1    13
2    11
4     7
Name: kcluster, dtype: int64

```

Figure 13-37 *Using Cluster Values to Filter out Interesting Dataframes*

You can see here that there are 27 possible servers in cluster 0 and 13 possible hosts in cluster 1. You can plot all of these clusters together, using the plot definition in Figure 13-38.

```

1 plt.scatter(kdf0['s'],kdf0['d'],s=20,label="Cluster0")
2 plt.scatter(kdf1['s'],kdf1['d'],s=20,marker="^",label="Cluster1")
3 plt.scatter(kdf2['s'],kdf2['d'],s=20,marker="*",label="Cluster2")
4 plt.scatter(kdf3['s'],kdf3['d'],s=20,marker="<",label="Cluster3")
5 plt.scatter(kdf4['s'],kdf4['d'],s=20,marker=">",label="Cluster4")
6 plt.scatter(kdf5['s'],kdf5['d'],s=20,marker=".",label="Cluster5")
7 plt.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
8 plt.title("Host Port Characterization Clusters", fontsize=12)
9 plt.xlabel("Source Port Average Communicating with this Host")
10 plt.ylabel("Destination Port Average")
11 plt.show()

```

Figure 13-38 *Cluster Scatterplot Definition for Average Port Clustering*

This definition results in the plot in Figure 13-39.

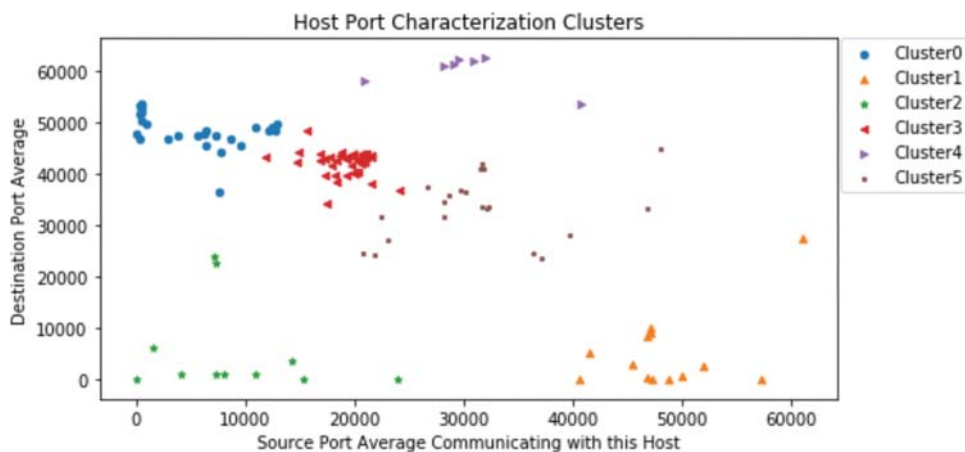


Figure 13-39 Scatterplot of Average Source and Destination Ports per Host

Notice that the clusters identified as interesting are in the upper-left and lower-right corners, and other hosts are scattered over a wide band on the opposite diagonal. Because you believe that cluster 0 contains servers by the port profile, you can use the loop in Figure 13-40 to generate a long list of profiles. Then you can browse each of the profiles of the hosts in that cluster. The results are very long because you loop through the host profile 27 times. But browsing a machine learning filtered set is much faster than browsing profiles of all hosts. Other server assets with source ports in the low ranges clearly emerge. You may recognize the 443 and 22 pattern as a possible VMware host. Here are a few examples of the per host patterns that you can find with this method:

192.168.207.4 source ports UDP -----

53 6330

192.168.21.254 source ports TCP ----- (Saw this pattern many times)

443 10087

22 2949

You can add these assets to the asset table. If you were programmatically developing a diagram or graph, you could add them programmatically.

The result of looking for servers here is quite interesting. You have found assets, but more importantly, you have found additional scanning that shows up across all possible servers. Some servers have 7 to 10 packets for every known server port. Therefore, the finding for cluster 0 had a secondary use for finding hosts that are scanning sets of popular server ports. A few of the scanning hosts show up on many other hosts, such as 192.168.202.96 in Figure 13-40, where you can see the output of host conversations from your function.

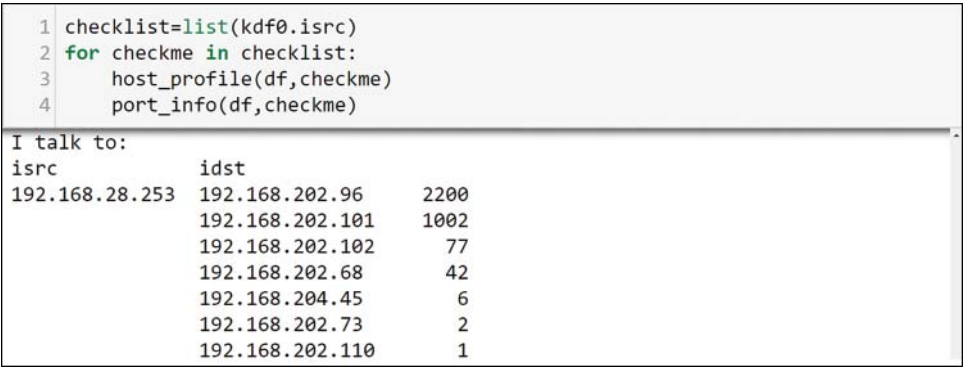


Figure 13-40 Destination Hosts Talking to 192.168.28.102

If you check the detailed port profiles of the scanning hosts that you have identified so far and overlay them as another entry to your scatterplot, you can see, as in Figure 13-41, that they are hiding in multiple clusters, some of which appear in the space you identified as clients. This makes sense because they have high port numbers on the response side.

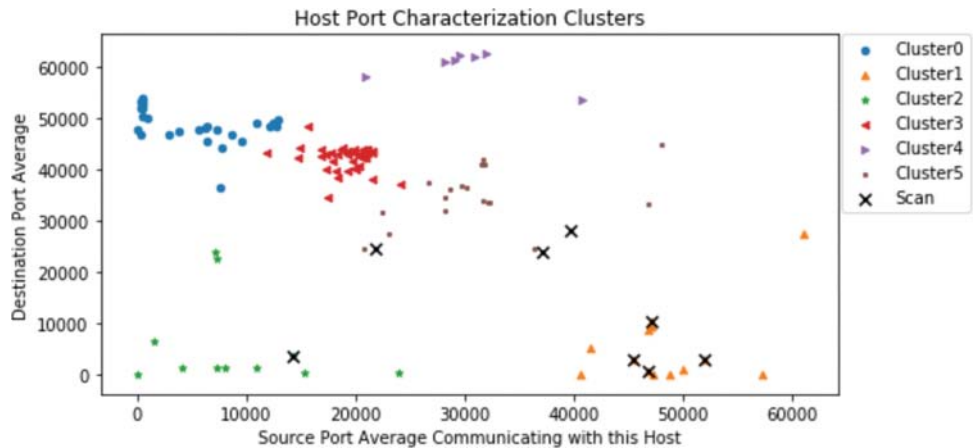


Figure 13-41 Overlay of Hosts Found to Be Scanning TCP Ports on the Network

You expected to find scanners in client cluster 1. These hosts are using many low destination ports, as reflected by their graph positions. Some hosts may be attempting to hide per-port scanning activity by equally scanning all ports, including the high ones. This shows up across the middle of this “average port” perspective that you are using. You have already identified some of these ports. By examining the rest of cluster 1 using the same loop, you find these additional insights from the profiles in there:

- Host 192.168.202.109 appears to be a Secure Shell (SSH) client, opening sessions on the servers that were identified as possible VMware servers from cluster 0 (443 and 22).

- Host 192.168.202.76, which was identified as a possible scanner, is talking to many IP addresses outside your domain. This could indicate exfiltration or web crawling.
- Host 192.168.202.79 has a unique activity pattern that could be a VMware functionality or a compromised host. You should add it to the list to investigate.
- Other hosts appear to have activity related to web surfing or VMware as well.

You can spend as much time as you like reviewing this information from the SME clustering perspective, and you will find interesting data across the clusters. See if you can find the following to test your skills:

- A cluster has some interesting groups using 11xx and 44xx. Can you map them?
- A cluster also has someone answering DHCP requests. Can you find it?
- A cluster has some interesting communications at some unexpected high ports. Can you find them?

This is a highly active environment, and you could spend a lot of time identifying more scanners and more targets. Finding legitimate servers and hosts is a huge challenge. There appears to be little security and segmentation, so it is a chaotic situation at the data plane layer in this environment. Whitelisting policy would be a huge help! Without policy, cleaning and securing this environment is an iterative and ongoing process. So far, you have used SME and SME profiling skills along with machine learning clustering to find items of interest to you as a data plane investigator.

You will find more items that are interesting in the data if you keep digging. You have not, for example, checked traffic that is using Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP), Bootstrap Protocol (BOOTP), Domain Name System (DNS), or Address Resolution Protocol (ARP). You have not dug into all the interesting port combinations and patterns that you have seen. All these protocols have purposes on networks. With a little research, you can identify legitimate usage versus attempted exploits. You have the data and the skills. Spend some time to see what you can find. This type of deliberate practice will benefit you. If you find something interesting, you can build an automated way to identify and parse it out. You have an atomic component that you can use on any set of packets that you bring in.

The following section moves on from the SME perspective and explores unsupervised machine learning.

Machine Learning: Creating Full Port Profiles

So far in this chapter, you have used your human evaluation of the traffic and looked at port behaviors. This section explores ways to hand profiles to machine learning to see what you can learn. To keep the examples simple, only source and destination TCP and UDP ports are used, as shown in Figure 13-42. However, you could use any of the fields to build host profiles for machine learning. Let's look at how this compares to the SME approach you have just tried.

```
1 h=df[['isrc','tdport']].copy()
2 h.rename(columns={"isrc": "host", "tdport": "ports"}, inplace=True)
3 h1=df[['isrc','tsport']].copy()
4 h1.rename(columns={"isrc": "host", "tsport": "ports"}, inplace=True)
5 h2=df[['isrc','utSPORT']].copy()
6 h2.rename(columns={"isrc": "host", "utSPORT": "ports"}, inplace=True)
7 h3=df[['isrc','utdport']].copy()
8 h3.rename(columns={"isrc": "host", "utdport": "ports"}, inplace=True)
9 h5=pd.concat([h,h1,h2,h3])
10 h6=h5.reset_index()
11 del h6['timestamp']
```

Figure 13-42 Building a Port Profile Signature per IP Host

In this example, you will create a dataframe for each aspect you want to add to a host profile. You will use only the source and destination ports from the data. By copying each set to a new dataframe and renaming the columns to the same thing (isrc=host, and any TCP or UDP port=ports), you can concatenate all the possible entries to a single dataframe that has any host and any port that it used, regardless of direction or protocol (TCP or UDP). You do not need the timestamp, so you can pull it out as the index in row 10 where you define a new simple numbered index with reset_index and delete it in row 11. You will have many duplicates and possibly some empty columns, and Figure 13-43 shows how you can work more on this feature engineering exercise.

```
1 h7=h6.drop_duplicates(['host','ports'])
2 h8=h7.dropna().copy()
3 h8['ports']=h8['ports'].astype(str)
4 h8['host']=h8['host'].astype(str)
5 h8['portprofile']=h8.groupby(['host'])['ports']\
6   .transform(lambda x: ' '.join(x))
```

Figure 13-43 Creating a Single String Host Port Profile

To use string functions to combine the items into a single profile, you need to convert everything to a text type in rows 3 and 4, and then you can join it all together into a string in a new column in line 5. After you do this combination, you can delete the duplicate profiles, as shown in Figure 13-44.

```
1 h9=h8.drop_duplicates(['host','portprofile']).reset_index().copy()
2 del h9['ports']
3 del h9['index']
4 h9[:2]
```

	host	portprofile
0	192.168.229.254	46117 46119 46121 46123 46125 46127 46129 4613...
1	192.168.202.79	443 80 445 135 139 143 1 25 53 88 110 3306 326...

Figure 13-44 Deduplicating Port Profile to One per Host

Now you have a list of random-order profiles for each host. Because you have removed duplicates, you do not have counts but just a fingerprint of activities. Can you guess where we are going next? Now you can encode this for machine learning and evaluate the visualization components (see Figure 13-45) as before.

```

1 from sklearn.feature_extraction.text import CountVectorizer
2 vectorizer = CountVectorizer()
3 word_transform = vectorizer.fit_transform(h9['portprofile'])
4 X_matrix = word_transform.todense()

1 from sklearn.decomposition import PCA
2 pca = PCA(n_components=8)
3 pca.fit(X_matrix)
4 pca.explained_variance_ratio_

array([ 0.84225116,  0.04394625,  0.03400552,  0.0252997 ,  0.0054314 ,
        0.00422903,  0.00396515,  0.00381091])

```

Figure 13-45 Encoding the Port Profiles and Evaluating PCA Component Options

You can see from the PCA evaluation that one component defines most of the variability. Choose two to visualize and generate the components as shown in Figure 13-46.

```

1 pca_data = PCA(n_components=2).fit_transform(X_matrix)
2 pca1=[]
3 pca2=[]
4 for index, instance in enumerate(pca_data):
5     pca_1, pca_2 = pca_data[index]
6     pca1.append(pca_1)
7     pca2.append(pca_2)
8 print(len(pca1))
9 print(len(pca2))

174
174

```

Figure 13-46 Using PCA to Generate Two Dimensions for Port Profiles

You have 174 source senders after the filtering and duplicate removal. You can add them back to the dataframe as shown in Figure 13-47.

```

1 h9['pca1']=pca1
2 h9['pca2']=pca2
3 h9[:2]

```

	host	portprofile	pca1	pca2
0	192.168.229.254 46117 46119 46121 46123 46125 46127 46129 4613...		-33.378071	-18.344733
1	192.168.202.79 443 80 445 135 139 143 1 25 53 88 110 3306 326...		215.256411	-2.197714

Figure 13-47 Adding the Generated PCA Components to the Dataframe

Notice that the PCA reduced components are now in the dataframe. You know that there are many distinct patterns in your data. What do you expect to see with this machine learning process, using the patterns that you have defined? You know there are scanners, legitimate servers, clients, some special conversations, and many other possible dimensions. Choose six clusters to see how machine learning segments things. Your goal is to find interesting things for further investigation, so you can try other cluster numbers as well. The PCA already defined where it will appear on a plot. You are just looking for segmentation of unique groups at this point.

Figure 13-48 shows the plot definition. Recall that you simply add an additional dataframe view for every set of data you want to visualize. It is very easy to overlay more data later by adding another entry.

```
1 pyplot.rcParams["figure.figsize"] = (8,4)
2 pyplot.scatter(df0['pca2'],df0['pca1'],s=20,label="c0")
3 pyplot.scatter(df1['pca2'],df1['pca1'],s=20,marker="x",label="c1")
4 pyplot.scatter(df2['pca2'],df2['pca1'],s=20,marker="^",label="c2")
5 pyplot.scatter(df3['pca2'],df3['pca1'],s=20,marker=">",label="c3")
6 pyplot.scatter(df4['pca2'],df4['pca1'],s=20,marker="<",label="c4")
7 pyplot.scatter(df5['pca2'],df5['pca1'],s=20,marker="*",label="c5")
8 pyplot.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
9 pyplot.show()
```

Figure 13-48 Scatterplot Definition for Plotting PCA Components

Figure 13-49 shows the plot that results from this definition.

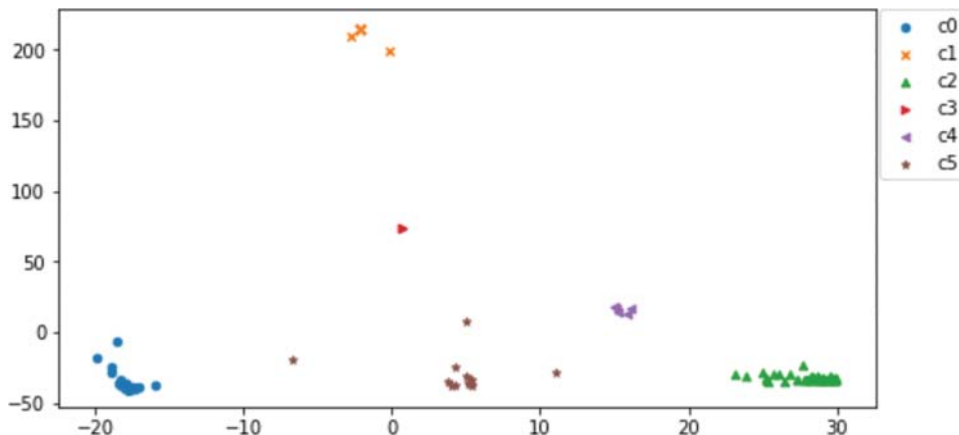


Figure 13-49 Scatterplot of Port Profile PCA Components

Well, this looks interesting. The plot has at least six clearly defined locations and a few outliers. You can see what this kind of clustering can show by examining the data behind what appears to be a single item in the center of the plot, cluster 3, in Figure 13-50.

1	df3										
	host					portprofile					
47	192.168.202.83	80	443	3389	1723	3306	110	993	256	8888	139 135...
48	192.168.206.44	53839	40497	39173	47342	38162	37552	50492	3495...	74.058252	0.648054
											3

Figure 13-50 *All Hosts in K-means Cluster 3*

What you learn here is that this cluster is very tight. What visually appears to be one entry is actually two. Do you recognize these hosts? If you check the table of items you have been gathering for investigation, you will find them as a potential scanner and the host that it is scanning.

If you consider the data you used to cluster, you may recognize that you built a clustering method that is showing affinity groups of items that are communicating with each other. The unordered source and destination port profiles of these hosts are the same. This can be useful for you. Recall that earlier in this chapter, you found a bunch of hosts with addresses ending in 254 that are communicating with something that appears to be a possible VMware server. Figure 13-51 shows how you filter some of them to see if they are related; as you can see here, they all fall into cluster 0.

1	h9[h9.host.str.endswith("254")]										
	host					portprofile					
0	192.168.229.254	46117	46119	46121	46123	46125	46127	46129	4613...	-33.378071	-18.344733
17	192.168.28.254	53633	51272	51283	51284	51285	46586	46597	4659...	-39.669804	-17.717913
23	192.168.23.254	35820	53768	53773	53774	53798	53799	53800	5380...	-39.167163	-17.871614
24	192.168.26.254	36254	22113	45992	45997	54061	54065	54066	5407...	-39.092159	-17.898002
											0

Figure 13-51 *Filtering to VMware Hosts with a Known End String*

Using this affinity, you are now closer to confirming a few other things you have noted earlier. This machine learning method is showing host conversation patterns that you were using your human brain to find from the loops that you were defining earlier. In Figure 13-52, look for the host that appears to be communicating to all the VMware hosts.

1	h9[h9.host=="192.168.202.76"]									
	host	portprofile			pca1	pca2	kcluster			
4	192.168.202.76	40006	40046	80 443 40030 40024 33033 40020 400...	-28.616951	-18.923414	0			

Figure 13-52 Finding a Possible vCenter Server in Same Cluster as the VMware Hosts

As expected, this host is also in cluster 0. You find this pattern of scanners in many of the clusters, so you add a few more hosts to your table of items to investigate.

This affinity method has proven useful in checking to see if there are scanners in all clusters. If you gather the suspect hosts that have been identified so far, you can create another dataframe view to add to your existing plot, as shown in Figure 13-53.

```
1 checked = ['192.168.202.110', '192.168.202.102', '192.168.202.81', \
2           '192.168.204.45', '192.168.202.73', '192.168.202.79', \
3           '192.168.202.96', '192.168.203.45', '192.168.202.83', \
4           '192.168.202.101']
5 sc=h9[h9.host.isin(checked)]
```

Figure 13-53 Building a Scatterplot Overlay for Hosts Suspected of Network Scanning

When you add this dataframe, you add a new row to the bottom of your plot definition and denote it with an enlarged marker, as shown on line 8 in Figure 13-54.

```
1 pyplot.rcParams["figure.figsize"] = (8,4)
2 pyplot.scatter(df0['pca2'], df0['pca1'], s=20, label="c0")
3 pyplot.scatter(df1['pca2'], df1['pca1'], s=20, marker="x", label="c1")
4 pyplot.scatter(df2['pca2'], df2['pca1'], s=20, marker="^", label="c2")
5 pyplot.scatter(df3['pca2'], df3['pca1'], s=20, marker=">", label="c3")
6 pyplot.scatter(df4['pca2'], df4['pca1'], s=20, marker="<", label="c4")
7 pyplot.scatter(df5['pca2'], df5['pca1'], s=20, marker="*", label="c5")
8 pyplot.scatter(sc['pca2'], sc['pca1'], s=60, marker="x", color='black', \
9               label="Scanners")
10 pyplot.legend(bbox_to_anchor=(1.005, 1), loc=2, borderaxespad=0.)
11 pyplot.show()
```

Figure 13-54 Adding the Network Scanning Hosts to the Scatterplot Definition

The resulting plot (see Figure 13-55) shows that you have identified many different affinity groups—and scanners within most of them—except for one cluster on the lower right.

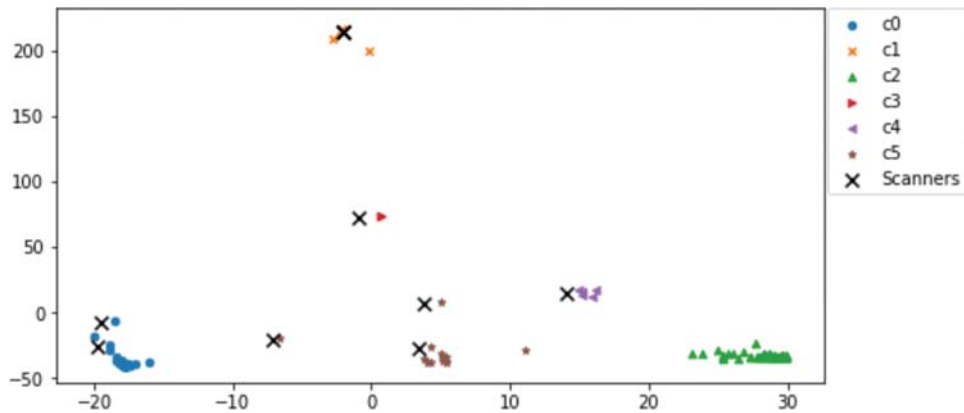


Figure 13-55 Scatterplot of Affinity Groups of Suspected Scanners and Hosts They Are Scanning

If you use the loop to go through each host in cluster 2, only one interesting profile emerges. Almost all hosts in cluster 2 have no heavy activity except for responses to between 4 and 10 packets each to scanners you have already identified, as well as a few minor services. This appears to be a set of devices that may not be vulnerable to the scanning activities or that may not be of interest to the scanning programs behind them. There were no obvious scanners in this cluster. But you have found scanning activity in every other cluster.

Machine Learning: Creating Source Port Profiles

This final section reuses the entire unsupervised analysis from the preceding section but with a focus on the source ports only. It uses the source port columns, as shown in Figure 13-56. The code for this section is a repeat of everything in this chapter since Figure 13-42, so you can make a copy of your work and use the same process. (The steps to do that are not shown here.)

```
1 h1=df[['isrc','tsport']].copy()
2 h1.rename(columns={"isrc": "host", "tsport": "ports"}, inplace=True)
3 h2=df[['isrc','utSPORT']].copy()
4 h2.rename(columns={"isrc": "host", "utSPORT": "ports"}, inplace=True)
5 h5=pd.concat([h1,h2])
6 h6=h5.reset_index()
7 del h6['timestamp']
```

Figure 13-56 Defining per-Host Port Profiles of Source Ports Only

You can use this smaller port set to run through the code used in the previous section with minor changes along the way. Using six clusters with K-means yielded some clusters with very small values. Backing down to five clusters for this analysis provides better

results. At only a few minutes per try, you can test any number of clusters. Look at the clusters in the scatterplot for this analysis in Figure 13-57.

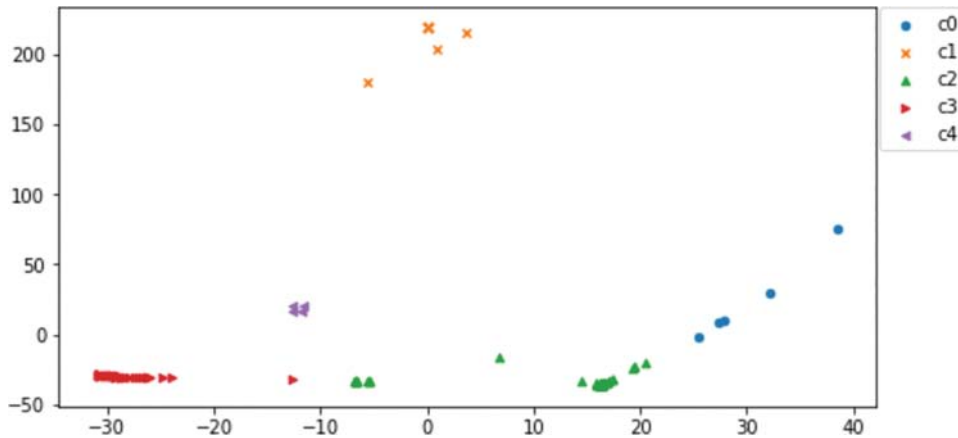


Figure 13-57 Scatterplot of Source-Only Port Profile PCA Components

You immediately see that the data plots differently here than with the earlier affinity clustering. Here you are only looking at host source ports. This means you are looking at a profile of the host and the ports used but not including any information about *who* was using the ports (the destination host's port). This profile also includes the ports that the host will use as the client side of services accessed on the network. Therefore, you are getting a first-person view from each host for services they provided, and services they requested from other hosts.

Recall the suspected scanner hosts dataframes that were generated as shown in Figure 13-58.

```
1 checked = ['192.168.202.110', '192.168.202.102', '192.168.202.81', \
2           '192.168.204.45', '192.168.202.73', '192.168.202.79', \
3           '192.168.202.96', '192.168.203.45', '192.168.202.83', \
4           '192.168.202.101']
5 sc=h9[h9.host.isin(checked)]
```

Figure 13-58 Creating a New Scatterplot Overlay for Suspected Scanning Hosts

When you overlay your scanner dataframe on the plot, as shown in Figure 13-59, you see that you have an entirely new perspective on the data when you profile source ports only. This is very valuable for you in terms of learning. These are the very same hosts as before, but with different feature engineering, machine learning sees them entirely differently. You have spent a large amount of time in this book looking at how to manipulate the data to engineer the machine learning inputs in specific ways. Now you know why feature

engineering is important: You can get an entirely different perspective on the same set of data by reengineering features.

Figure 13-59 shows that cluster 0 is full of scanners (the c0 dots are under the scanner Xs).

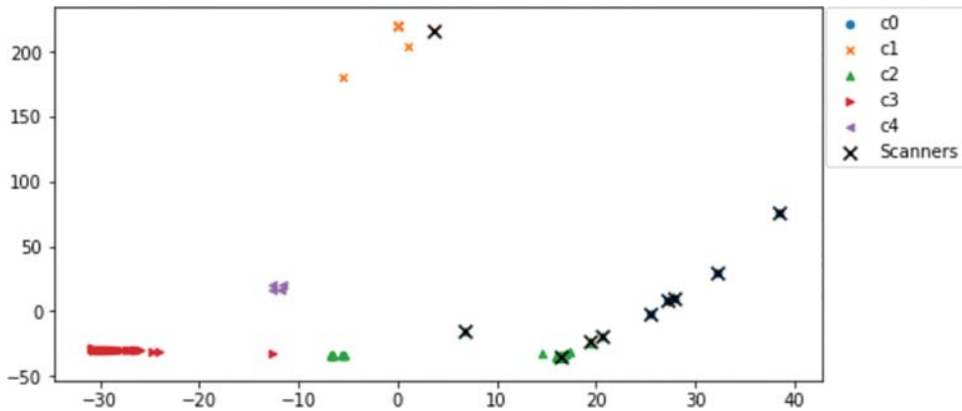


Figure 13-59 *Overlay of Suspected Scanning Hosts on Source Port PCA*

Almost every scanner identified in the analysis so far is on the right side of the diagram. In Figure 13-60, you can see that cluster 0 consists entirely of hosts that you have already identified as scanners. Their different patterns of scanning represent variations within their own cluster, but they are still far away from other hosts. You have an interesting new way to identify possible bad actors in the data.

1	df0										
	host	portprofile							pca1	pca2	kcluster
1	192.168.202.79	46117	50463	50465	55173	46119	50467	46121	8.645602	27.277811	0
								5046...			
47	192.168.202.83	53839	40497	39173	47342	38162	37552	50492	76.080197	38.560548	0
								3495...			
49	192.168.202.81	49174	40270	46822	34893	53775	57555	57156	-2.091625	25.413788	0
								5446...			
130	192.168.203.45	1337	31337	5432	54872	39161	57374	35183	10.116133	27.845451	0
								41061 ...			
135	192.168.204.45	37988	37989	37990	37994	37995	37996	46500	29.602374	32.184386	0
								5900...			

Figure 13-60 *Full Cluster of Hosts Scanning the Network*

The book use case ends here, but you have many possible next steps in this space. Using what you have learned throughout this book, here are a few ideas:

- Create similarity indexes for these hosts and look up any new host profile to see if it behaves like the bad profiles you have identified.
- Wrap the functions you created in this chapter in web interfaces to create host profile lookup tools for your users.
- Add labels to port profiles just as you added crash labels to device profiles. Then develop classifiers for traffic on your networks.
- Use profiles to aid in development of your own policies to use in the new intent-based networking (IBN) paradigm.
- Automate all this into a new system. If you add in supervised learning and some artificial intelligence, you could build the next big startup.

Okay, maybe the last one is a bit of a stretch, but why aim low?

Asset Discovery

Table 13-2 lists many of the possible assets discovered while analyzing the packet data in this chapter. This is all speculation until you validate the findings, but this gives you a good idea of the insights you can find in packet data. Keep in mind that this is a short list from a subset of ports. Examining all ports combined with patterns of use could result in a longer table with much more detail.

Table 13-2 *Interesting Assets Discovered During Analysis*

Asset	How You Found It
Layer 3 with 21 VLANs at 192.168.x.1	Found EIGRP routing protocol
DNS at 192.168.207.4	Port 53
Many web server interfaces	Lots of port 80, 443, 8000, and 8080
Windows NetBIOS activity	Ports 137, 138, and 139
BOOTP and DHCP from VLAN helpers	Ports 67, 68, and 69
Time server at 192.168.208.18	Port 123
Squid web proxy at 192.168.27.102	Port 3128
MySQL database at 192.168.21.203	Port 3306
PostgreSQL database at 192.168.203.45	Port 5432
Splunk admin port at 192.168.27.253	Port 8089
VMware ESXI 192.168.205.253	443 and 902
Splunk admin at 192.168.21.253	8089

Asset	How You Found It
Web and SSH 192.168.21.254	443 and 22
Web and SSH 192.168.22.254	443 and 22
Web and SSH 192.168.229.254	443 and 22
Web and SSH 192.168.23.254	443 and 22
Web and SSH 192.168.24.254	443 and 22
Web and SSH 192.168.26.254	443 and 22
Web and SSH 192.168.27.254	443 and 22
Web and SSH 192.168.28.254	443 and 22
Possible vCenter 192.168.202.76	Appears to be connected to many possible VMware hosts

Investigation Task List

Table 13-3 lists the hosts and interesting port uses identified while browsing the data in this chapter. These could be possible scanners on the network or targets of scans or attacks on the network. In some cases, they are just unknown hotspots you want to know more about. This list could also contain many more action items from this data set. If you loaded the data, continue to work with it to see what else you can find.

Table 13-3 *Hosts That Need Further Investigation*

To Investigate	Observed Behavior
192.168.202.110	Probed all TCP ports on thousands of hosts
192.168.206.44	1000 ports very active, seem to be under attack by one host
192.168.202.83	Hitting 206.44 on many ports
192.168.202.81	Database or discovery server?
192.168.204.45	Probed all TCP ports on thousands of hosts
192.168.202.73	DoS attack on port 445?
192.168.202.96	Possible scanner on 1000 specific low ports
192.168.202.102	Possible scanner on 200 specific low ports
192.168.202.79	Unique activity, possible scan and pull; maybe VMware
192.168.202.101	Possible scanner on 10,000 specific low ports
192.168.203.45	Scanning segment 21

To Investigate	Observed Behavior
Many	Well-known discovery ports 427, 1900, and others being probed
Many	Group using unknown 55553-4 for an unknown application
Many	Group using unknown 11xx for an unknown application

Summary

In this chapter, you have learned how to take any standard packet capture file and get it loaded into a useful dataframe structure for analysis. If you captured traffic from your own environment, you could now recognize clients, servers, and patterns of use for different types of components on the network. After four chapters of use cases, you now know how to manipulate the data to search, filter, slice, dice, and group to find any perspective you want to review. You can perform the same functions that many basic packet analysis packages provide. You can write your own functions to do things those packages cannot do.

You have also learned how to combine your SME knowledge with programming and visualization techniques to examine packet data in new ways. You can make your own SME data (part of feature engineering) and combine it with data from the data set to find new interesting perspectives. Just like innovation, sometimes analysis is about taking many perspectives.

You have learned two new ways to use unsupervised machine learning on profiles. You have seen that the output of unsupervised machine learning varies widely, depending on the inputs you choose (feature engineering again). Each method and perspective can provide new insight to the overall analysis. You have seen how to create affinity clusters of bad actors and their targets, as well as how to separate the bad actors into separate clusters.

You have made it through the use-case chapters. You have seen in Chapters 10 through 13 how to take the same machine learning technique, do some creative feature engineering, and apply it to data from entirely different domains (device data, syslogs, and packets). You have found insights in all of them. You can do this with each machine learning algorithm or technique that you learn. Do not be afraid to use your LED flashlight as a hammer. Apply to your own situation use cases from other industries and algorithms used for other purposes. You may or may not find insights, but you will learn something.

Cisco Analytics

As you know by now, this book is not about Cisco analytics products. You have learned how to develop innovative analytics solutions by taking new perspectives to develop atomic parts that you can grow into full use cases for your company. However, you do not have to start from scratch with all the data and the atomic components. Sometimes you can source them directly from available products and services.

This chapter takes a quick trip through the major pockets of analytics from Cisco. It includes no code, no algorithms, and no detailed analysis. It introduces the major Cisco platforms related to your environment so you can spend your time building new solutions and gaining insights and data from Cisco solutions that you already have in place. You can bring analytics and data from these platforms into your solutions, or you can use your solutions as customized add-ons to these environments. You can use these platforms to operationalize what you build.

In this book, you have learned how to create some of the very same analytics that Cisco uses within its Business Critical Insights (BCI), Migration Analytics, and Service Assurance Analytics areas (see Figure 14-1). This book only scratches the surface of the analytics used to support customers in those service offers. A broad spectrum of analytics is not addressed anywhere in this book. Cisco offers a wide array of analytics used internally and provided in products for customers to use directly. Figure 14-1 shows the best fit for these products and services in your environment.

Cisco has additional analytics built into Services offerings that focus on other enterprise needs, such as IoT analytics, architecture, and advisory services for building analytics solutions and automation/orchestration analytics for building full-service assurance platforms for networks. Cisco Managed Services (CMS) uses analytics to enhance customer networks that are fully managed by Cisco.

In the product space, Cisco offers analytics solutions for the following:

- IoT with Jasper and Kinetic
- Security with Stealthwatch
- Campus, wide area network, and wireless with digital network architecture (DNA) solutions
- Deep application analysis with AppDynamics
- Data center with Tetration

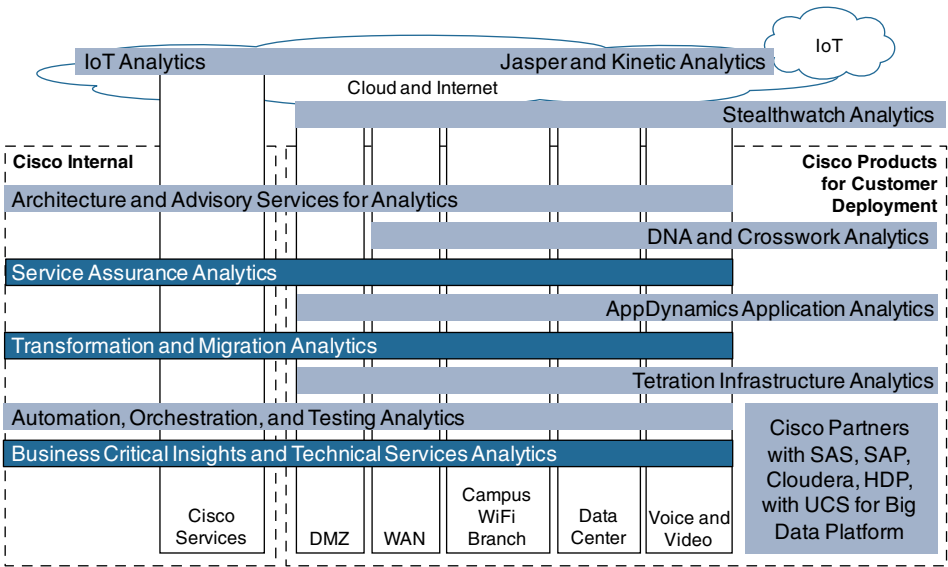


Figure 14-1 *Cisco Analytics Products and Services*

Architecture and Advisory Services for Analytics

As shown in Figure 14-1, you can get many analytics products and services from Cisco. You can uncover the feasibility and viability of these service offers or analytics products for your business by engaging Cisco Services. The workshops, planning, insights, and requirements assessment from these services will help your business, regardless of whether you engage further with Cisco.

For more about architecture and advisory services for analytics, see <https://www.cisco.com/c/en/us/services/advisory.html>.

Over the years, Cisco has seen more possible network situations than any other company. You can take advantage of these lessons learned to avoid taking paths that may end in undesirable outcomes.

Stealthwatch

Security is a common concern in any networking department. From visibility to policy enforcement, to data gathering and Encrypted Traffic Analytics (ETA), Stealthwatch (see Figure 14-2) provides the enterprise-wide visibility and policy enforcement you need at a foundational level.

For more about Stealthwatch, see <https://www.cisco.com/c/en/us/products/security/stealthwatch/index.html>.

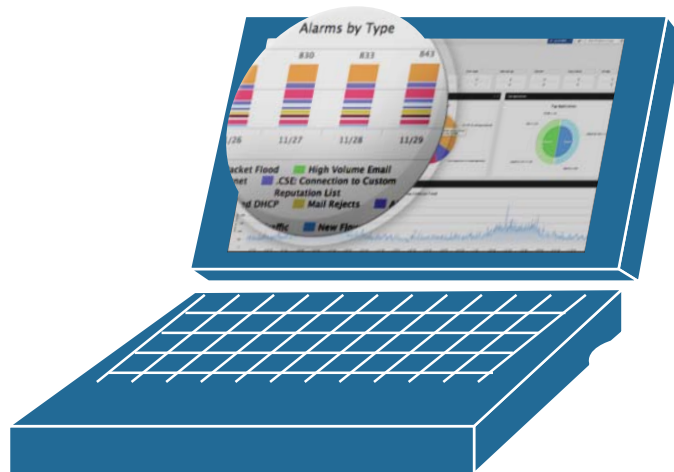


Figure 14-2 *Cisco Stealthwatch*

Stealthwatch can cover all your assets, including those that are internal, Internet facing, or in the cloud. Stealthwatch uses real-time telemetry data to detect and remediate advanced threats. You can use Stealthwatch with any Cisco or third-party product or technology. Stealthwatch directly integrates with Cisco Identity Service Engine (ISE) and Cisco TrustSec. Stealthwatch also includes the ability to analyze encrypted traffic with ETA (see <https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-security/eta.html>).

You can use Stealthwatch out of the box as a premium platform, or you can use Stealthwatch data to provide additional context to your own solutions and use cases.

Digital Network Architecture (DNA)

Cisco Digital Network Architecture (DNA) is an architectural approach that brings intent-based networking (IBN) to the campus, wide area networks (WANs), and branch local area networks (LANs), both wired and wireless. Cisco DNA is about moving your infrastructure from a box configuration paradigm to a fully automated network environment with complete service assurance, automation, and analytics built right in. For more information, see <https://www.cisco.com/c/en/us/solutions/enterprise-networks/index.html>.

Cisco DNA incorporates many years of learning from Cisco into an automated system that you can deploy in your own environment. Thanks to the incorporation of these years of learning, you can operate DNA technologies such as Secure Defined Access (SDA), Intelligent Wide Area Networks (iWAN), and wireless with a web browser and a defined policy. Cisco has integrated years of learning from customer environments into the assurance workflow to provide automated and guided remediation, as shown in Figure 14-3.

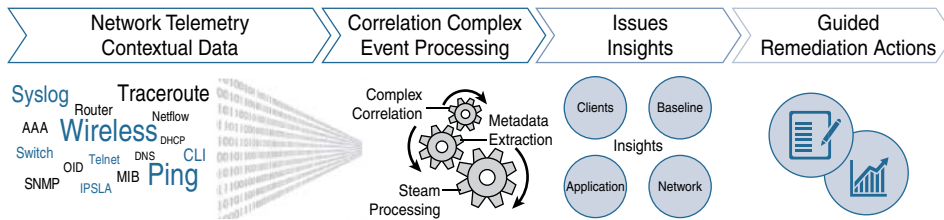


Figure 14-3 Cisco Digital Network Architecture (DNA) Analytics

If you want to explore on your own, you can access data from the centralized DNA Center (DNAC), which is the source of the DNA architecture. You can use context from DNAC in your own solutions in a variety of areas. Benefits of DNA include the following:

- Infrastructure visualization (network topology auto-discovery)
- User visualization, policy visualization, and user policy violation
- Service assurance, including the interlock between assurance and provisioning
- Closed-loop assurance and automation (self-driving and self-healing networks)
- An extensible platform that enables third-party apps
- A modular microservices-based architecture
- End-to-end real-time visibility of the network, clients, and applications
- Proactive and predictive insights with guided remediation

AppDynamics

Shifting focus from the broad enterprise to the application layer, you can secure, analyze, and optimize the applications that support your business to a very deep level with AppDynamics (see <https://www.appdynamics.com>). You can secure, optimize, and

analyze the data center infrastructure underlay that supports these applications with Tetration (see next section). AppDynamics and Tetration together cover all aspects of the data center from applications to infrastructure. Cisco acquired AppDynamics in 2017. For an overview of the AppDynamics architecture, see Figure 14-4.

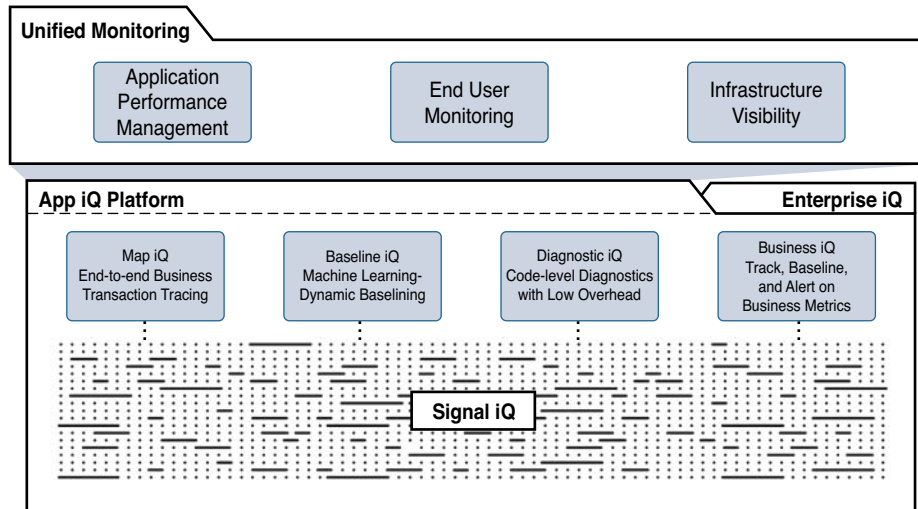


Figure 14-4 *Cisco AppDynamics Analytics Engines*

AppDynamics monitors application deployments from many different perspectives—and you know the value of using different perspectives to uncover innovations. AppDynamics uses intelligence engines that collect and centralize real-time data to identify and visualize the details of individual applications and transactions.

AppDynamics uses machine learning and anomaly detection as part of the foundational platform, and it uses these for both application-diagnostic and business intelligence.

Benefits of AppDynamics include the following:

- Provides real-time business, user, and application insights in one environment
- Reduces MTTR (mean time to resolution) through early detection of application and user experience problems
- Reduces incident cost and improves the quality of applications in your environment
- Provides accurate and near-real-time business impact analysis on top of application performance impact
- Provides a rich end-to-end view from the customer to the application code and in between

AppDynamics performance management solutions are built on and powered by the App iQ Platform, developed over many years based on understanding of complex enterprise applications. The App iQ platform features six proprietary performance engines that give customers the ability to thrive in that complexity.

You can use AppDynamics data and reporting as additional context and guidance for where to target your new infrastructure analytics use cases. AppDynamics provides Cisco's deepest level of application analytics.

Tetration

Tetration infrastructure analytics integrates with the data center and cloud fabric that support business applications. Tetration surrounds critical business applications with many layers of capability, including policy, security, visibility, and segmentation. Cisco built Tetration from the ground up specifically for data center and Application Centric Infrastructure (ACI) environments. Your data center or hybrid cloud data layer is unique and custom built, and it requires analytics with that perspective. Tetration (see Figure 14-5) is custom built for such environments. For more information about Tetration, see <https://www.cisco.com/c/en/us/products/data-center-analytics/index.html>.

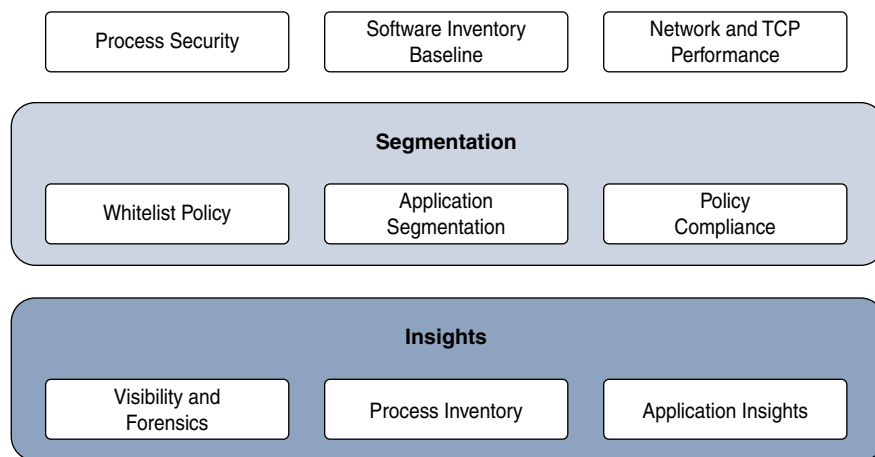


Figure 14-5 *Cisco Tetration Analytics*

Tetration offers full visibility into software and process inventory, as well as forensics, security, and applications; it is similar to enterprise-wide Stealthwatch but is for the data center. Cisco specifically designed Tetration with a deep-dive focus on data and cloud application environments, where it offers the following features:

- Flow-based unsupervised machine learning for discovery
- Whitelisting group development for policy-based networking

In Figure 14-6 you may notice many of the same things you learned to use in your solutions in the previous chapters. Crosswork is also extensible and can be a place where you implement your use case or atomic components. With Crosswork as a starter kit, you can build your analysis into fully automated solutions. Crosswork is a full-service assurance solution that includes automation.

IoT Analytics

The number of connected devices on the Internet is already in the billions. Cisco has platforms to manage both the networking and analytics required for massive-scale deployments of Internet of Things (IoT) devices. Cisco Jasper (<https://www.jasper.com>) is Cisco's intent-based networking (IBN) control, connectivity, and data access method for IoT. As shown in Figure 14-7, Jasper can connect all the IoT devices from all areas of your business.

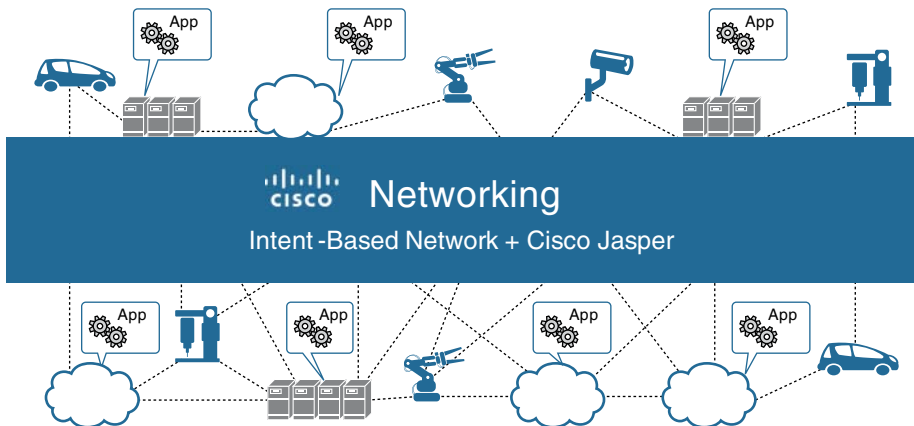


Figure 14-7 *Cisco Jasper IoT Networking*

Cisco Kinetic is Cisco's data platform for IoT analytics (see <https://www.cisco.com/c/en/us/solutions/internet-of-things/iot-kinetic.html>).

When you have connectivity established with Jasper, the challenge moves to having the right data and analysis in the right places. Cisco Kinetic (see Figure 14-8) was custom built for data and analytics in IoT environments. Cisco Kinetic makes it easy to connect distributed devices (“things”) to the network and then extract, normalize, and securely move data from those devices to distributed applications. In addition, this platform plays a vital role in enforcing policies defined by data owners in terms of which data goes where and when.

Note As mentioned in Chapter 4, “Accessing Data from Network Components,” service providers (SP) typically offer these IoT platforms to their customers, and data access for your IoT-related analysis may be dependent upon your specific deployment and SP capabilities.

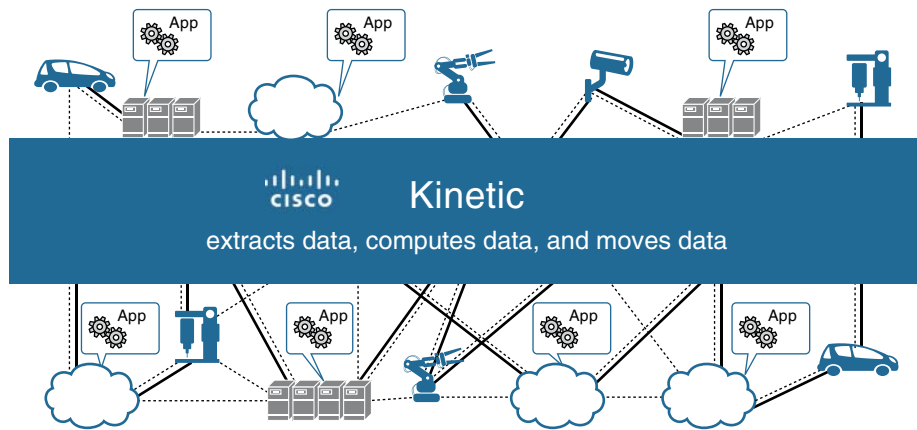


Figure 14-8 *Cisco Kinetic IoT Analytics*

Analytics Platforms and Partnerships

Cisco has many partnerships with analytics software and solution companies, including the following:

- SAS: https://www.sas.com/en_us/partners/find-a-partner/alliance-partners/Cisco.html
- IBM: <https://www.ibm.com/blogs/internet-of-things/ibm-and-cisco/>
- Cloudera: <https://www.cloudera.com/partners/solutions/cisco.html>
- Hortonworks: <https://hortonworks.com/partner/cisco/>

If you have analytics platforms in place, the odds are that Cisco built an architecture or solution with your vendor to maximize the effectiveness of that platform. Check with your provider to understand where it collaborates with Cisco.

Cisco Open Source Platform

Cisco provides analytics to the open source community in many places. Platform for Network Data Analytics (PNDA) is an open source platform built by Cisco and put into the open source community. You can download and install PNDA from <http://pnda.io/>. PNDA is a complete platform (see Figure 14-9) that you can use to build the entire data engine of the analytics infrastructure model.

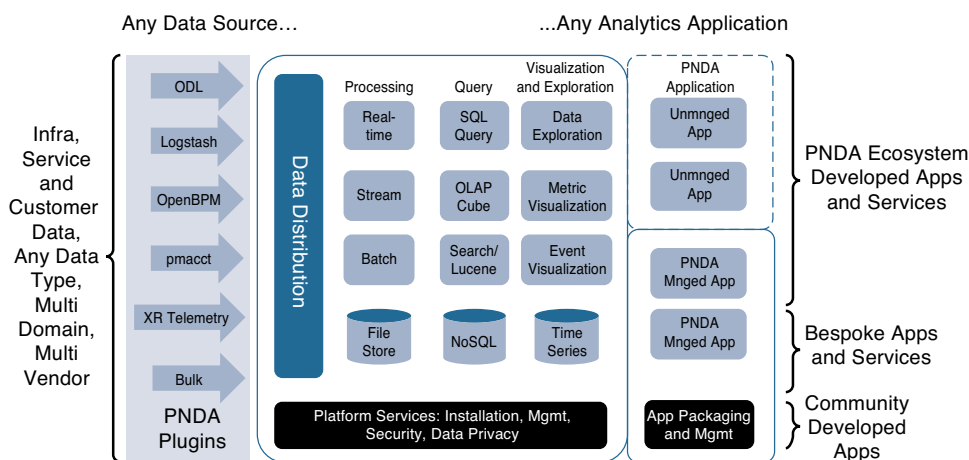


Figure 14-9 Platform for Network Data Analytics (PNDA)

Summary

The point of this short chapter is to let you know how Cisco can help with analytics products, services, or data sources for your own analytics platforms. Cisco has many other analytics capabilities that are part of other products, architectures, and solutions. Only the biggest ones are highlighted here because you can integrate solutions and use cases that you develop into these platforms.

Your company has many analytics requirements. In some cases, it is best to build your own customized solutions. In other cases, it makes more sense to accelerate your analytics use-case development by bringing in a full platform that moves you well along the path toward predictive, preemptive, and prescriptive capability. Then you can add your own solution enhancements and customization on top.

Book Summary

I would like to start this final chapter by thanking you for choosing this book. I realize that you have many choices and limited time. I hope you found that spending your time reading this book was worthwhile for you and that you learned more about analytics solutions and use cases related to computer data networking. If you were able to generate a single business-affecting idea, then it was all worth it.

Today everything is connected, and data is widely available. You build data analysis components and assemble complex solutions from atomic parts. You can combine them with stakeholder workflows and other complex solutions. You now have the foundation you need to get started assembling your own solutions, workflows, automations, and insights into use cases. Save your work and save your atomic parts. As you gain more skills, you will improve and add to them. As you saw in the use-case chapters of this book (Chapters 10, “Developing Real Use Cases: The Power of Statistics,” 11, “Developing Real Use Cases: Network Infrastructure Analytics,” 12, “Developing Real Use Cases: Control Plane Analytics Using Syslog Telemetry,” and 13, “Developing Real Use Cases: Data Plane Analytics”), there are some foundational techniques that you will use repeatedly, such as working with data in dataframes, working with text, and exploring data with statistics and unsupervised learning.

If you have opened up your mind and looked into the examples and innovation ideas described in this book, you realize that analytics is everywhere, and it touches many parts of your business. In this chapter I summarize what I hope you learned as you went through the broad journey starting from networking and traversing through analytics solution development, bias, innovation, algorithms, and real use cases.

While the focus here is getting you started with analytics in the networking domain, the same concepts apply to data from many other industries. You may have noticed that in this book, you often took a single idea, such as Internet search encoding, and used it for searching; dimensionality reduction; and clustering for device data, network device logs, and network packets. When you learn a technique and understand how to apply it, you can use your SME side to determine how to make your data fit that technique.

You can do this one by one with popular algorithms, and you will find amazing insights

in your own data. This chapter goes through one final summary of what I hope you learned from this book.

Analytics Introduction and Methodology

In Chapter 1, “Getting Started with Analytics,” I identified that you would be provided depth in the areas of networking data, innovation and bias, analytics use cases, and data science algorithms (see Figure 15-1).

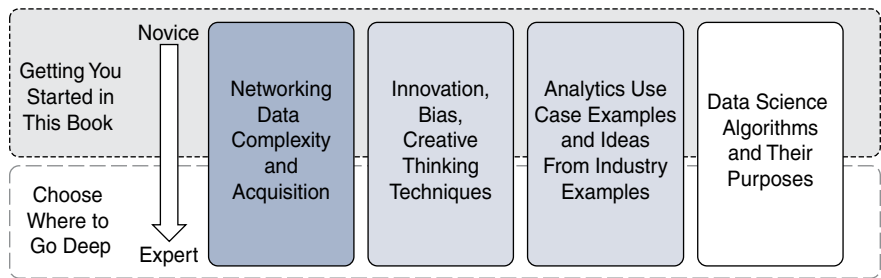


Figure 15-1 *Your Learning from This Book*

You should now have a foundational level of knowledge in each of these areas that you can use to further research and start your deliberate practice for moving to the expert level in your area of interest.

Also in Chapter 1, you first saw the diagram shown in Figure 15-2 to broaden your awareness of the perspective in analytics in the media. You may already be thinking about how to move to the right if you followed along with any of your own data in the use-case chapters.

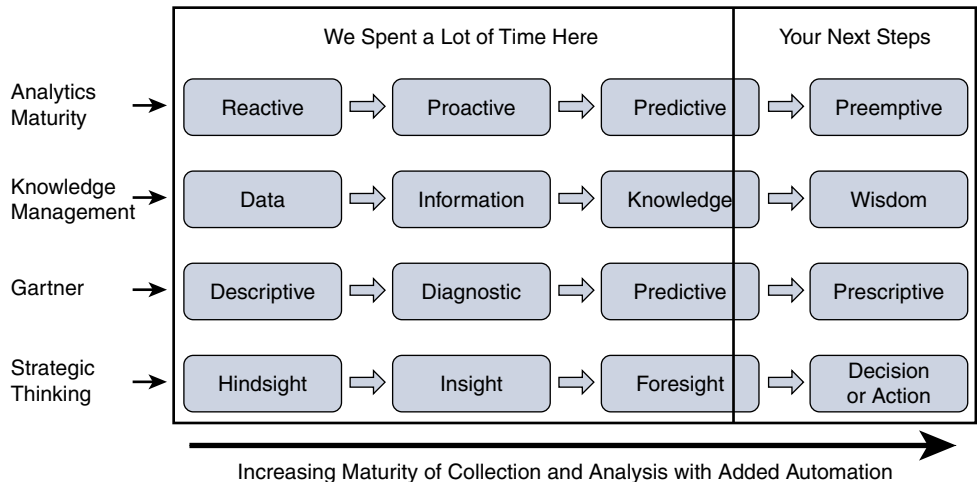


Figure 15-2 *Analytics Scales to Measure Your Level*

I hope that you are approaching or surpassing the line in the middle and thinking about how your solutions can be preemptive and prescriptive. Think about how to make wise decisions about the actions you take, given the insights you discover in your data.

In Chapter 2, “Approaches for Analytics and Data Science,” you learned a generalized flow (see Figure 15-3) for high-level thinking about what you need to do to put together a full use case. You should now feel comfortable working on any area of the analytics solutions using this simple process as a guideline.

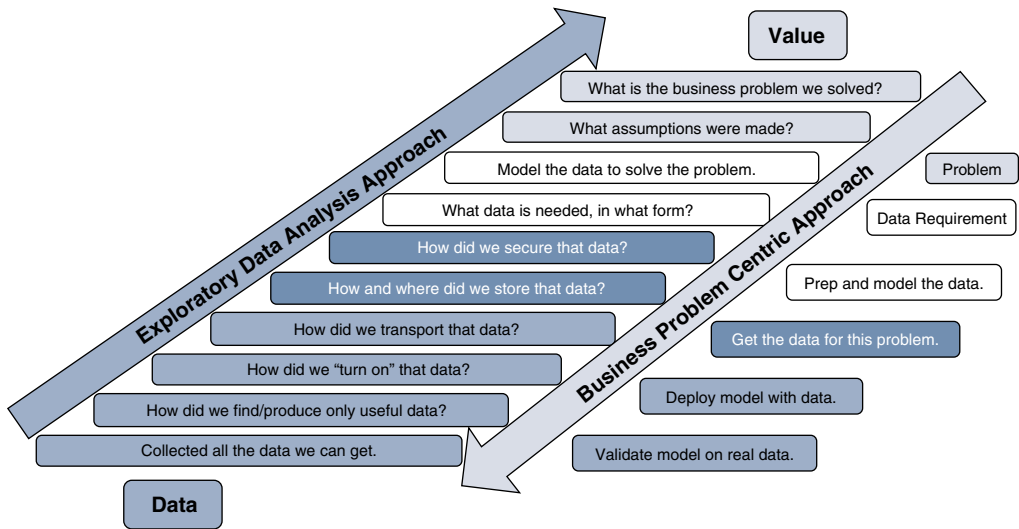


Figure 15-3 *Common Analytics Process*

You know that you can quickly get started by engaging others or engaging yourself in the multiple facets of analytics solutions. You can use the analytics infrastructure model shown in Figure 15-4 to engage with others who come from other areas of the use-case spectrum.

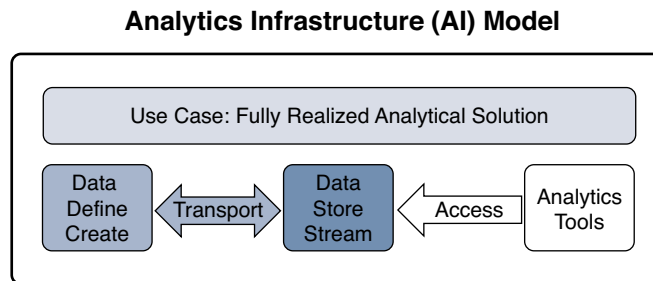


Figure 15-4 *Analytics Infrastructure Model*

All About Networking Data

In Chapter 3, “Understanding Networking Data Sources,” you learned all about planes of operation in networking, and you learned that you can apply this planes concept to other areas in IT, such as cloud, using the simple diagram in Figure 15-5.

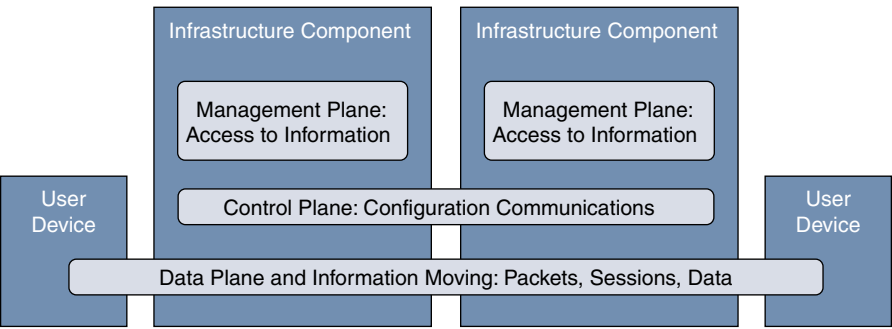


Figure 15-5 *Planes of Operation*

Whether the components you analyze identify these areas as planes or not, the concepts still apply. There is management plane data about components you analyze, control plane data about interactions within the environment, and data plane activity for the function the component is performing.

You also understand the complexities of network and server virtualization and segmentation. You realize that these technologies can result in complex network architectures, as shown in Figure 15-6. You now understand the context of the data you are analyzing from any environment.

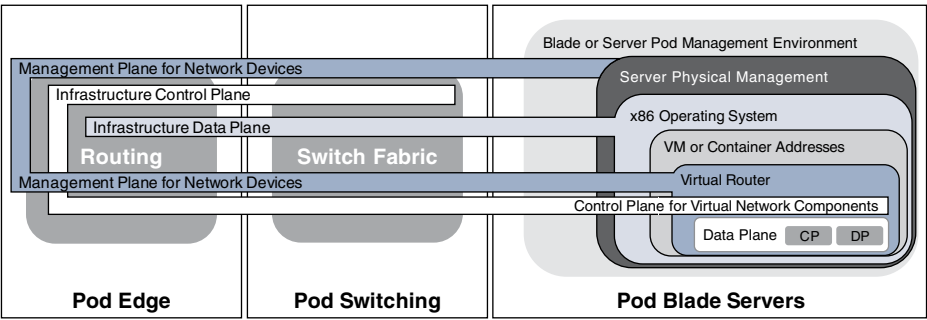


Figure 15-6 *Planes of Operation in a Virtualized Environment*

In Chapter 4, “Accessing Data from Network Components,” you dipped into the details of data. You should now understand the options you have for push and pull of data from networks, including how you get it and how you can represent it in useful ways. As you worked through the use cases, you may have recognized the sources of much of the data

that you worked with, and you should understand ways to get that same data from your own environments. Whether the data is from any plane of operation or any database or source, you now have a way to gather and manipulate it to fit the analytics algorithms you want to try.

Using Bias and Innovation to Discover Solutions

Chapter 5, “Mental Models and Cognitive Bias,” moved you out of the network engineer comfort zone and reviewed the biases that will affect you and the stakeholders for whom you build solutions. The purpose of this chapter was to make you slow down and examine how you think (mental models) and how you think about solutions that you choose to build. If the chapter’s goal was achieved, after you finished the chapter, you immediately started to recognize biases in yourself and others. You need to work with or around these biases as necessary to achieve results for yourself and your company. Understanding these biases will help you in many other areas of your career as well.

With your mind in this open state of paying attention to biases, you should have been ready for Chapter 6, “Innovative Thinking Techniques,” which is all about innovation. Using your ability to pay closer attention from Chapter 5, you were able to examine known techniques for uncovering new and innovative solutions by engaging with industry and others in many ways. Your new attention to detail, combined with these interesting ways to foster ideas, may have already gotten your innovation motor running.

Analytics Use Cases and Algorithms

Chapter 7, “Analytics Use Cases and the Intuition Behind Them,” is meant to give you ideas for using your newfound innovation methods from Chapter 6. This is the longest chapter in the book, and it is filled with use-case concepts from a wide variety of industries. You should have left this chapter with many ideas for use cases that you wanted to build with your analytics solutions. Each time you complete a solution and gain more and more skills and perspectives, you should come back to this chapter and read the use cases again. Your new perspectives will highlight additional areas where you can innovate or give you some guidance to hit the Internet for possibilities. You should save each analysis you build to contribute to a broader solution now or in the future.

Chapter 8, “Analytics Algorithms and the Intuition Behind Them,” provides a broad and general overview of the types of algorithms most commonly used to develop the use cases you wish to carry forward. You learned that there are techniques and algorithms as simple as box plots and as complex as long short-term memory (LSTM) neural networks. You have an understanding of the categories of algorithms that you can research for solving your analytics problems. If you have done any research yet, then you understand that this chapter could have been a book or a series of books. The bells, knobs, buttons, whistles, and widgets that were not covered for each of the algorithms are overwhelming. Chapter 8 is about just knowing where to start your research.

Building Real Analytics Use Cases

In Chapter 9, “Building Analytics Use Cases,” you learned that you would spend more time in your analytics solutions as you move from idea generation to actual execution and solution building, as shown in Figure 15-7.

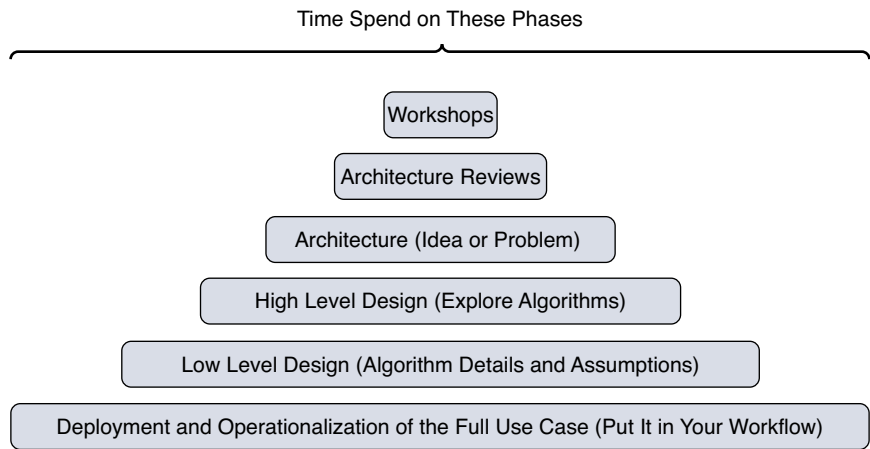


Figure 15-7 *Time Spend on Phases of Analytics Design*

Conceptualizing and getting the high-level flow for your idea can generally be quick, but getting the data, details of the algorithms, and scaling systems up for production use can be very time-consuming. In Chapter 9 you got an introduction to how to set up a Python environment for doing your own data science work in Jupyter Notebooks.

In Chapter 10, “Developing Real Use Cases: The Power of Statistics,” you saw your first use case in the book and learned a bit about how to use Python, Jupyter, statistical methods, and statistical tests. You now understand how to explore data and how to ensure that the data is in the proper form for the algorithms you want to use. You know how to calculate base rates to get the ground truth, and you know how to prepare your data in the proper distributions for use in analytics algorithms. You have gained the statistical skills shown in Figure 15-8.

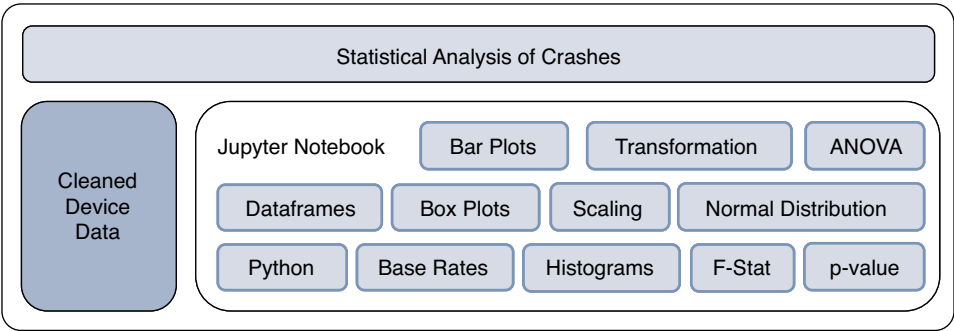


Figure 15-8 *Your Learning from Chapter 10*

In Chapter 11, *Developing Real Use Cases: Network Infrastructure Analytics*, you explored unsupervised machine learning. You also learned how to build a search index for your assets and how to cluster data to provide interesting perspective. You were exposed to encoding methods used to make data fit algorithms. You now understand text and categorical data, and you know how to encode it to build solutions using the techniques shown in Figure 15-9.

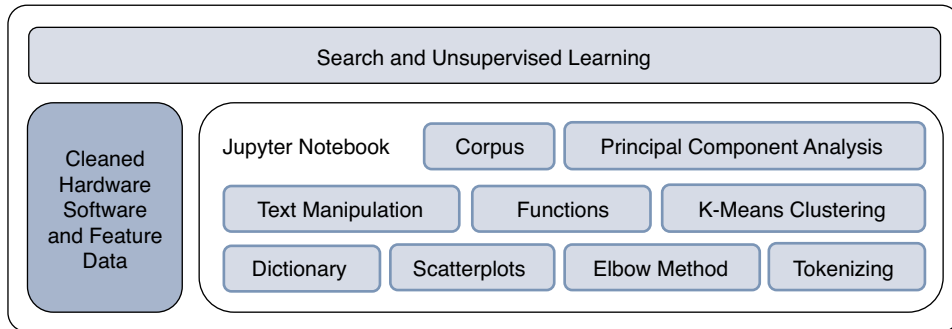


Figure 15-9 *Your Learning from Chapter 11*

In Chapter 12, *Developing Real Use Cases: “Control Plane Analytics Using Syslog Telemetry,”* you learned how to analyze event-based telemetry data. You can easily find most of the same things that you see in many of the common log packages with some simple dataframe manipulations and filters. You learned how to analyze data with Python and how to plot time series into visualizations. You again used encoding to encode logs into dictionaries and vectorized representations that work with the analytics tools available to you. You learned how to use SME evaluation and machine learning together to find actionable insights in large data sets. Finally, you saw the apriori algorithm in action on log messages treated as market baskets. You added to your data science skills with the components shown in Figure 15-10.

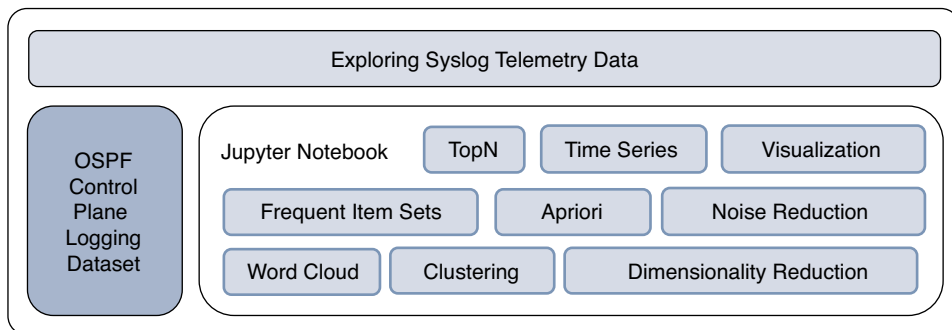


Figure 15-10 *Your Learning from Chapter 12*

In Chapter 13, “Developing Real Use Cases: Data Plane Analytics,” you learned what to do with data plane packet captures in Python. You now know how to load these files from raw packet captures into Jupyter Notebook in pandas dataframes so you can slice and dice them in many ways. You learned another case of combining SME knowledge with some simple math to make your own data by creating new columns of average ports, which you used for unsupervised machine learning clustering. You saw how to use unsupervised learning for cybersecurity investigation on network data plane traffic. You learned how to combine your SME skills with the techniques shown in Figure 15-11.

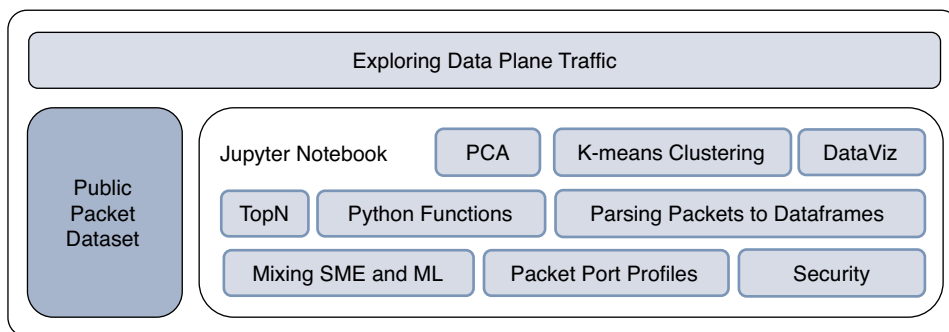


Figure 15-11 *Your Learning from Chapter 13*

Cisco Services and Solutions

In Chapter 14, “Cisco Analytics,” you got an overview of Cisco solutions that will help you bring analytics to your company environment. These solutions can provide data to use as context and input for your own use cases. You saw how Cisco covers many parts of the cloud, IoT, enterprise, and service provider environments with custom analytics services and solutions. You learned how Cisco provides learning for you to build your own (for example, this book) or Cisco training.

In Closing

I hope that you now understand that exploring data and building models is one thing, and building them into productive tools with good workflows is an important next step. You can now get started on the exploration in order to find what you need to build your analytics tools, solutions, and use cases. Getting people to use your tools to support the business is yet another step, and you are now better prepared for that step. You have learned how to identify what is important to your stakeholders so you can build your analytics solutions to solve their business problems. You have learned how to design and build components for your use cases from the ground up. You can manipulate and encode your data to fit available algorithms. You are ready.

This is the end of the book but only the beginning of your analytics journey. Buckle up and enjoy the ride.

Function for Parsing Packets from pcap Files

The following function is for parsing packets from pcap files for Chapter 13:

```
def parse_scapy_packets(packetlist):
    count=0
    datalist=[]
    for packet in packetlist:
        dpack={}
        dpack['id']=str(count)
        dpack['len']=str(len(packet))
        dpack['timestamp']=datetime.datetime.fromtimestamp(packet.time)\
            .strftime('%Y-%m-%d %H:%M:%S.%f')
        if packet.haslayer(Ether):
            dpack.setdefault('esrc', packet[Ether].src)
            dpack.setdefault('edst', packet[Ether].dst)
            dpack.setdefault('etype', str(packet[Ether].type))
        if packet.haslayer(Dot1Q):
            dpack.setdefault('vlan', str(packet[Dot1Q].vlan))
        if packet.haslayer(IP):
            dpack.setdefault('isrc', packet[IP].src)
            dpack.setdefault('idst', packet[IP].dst)
            dpack.setdefault('iproto', str(packet[IP].proto))
            dpack.setdefault('iplen', str(packet[IP].len))
            dpack.setdefault('ipttl', str(packet[IP].ttl))
        if packet.haslayer(TCP):
            dpack.setdefault('tsport', str(packet[TCP].sport))
            dpack.setdefault('tdport', str(packet[TCP].dport))
            dpack.setdefault('twindow', str(packet[TCP].window))
```



```

if packet.haslayer(UDP):
    dpack.setdefault('utsport', str(packet[UDP].sport))
    dpack.setdefault('utdport', str(packet[UDP].dport))
    dpack.setdefault('ulen', str(packet[UDP].len))
if packet.haslayer(ICMP):
    dpack.setdefault('icmptype', str(packet[ICMP].type))
    dpack.setdefault('icmpcode', str(packet[ICMP].code))
if packet.haslayer(IPError):
    dpack.setdefault('iperrorsrc', packet[IPError].src)
    dpack.setdefault('iperrordst', packet[IPError].dst)
    dpack.setdefault('iperrorproto', str(packet[IPError].proto))
if packet.haslayer(UDPError):
    dpack.setdefault('uerrorsrc', str(packet[UDPError].sport))
    dpack.setdefault('uerrordst', str(packet[UDPError].dport))
if packet.haslayer(BOOTP):
    dpack.setdefault('bootpop', str(packet[BOOTP].op))
    dpack.setdefault('bootpciaddr', packet[BOOTP].ciaddr)
    dpack.setdefault('bootpyiaddr', packet[BOOTP].yiaddr)
    dpack.setdefault('bootpsiaddr', packet[BOOTP].siaddr)
    dpack.setdefault('bootpgiaddr', packet[BOOTP].giaddr)
    dpack.setdefault('bootpchaddr', packet[BOOTP].chaddr)
if packet.haslayer(DHCP):
    dpack.setdefault('dhcptions', packet[DHCP].options)
if packet.haslayer(ARP):
    dpack.setdefault('arpop', packet[ARP].op)
    dpack.setdefault('arpsrc', packet[ARP].hwsrc)
    dpack.setdefault('arpdst', packet[ARP].hwdst)
    dpack.setdefault('arppsrc', packet[ARP].psrc)
    dpack.setdefault('arppdst', packet[ARP].pdst)
if packet.haslayer(NTP):
    dpack.setdefault('ntpmode', str(packet[NTP].mode))
if packet.haslayer(DNS):
    dpack.setdefault('dnsopcode', str(packet[DNS].opcode))
if packet.haslayer(SNMP):
    dpack.setdefault('snmpversion', packet[SNMP].version)
    dpack.setdefault('snmpcommunity', packet[SNMP].community)
datalist.append(dpack)
count+=1
return datalist

```

Index

Symbols

& (ampersand), 306
\
~ (tilde), 291–292, 370
2×2 charts, 9–10
5-tuple, 65

A

access, data. *See* data access
ACF (autocorrelation function), 262
ACI (Application Centric Infrastructure), 20, 33, 430–431
active-active load balancing, 186
activity prioritization, 170–173
AdaBoost, 252
Address Resolution Protocol (ARP), 61
addresses
 IP (Internet Protocol)
 packet counts, 395–397
 packet format, 390–391
 MAC, 61, 398
algorithms, 3–4, 217–218, 439
 apriori, 242–243, 381–382
 artificial intelligence, 267
 assumptions of, 218–219

classification
 choosing algorithms for, 248–249
 decision trees, 249–250
 gradient boosting methods, 251–252
 neural networks, 252–258
 random forest, 250–251
 SVMs (*support vector machines*), 258–259
 time series analysis, 259–262
confusion matrix, 267–268
contingency tables, 267–268
cumulative gains and lift, 269–270
data-encoding methods, 232–233
dimensionality reduction, 233–234
feature selection, 230–232
regression analysis, 246–247
simulation, 271
statistical analysis
 ANOVA (*analysis of variance*), 227
 Bayes' theorem, 228–230
 box plots, 221–222
 correlation, 224–225
 longitudinal data, 225–226
 normal distributions, 222–223
 outliers, 223
 probability, 228
 standard deviation, 222–223

- supervised learning, 246
- terminology, 219–221
- text and document analysis, 256–262
 - information retrieval*, 263–264
 - NLP (natural language processing)*, 262–263
 - sentiment analysis*, 266–267
 - topic modeling*, 265–266
- unsupervised learning
 - association rules*, 240–243
 - clustering*, 234–239
 - collaborative filtering*, 244–246
 - defined*, 234
 - sequential pattern mining*, 243–244
- alpha, 261
- Amazon, recommender system for, 191–194
- ambiguity bias, 115–116
- ampersand (&), 306
- analysis of variance. *See* ANOVA (analysis of variance)
- analytics algorithms. *See* algorithms
- analytics experts, 25
- analytics infrastructure model, 22–25, 275–276
 - data and transport, 26–28
 - data engine, 28–30
 - data science, 30–32
 - data streaming example, 30
 - illustrated, 437
 - publisher/subscriber environment, 29
 - roles, 24–25
 - service assurance, 33
 - traditional thinking versus, 22–24
 - use cases
 - algorithms*, 3–4
 - defined*, 18–19
 - development*, 2–3
 - examples of*, 32–33
- analytics maturity, 7–8
- analytics models, building, 2, 14–15, 19–20. *See also* use cases
- analytics infrastructure model, 22–25, 275–276, 437
 - data and transport*, 26–28
 - data engine*, 28–30
 - data science*, 30–32
 - data streaming example*, 30
 - publisher/subscriber environment*, 29
 - roles*, 24–25
 - service assurance*, 33
 - traditional thinking versus*, 22–24
- deployment, 2, 14–15, 17–18
- EDA (exploratory data analysis)
 - defined*, 15–16
 - use cases versus solutions*, 18–19
 - walkthrough*, 17–18
- feature engineering, 219
- feature selection, 219
- interpretation, 220
- overfitting, 219
- overlay, 20–22
- problem-centric approach
 - defined*, 15–16
 - use cases versus solutions*, 18–19
 - walkthrough*, 17–18
- underlay, 20–22
- validation, 219
- analytics process, 437
- analytics scales, 436
- analytics solutions, defined, 150
- anchoring effect, 107–109
- AND operator, 306
- ANNs (artificial neural networks), 254–255
- anomaly detection, 153–155
 - clustering, 239
 - statistical, 318–320
- ANOVA (analysis of variance), 227, 305–310
 - data filtering, 305–306
 - describe function, 308
 - drop command, 309
 - groupby command, 307

- homogeneity of variance, 313–318
- Levene's test, 313
- outliers, dropping, 307–310
- pairwise, 317
- Apache Kafka, 28–29
- API (application programming interface)
 - calls, 29
- App iQ platform, 430
- AppDynamics, 6, 428–430
- Application Centric Infrastructure (ACI), 20, 33, 430–431
- application programming interface (API)
 - calls, 29
- application-specific integrated circuits (ASICs), 67
- apply method, 295–296, 346
- approaches. *See* methodology and approach
- apriori algorithms, 242–243, 381–382
- architecture
 - architecture and advisory services, 426–427
 - big data, 4–5
 - microservices, 5–6
- Ariely, Dan, 108
- ARIMA (autoregressive integrated moving average), 101–102, 262
- ARP (Address Resolution Protocol), 61
- artificial general intelligence, 267
- artificial intelligence, 11, 267
- artificial neural networks (ANNs), 254–255
- ASICs (application-specific integrated circuits), 67
- assets
 - data plane analytics use case, 422–423
 - tracking, 173–175
- association rules, 240–243
- associative thinking, 131–132
- authority bias, 113–114
- autocorrelation function (ACF), 262
- automation, 11, 33, 431–432
- autonomous applications, use cases for, 200–201

- autoregressive integrated moving average (ARIMA), 101–102, 262
- autoregressive process, 262
- availability bias, 111
- availability cascade, 112, 141
- averages
 - ARIMA (autoregressive integrated moving average), 262
 - moving averages, 262
- Azure Cloud Network Watcher, 68

B

- BA (business analytics) dashboards, 13, 42
- back-propagation, 254
- backslash (\), 288
- bagging, 250–251
- bar charts, platform crashes example, 289–290
- base-rate neglect, 117
- Bayes' theorem, 228–230
- Bayesian methods, 230
- BCI (Business Critical Insights), 335, 425
- behavior analytics, 175–178
- benchmarking use cases, 155–157
- BGP (Border Gateway Protocol), 41, 61
- BI (business intelligence) dashboards, 13, 42
- bias, 2–3, 439
 - ambiguity, 115–116
 - anchoring effect, 107–109
 - authority, 113–114
 - availability, 111
 - availability cascade, 112
 - base-rate neglect, 117
 - clustering, 112
 - concept of, 104–105
 - confirmation, 114–115
 - context, 116–117
 - correlation, 112
 - “curse of knowledge”, 119
 - Dunning-Kruger effect, 120–121

- empathy gap, 123
- endowment effect, 121
- expectation, 114–115
- experimenter's, 116
- focalism, 107
- framing effect, 109–110, 151
- frequency illusion, 117
- group, 120
- group attribution error, 118
- halo effect, 123–124
- hindsight, 9, 123–124
- HIPPO (highest paid persons' opinion)
 - impact, 113–114
- IKEA effect, 121–122
- illusion of truth effect, 112–113
- impact of, 105–106
- imprinting, 107
- innovation and, 128
- “law of small numbers”, 117–118
- mirroring, 110–111
- narrative fallacy, 107–108
- not-invented-here syndrome, 122
- outcome, 124
- priming effect, 109, 151
- pro-innovation, 121
- recency, 111
- solutions and, 106–107
- status-quo, 122
- sunk cost fallacy, 122
- survivorship, 118–119
- table of, 124–126
- thrashing, 122
- tunnel vision, 107
- WYSIATI (What You See Is All There Is), 118
- zero price effect, 123

Bias, Randy, 204

big data, 4–5

Border Gateway Protocol (BGP), 41, 61

box plots, 221–222

- platform crashes example, 297–299
- software crashes example, 300–305

Box-Jenkins method, 262

breaking anchors, 140

Breusch-Pagan tests, 220

budget analysis, 169

bug analysis use cases, 178–179

business analytics (BA) dashboards, 13, 42

Business Critical Insights (BCI), 335, 425

business domain experts, 25

business intelligence (BI) dashboards, 13, 42

business model

- analysis, 200–201
- optimization, 201–202

C

capacity planning, 180–181

CARESS technique, 137

cat /etc/*release command, 61

categorical data, 77–78

causation, correlation versus, 112

CDP (Cisco Discovery Protocol), 60, 93

charts

- cumulative gains, 269–270
- lift, 269–270
- platform crashes use case, 289–290

churn use cases, 202–204

Cisco analytics solutions, 6, 425–426, 442

- analytics platforms and partnerships, 433
- AppDynamics, 428–430
- architecture and advisory services, 426–427
- BCI (Business Critical Insights), 335, 425
- CMS (Cisco Managed Services), 425
- Crosswork automation, 431–432
- DNA (Digital Network Architecture), 428
- IoT (Internet of Things) analytics, 432
- open source platform, 433–434
- Stealthwatch, 427
- Tetration, 430–431

- Cisco Application Centric Infrastructure (ACI), 20
- Cisco Discovery Protocol (CDP), 60
- Cisco Identity Service Engine (ISE), 427
- Cisco IMC (Integrated Management Controller), 40–41
- Cisco iWAN+Viptela, 20
- Cisco TrustSec, 427
- Cisco Unified Computing System (UCS), 62
- citizen data scientists, 11
- classification, 157–158
 - algorithms
 - choosing*, 248–249
 - decision trees*, 249–250
 - gradient boosting methods*, 251–252
 - neural networks*, 252–258
 - random forest*, 250–251
 - SVMs (support vector machines), 258–259
 - time series analysis, 259–262
- cleansing data, 29, 86
- CLI (command-line interface) scraping, 59, 92
- cloud software, 5–6
- Cloudera, 433
- clustering, 234–239
 - K-means, 344–349, 373–375
 - machine learning-guided troubleshooting, 350–353
 - SME port clustering, 407–413
 - cluster scatterplot*, 410–411
 - host patterns*, 411–413
 - K-means clustering*, 408–410
 - port profiles*, 407–408
 - use cases, 158–160
- clustering bias, 112
- CMS (Cisco Managed Services), 425
- CNNs (convolutional neural networks), 254–255
- cognitive bias. *See* bias
- Cognitive Reflection Test (CRT), 98
- cognitive trickery, 143
- cohorts, 160
- collaborative filtering, 244–246
- collinearity, 225
- columns
 - dropping, 287
 - grouping, 307
- columns command, 286
- Colvin, Geoff, 103
- command-line interface (CLI) scraping, 59, 92
- commands. *See also* functions
 - `cat /etc/*release`, 61
 - columns, 286
 - drop, 309
 - groupby, 307, 346, 380, 398
 - head, 396, 404
 - join, 291
 - tcpdump, 68
- comma-separated values (CSV) files, 82
- communication, control plane, 38
- Competing on Analytics* (Davenport and Harris), 148
- compliance to benchmark, 155
- computer thrashing, 140
- condition-based maintenance, 189
- confirmation bias, 114–115
- confusion matrix, 267–268
- container on box, 74–75
- context
 - context bias, 116–117
 - context-sensitive stop words, 329
 - external data for, 89
- contingency tables, 267–268
- continuous numbers, 78–79
- control plane, 441
 - activities in, 41
 - communication, 38
 - data examples, 46–47, 67–68
 - defined, 37
 - syslog telemetry use case, 355
 - data encoding*, 371–373
 - data preparation*, 356–357, 369–371

- high-volume producers, identifying*, 362–366
- K-means clustering*, 373–375
- log analysis with pandas*, 357–360
- machine learning-based evaluation*, 366–367
- noise reduction*, 360–362
- OSP (Open Shortest Path First) routing, 357
- syslog severities*, 359–360
- task list*, 386–387
- transaction analysis*, 379–386
- word cloud visualization*, 367–369, 375–379
- convolutional neural networks (CNNs), 254–255
- correlation
 - correlation bias, 112
 - explained, 224–225
 - use cases, 160–162
- cosine distance, 236
- count-encoded matrix, 336–338
- CountVectorizer method, 338
- covariance, 167
- Covey, Stephen, 10
- crashes, device. *See* device crash use cases
- crashes, network. *See* network infrastructure analytics use case
- CRISP-DM (cross-industry standard process for data mining), 18
- critical path, 172, 211
- CRM (customer relationship management) systems, 25, 187
- cross-industry standard process for data mining (CRISP-DM), 18
- Crosswork Network Automation, 33, 431–432
- crowdsourcing, 133–134
- CRT (Cognitive Reflection Test), 98
- CSV (comma-separated value) files, 82
- cumulative gains, 269–270
- curse of dimensionality, 159
- “curse of knowledge”, 119
- custom labels, 93

- customer relationship management (CRM) systems, 25, 187
- customer segmentation, 160

D

- data. *See also* data access
 - domain experts, 25
 - encoding, 232–233
 - network infrastructure analytics use case*, 328–336
 - syslog telemetry use case*, 371–373
 - engine, 28–30
 - gravity, 76
 - loading
 - data plane analytics use case*, 390–394
 - network infrastructure analytics use case*, 325–328
 - statistics use cases*, 286–288
 - mining, 150
 - munging, 85
 - network, 35–37
 - business and applications data relative to*, 42–44
 - control plane*, 37, 38, 41, 46–47
 - data plane*, 37, 41, 47–49
 - management plane*, 37, 40–41, 44–46
 - network virtualization*, 49–51
 - OpenStack nodes*, 39–40
 - planes, combining across virtual and physical environments*, 51–52
 - sample network*, 38
 - normalization, 85
 - preparation, 29, 86
 - encoding methods*, 85
 - KPIs (key performance indicators)*, 86–87
 - made-up data*, 84–85
 - missing data*, 86
 - standardized data*, 85
 - syslog telemetry use case*, 355, 369–371, 379

- reconciliation, 29
- regularization, 85
- scaling, 298
- standardizing, 85
- storage, 6
- streaming, 30
- structure, 82
 - JSON (JavaScript Object Notation)*, 82–83
 - semi-structured data*, 84
 - structured data*, 82
 - unstructured data*, 83–84
- transformation, 310
- transport, 89–90
 - CLI (command-line interface) scraping*, 92
 - HLD (high-level design)*, 90
 - IPFIX (IP Flow Information Export)*, 95
 - LLD (low-level design)*, 90
 - NetFlow*, 94
 - other data*, 93
 - sFlow*, 95
 - SNMP (Simple Network Management Protocol)*, 90–92
 - SNMP (Simple Network Management Protocol) traps*, 93
 - Syslog*, 93–94
 - telemetry*, 94
- types, 76–77
 - continuous numbers*, 78–79
 - discrete numbers*, 79
 - higher-order numbers*, 81–82
 - interval scales*, 80
 - nominal data*, 77–78
 - ordinal data*, 79–80
 - ratios*, 80–81
- warehouses, 29
- data access. *See also* data structure; transport of data; types**
 - container on box, 74–75
 - control plane data, 67–68
 - data plane traffic capture, 68–69
 - ERSPAN (Encapsulated Remote Switched Port Analyzer)*, 69
 - inline security appliances*, 69
 - port mirroring*, 69
 - RSPAN (Remote SPAN)*, 69
 - SPAN (Switched Port Analyzer)*, 69
 - virtual switch operations*, 69–70
- DPI (deep packet inspection), 56
- external data for context, 89
- IoT (Internet of Things) model, 75–76
- methods of, 55–57
- observation effect, 88
- packet data, 70–74
 - HTTP (Hypertext Transfer Protocol)*, 71–72
 - IPsec (Internet Protocol Security)*, 73–74
 - IPv4*, 70–71
 - SSL (Secure Sockets Layer)*, 74
 - TCP (Transmission Control Protocol)*, 71–72
 - VXLAN (Virtual Extensible LAN)*, 74
- panel data, 88
- pull data availability
 - CLI (command-line interface) scraping*, 59, 92
 - NETCONF (Network Configuration Protocol)*, 60
 - SNMP (Simple Network Management Protocol)*, 57–59
 - unconventional data sources*, 60–61
 - YANG (Yet Another Next Generation)*, 60
- push data availability
 - IPFIX (IP Flow Information Export)*, 64–67
 - NetFlow*, 65–66
 - sFlow*, 67, 95
 - SNMP (Simple Network Management Protocol) traps*, 61–62, 93
 - Syslog*, 62–63, 93–94
 - telemetry*, 63–64
- timestamps, 87–88

- data lake, 29
- data pipeline engineering, 90
- data plane. *See also* data plane analytics use case
 - activities in, 41
 - data examples, 47–49
 - defined, 37
 - traffic capture, 68–69
 - ERSPAN (Encapsulated Remote Switched Port Analyzer)*, 69
 - inline security appliances*, 69
 - port mirroring*, 69
 - RSPAN (Remote SPAN)*, 69
 - SPAN (Switched Port Analyzer)*, 69
 - virtual switch operations*, 69–70
- data plane analytics use case, 389, 442
 - assets, 422–423
 - data loading and exploration, 390–394
 - IP package format*, 390–391
 - packet file loading*, 390
 - parsed fields*, 392–393
 - Python packages, importing*, 390
 - TCP package format*, 391
 - full port profiles, 413–419
 - investigation task list, 423–424
 - SME analysis
 - dataframe and visualization library loading*, 394
 - host analysis*, 399–404
 - IP address packet counts*, 395–397
 - IP packet protocols*, 398
 - MAC addresses*, 398
 - output*, 404–406
 - time series counts*, 395
 - timestamps and time index*, 394–395
 - topology mapping information*, 398
 - SME port clustering, 407–413
 - cluster scatterplot*, 410–411
 - host patterns*, 411–413
 - K-means clustering*, 408–410
 - port profiles*, 407–408
 - source port profiles, 419–422
- data science, 25, 30–32, 278–280
- data structure, 82
- databases, 6
- dataframes
 - combining, 292–293
 - defined, 286–287
 - dropping columns from, 287
 - filtering, 287, 290–292, 300, 330, 370
 - grouping, 293–296, 299–300, 307
 - loading, 394
 - outlier analysis, 318–320
 - PCA (principal component analysis), 339–340, 372–373
 - sorting without, 326–327
 - `value_counts` function, 288–290
 - views, 329–330, 347
- data-producing sensors, 210–211
- Davenport, Thomas, 148
- de Bono, Edward, 132
- decision trees
 - example of, 249–250
 - random forest, 250–251
- deep packet inspection (DPI), 56
- defocusing, 140
- deliberate practice, 100, 102
- delivery models, use cases for, 210–212
- delta, 262
- dependence, 261
- deployment of models, 2, 14–15, 17–18
- describe function, 308
- descriptive analytics, 8–9
- descriptive analytics use cases, 167–168
- designing solutions. *See* solution design
- destination IP address packet counts, 396–397
- deviation, standard, 222–223
- device crash use cases, 285
 - anomaly detection, 318–320
 - ANOVA (analysis of variance), 305–310
 - data filtering*, 305–306
 - describe function*, 308

- drop command*, 309
- groupby command*, 307
- homogeneity of variance*, 313–318
- outliers, dropping*, 307–310
- pairwise*, 317
- data loading and exploration, 286–288
- data transformation, 310
- normality, tests for, 311–313
- platform crashes, 288–299
 - apply method*, 295–296
 - box plot*, 297–298
 - crash counts by product ID*, 294–295
 - crash counts/rate comparison plot*, 298–299
 - crash rates by product ID*, 296–298
 - crashes by platform*, 292–294
 - data scaling*, 298
 - dataframe filtering*, 290–292
 - groupby object*, 293–296
 - horizontal bar chart*, 289–290
 - lambda function*, 296
 - overall crash rates*, 292
 - router reset reasons*, 290
 - simple bar chart*, 289
 - value_counts function*, 288–289
- software crashes, 299–305
 - box plots*, 300–305
 - dataframe filtering*, 300
 - dataframe grouping*, 299–300
- diagnostic targeting, 209
- “dial-in” telemetry configuration, 64
- “dial-out” telemetry configuration, 64
- dictionaries, tokenization and, 328
- diffs function, 352
- Digital Network Architecture (DNA), 33, 428
- dimensionality
 - curse of, 159
 - reduction, 233–234, 337–340
- discrete numbers, 79
- distance methods, 236
- divisive clustering, 236
- DNA (Digital Network Architecture), 33, 428
- DNA mapping, 324–325
- DNAC (DNA Center), 428
- doc2bow, 331–332
- document analysis, 256–262
 - information retrieval, 263–264
 - NLP (natural language processing), 262–263
 - sentiment analysis, 266–267
 - topic modeling, 265–266
- DPI (deep packet inspection), 56
- drop command, 309
- dropouts, 204–206
- dropping columns, 287
- Duhigg, Charles, 99
- dummy variables, 232
- Dunning-Kruger effect, 120–121

E

- EDA (exploratory data analysis)
 - defined, 15–16
 - use cases versus solutions, 18–19
 - walkthrough, 17–18
- edit distance, 236
- EDT (event-driven telemetry), 64
- EIGRP (Enhanced Interior Gateway Routing Protocol), 61, 398
- ElasticNet regression, 247
- electronic health records, 210
- empathy gap, 123
- Encapsulated Remote Switched Port Analyzer (ERSPAN), 69
- encoding methods, 85, 232–233
 - network infrastructure analytics use case, 328–336
 - syslog telemetry use case, 371–373
- Encrypted Traffic Analytics (ETA), 427
- endowment effect, 121
- engagement models, 206–207

engine, analytics infrastructure model, 28–30
 Enhanced Interior Gateway Routing Protocol (EIGRP), 61, 398
 entropy, 250
 environment setup, 282–284, 325–328
 episode mining, 244
 errors, group attribution, 118. *See also* bias
 ERSPAN (Encapsulated Remote Switched Port Analyzer), 69
 ETA (Encrypted Traffic Analytics), 427
 ETL (Extract, Transform, Load), 26
 ETSI (European Telecommunications Standards Institute), 75
 Euclidean distance, 236
 European Telecommunications Standards Institute (ETSI), 75
 event log analysis use cases, 181–183
 event-driven telemetry (EDT), 64
 expectation bias, 114–115
 experimentation, 141–142
 experimenter's bias, 116
 expert systems deployment, 214
 exploratory data analysis. *See* EDA (exploratory data analysis)
 exponential smoothing techniques, 261
 external data for context, 89
 Extract, Transform, Load (ETL), 26

F

F statistic, 220
 failure analysis use cases, 183–185
 fast path, 211
 features
 defined, 42–43
 feature engineering, 219
 selection, 219, 230–232
 Few, Stephen, 163
 fields, data plane analytics use case, 392–393
 files, CSV (comma-separated value), 82. *See also* logs
 fillna, 342–343

filtering
 ANOVA and, 305–306
 collaborative, 244–246
 dataframes, 287, 290–292, 300, 330, 370
 platform crashes example, 290–292
 software crashes example, 300
 fingerprinting, 324–325
 “Five whys” technique, 137–138
 Flexible NetFlow, 65
 Flight 1549, 99–100
 focalism, 107
 fog computing, 76
 foresight, 9
 FP growth algorithms, 242
 framing effect, 109–110, 151
 Franks, Bill, 147
 fraud detection use cases, 207–209
 Frederick, Shane, 98
 FreeSpan, 244
 frequency illusion, 117
 F-tests, 227, 314
 full host profiles, 401–403
 full port profiles, 413–419
 functions
 apply, 295–296, 346
 apriori, 242–243, 381–382
 CountVectorizer, 338
 describe, 308
 diffs, 352
 host_profile, 403
 join, 370
 lambda, 296
 max, 347
 reset_index, 414
 split, 368
 value_counts, 288–289, 396, 400, 403

G

gains, cumulative, 269–270
 gamma, 261
 Gartner analytics, 8
 gender bias, 97–98

generalized sequential pattern (GSP), 244
 Gensim package, 264, 283, 328, 331–332
 Gladwell, Malcolm, 99
 Global Positioning System (GPS), 210–211
 Goertzel, Ben, 267
 GPS (Global Positioning System), 210–211
 gradient boosting methods, 251–252
 gravity, data, 76
 group attribution error, 118
 group bias, 120
 group-based strong learners, 250
 groupby command, 307, 346, 380, 398
 groupby object, 293–296
 grouping
 columns, 307
 dataframes, 293–296, 299–300
 GSP (generalized sequential pattern), 244

H

Hadoop, 28–29
 halo effect, 123–124
 hands-on experience, mental models and, 100
 hard data, 150
 Harris, Jeanne, 148
 head command, 396, 404
Head Game (Mudd), 110
 healthcare use cases, 209–210
 Hewlett-Packard iLO (Integrated Lights Out), 40–41
 hierarchical agglomerative clustering, 236–237
 higher-order numbers, 81–82
 highest paid persons' opinion (HIPPO) impact, 113–114
 high-level design (HLD), 90
 high-volume producers, identifying, 362–366
 hindsight bias, 9, 123–124
 HIPPO (highest paid persons' opinion) impact, 113–114

HLD (high-level design), 90
 homogeneity of variance, 313–318
 homoscedasticity, 313–318
 Hortonworks, 433
 host analysis, 399–404
 data plane analytics use case, 411–413
 full host profile analysis, 401–403
 per-host analysis function, 399
 per-host conversion analysis, 400–401
 per-host port analysis, 403
 host_profile function, 403
How Not to Be Wrong (Ellenberg), 118–119
 HTTP (Hypertext Transfer Protocol), 71–72
 human bias, 97–98
 Hypertext Transfer Protocol (HTTP), 71–72
 Hyper-V, 70

I

IBM, Cisco's partnership with, 433
 IBN (intent-based networking), 11, 428
 ICMP (Internet Control Message Protocol), 398
 ID3 algorithm, 250
 Identity Service Engine (ISE), 427
 IETF (Internet Engineering Task Force), 66–67, 95
 IGMP (Internet Group Management Protocol), 398
 IGP (interior gateway protocols), 357
 IIA (International Institute for Analytics), 147
 IKEA effect, 121–122
 illusion of truth effect, 112–113
 iLO (Integrated Lights Out), 40–41
 image recognition use cases, 170
 IMC (Integrated Management Controller), 40–41
 importing Python packages, 390
 imprinting, 107
 industry terminology, 7

inference, statistical, 228

influence, 227

information retrieval

algorithms, 263–264

use cases, 185–186

Information Technology Infrastructure Library (ITIL), 161

infrastructure analytics use case, 323–324

data encoding, 328–336

data loading, 325–328

data visualization, 340–344

dimensionality reduction, 337–340

DNA mapping and fingerprinting, 324–325

environment setup, 325–328

K-means clustering, 344–349

machine learning-guided troubleshooting, 350–353

search challenges and solutions, 331–336

in-group bias, 120

inline security appliances, 69

innovative thinking techniques, 127–128, 439

associative thinking, 131–132

bias and, 128

breaking anchors, 140

cognitive trickery, 143

crowdsourcing, 133–134

defocusing, 140

experimentation, 141–142

inverse thinking, 139–140, 204–206

lean thinking, 142

metaphoric thinking, 130–131

mindfulness, 128

networking, 133–135

observation, 138–139

perspectives, 130–131

questioning

CARESS technique, 137

example of, 135–137

“Five whys”, 137–138

quick innovation wins, 143–144

six hats thinking approach, 132–133

unpriming, 140

The Innovator’s DNA (Dyer et al), 128

insight, 9

installing Jupyter Notebook, 282–283

Integrated Lights Out (iLO), 40–41

Integrated Management Controller (IMC), 40–41

Intelligent Wide Area Networks (iWAN), 20, 428

intent-based networking (IBN), 11, 428

interior gateway protocols (IGPs), 357

International Institute for Analytics (IIA), 147

Internet clickstream analysis, 169

Internet Control Message Protocol (ICMP), 398

Internet Engineering Task Force (IETF), 66–67, 95

Internet Group Management Protocol (IGMP), 398

Internet of Things (IoT), 75–76

analytics, 432

growth of, 214

Internet of Things—From Hype to Reality (Rayes and Salam), 75

Internet Protocol (IP)

IP address packet counts, 395–397

packet format, 390–391

packet protocols, 398

Internet Protocol Security (IPsec), 73–74

interval scales, 80

intrusion detection use cases, 207–209

intuition

explained, 103–104

System 1/System 2, 102–103

inventory management, 169

inverse problem, 206

inverse thinking, 139–140, 204–206

IoT. See Internet of Things (IoT)

IP (Internet Protocol)

IPFIX (IP Flow Information Export),
64–67, 95
packet counts, 395–397
packet data, 70–71
packet format, 390–391
packet protocols, 398
IPFIX (IP Flow Information Export),
64–67, 95
IPsec (Internet Protocol Security),
73–74
ISE (Identity Service Engine), 427
isin keyword, 366
IT analytics use cases, 170
activity prioritization, 170–173
asset tracking, 173–175
behavior analytics, 175–178
bug and software defect analysis,
178–179
capacity planning, 180–181
event log analysis, 181–183
failure analysis, 183–185
information retrieval, 185–186
optimization, 186–188
prediction of trends, 190–194
predictive maintenance, 188–189
scheduling, 194–195
service assurance, 195–197
transaction analysis, 197–199
ITIL (Information Technology
Infrastructure Library), 161
iWAN (Intelligent Wide Area Networks),
20, 428

J

Jaccard distance, 236
Jasper, 432
JavaScript Object Notation (JSON),
82–83
join command, 291
join function, 370
JSON (JavaScript Object Notation), 82–83
Jupyter Notebook, installing, 282–283

K

Kafka (Apache), 28–29
Kahneman, Daniel, 102–103
kcluster values, 347. *See also* K-means
clustering
Kendall's tau, 225, 236
Kenetic, 430–433
key performance indicators (KPIs),
86–87
keys, 82–83
key/value pairs, 82–83
keywords, isin, 366
Kinetic, 430–433
K-means clustering
data plane analytics use case, 408–410
network infrastructure analytics use case,
344–349
syslog telemetry use case, 373–375
knowledge
curse of, 119
management of, 8
known attack vectors, 214
KPIs (key performance indicators), 86–87
Kurzweil, Ray, 267

L

labels, 151
ladder of powers methods, 310
lag, 262
lambda function, 296
language
selection, 6
translation, 11
lasso regression, 247
latent Dirichlet allocation (LDA), 265,
334–335
latent semantic indexing (LSI), 265–266,
334–335
law of parsimony, 120, 152
“law of small numbers”, 117–118

LDA (latent Dirichlet allocation), 265, 334–335
The Lean Startup (Ries), 142
 lean thinking, 142
 learning reinforcement, 212–213
 left skewed distribution, 310
 lemmatization, 263
 Levene's test, 313
 leverage, 227
 lift charts, 269–270
 lift-and-gain analysis, 194
 LightGBM, 252
 linear regression, 246–247
 Link Layer Discovery Protocol (LLDP), 61
 Linux servers, pull data availability, 61
 LLD (low-level design), 90
 LLDP (Link Layer Discovery Protocol), 61, 93
 load balancing, active-active, 186
 loading data
 data plane analytics use case, 390–394
 dataframes, 394
 IP package format, 390–391
 packet file loading, 390
 parsed fields, 392–393
 Python packages, importing, 390
 TCP package format, 391
 network infrastructure analytics use case, 325–328
 statistics use cases, 286–288
 logical AND, 306
 logistic regression, 101–102, 247
 logistics use cases, 210–212
 logs
 event log analysis, 181–183
 syslog telemetry use case, 355
 data encoding, 371–373
 data preparation, 356–357, 369–371
 high-volume producers, identifying, 362–366
 K-means clustering, 373–375

log analysis with pandas, 357–360
machine learning-based evaluation, 366–367
noise reduction, 360–362
OSPF (Open Shortest Path First) routing, 357
syslog severities, 359–360
task list, 386–387
transaction analysis, 379–386
word cloud visualization, 367–369, 375–379

Long Short Term Memory (LSTM)
 networks, 254–258

longitudinal data, 225–226

low-level design (LLD), 90

LSI (latent semantic indexing), 265–266, 334–335

LSTM (Long Short Term Memory)
 networks, 254–258

M

M2M initiatives, 75

MAC addresses, 61, 398

machine learning

 classification algorithms

choosing, 248–249

decision trees, 249–250

gradient boosting methods, 251–252

neural networks, 252–258

random forest, 250–251

 defined, 150

 machine learning-based log evaluation, 366–367

 supervised, 151, 246

 troubleshooting with, 350–353

 unsupervised

association rules, 240–243

clustering, 234–239

collaborative filtering, 244–246

defined, 151, 234

sequential pattern mining, 243–244

- use cases, 153
 - anomalies and outliers*, 153–155
 - benchmarking*, 155–157
 - classification*, 157–158
 - clustering*, 158–160
 - correlation*, 160–162
 - data visualization*, 163–165
 - descriptive analytics*, 167–168
 - NLP (natural language processing)*, 165–166
 - time series analysis*, 168–169
 - voice, video, and image recognition*, 170
- making your own data**, 84–85
- Management Information Bases (MIBs)**, 57
- management plane**
 - activities in, 40–41
 - data examples, 44–46
 - defined, 37
- Manhattan distance**, 236
- manipulating data**
 - encoding methods, 85
 - KPIs (key performance indicators), 86–87
 - made-up data, 84–85
 - missing data, 86
 - standardized data, 85
- manufacturer's suggested retail price (MSRP)**, 108
- mapping, DNA**, 324–325
- market basket analysis**, 199
- Markov Chain Monte Carlo (MCMC) systems**, 271
- matplotlib package**, 283
- maturity levels**, 7–8
- max method**, 347
- MBIs (Management Information Bases)**, 57
- MCMC (Markov Chain Monte Carlo) systems**, 271
- MDT (model-driven telemetry)**, 64
- mean squared error (MSE)**, 227
- memory, muscle**, 102
- mental models**
 - bias
 - ambiguity*, 115–116
 - anchoring effect*, 107–109
 - authority*, 113–114
 - availability*, 111, 112
 - base-rate neglect*, 117
 - clustering*, 112
 - concept of*, 104–105
 - confirmation*, 114–115
 - context*, 116–117
 - correlation*, 112
 - “curse of knowledge”*, 119
 - Dunning-Kruger effect*, 120–121
 - empathy gap*, 123
 - endowment effect*, 121
 - expectation*, 114–115
 - experimenter's*, 116
 - focalism*, 107
 - framing effect*, 109–110, 151
 - frequency illusion*, 117
 - group*, 120
 - group attribution error*, 118
 - halo effect*, 123–124
 - hindsight*, 9, 123–124
 - HIPPO (highest paid persons' opinion) impact*, 113–114
 - IKEA effect*, 121–122
 - illusion of truth effect*, 112–113
 - impact of*, 105–106
 - imprinting*, 107
 - “law of small numbers”*, 117–118
 - mirroring*, 110–111
 - narrative fallacy*, 107–108
 - not-invented-here syndrome*, 122
 - outcome*, 124
 - priming effect*, 109, 151
 - pro-innovation*, 121
 - recency*, 111
 - solutions and*, 106–107
 - status-quo*, 122
 - sunk cost fallacy*, 122

- survivorship*, 118–119
- table of*, 124–126
- thrashing*, 122
- tunnel vision*, 107
- WYSIATI (What You See Is All There Is)*, 118
- zero price effect*, 123
- changing how you think, 98–99
- concept of, 97–98, 99–102
- CRT (Cognitive Reflection Test), 98
- human bias, 97–98
- intuition, 103–104
- System 1/System 2, 102–103
- metaphoric thinking**, 130–131
- meters, smart**, 189
- methodology and approach**, 13–14
 - analytics infrastructure model, 22–25.
 - See also* use cases
 - data and transport*, 26–28
 - data engine*, 28–30
 - data science*, 30–32
 - data streaming example*, 30
 - publisher/subscriber environment*, 29
 - roles*, 24–25
 - service assurance*, 33
 - traditional thinking versus*, 22–24
- BI/BA dashboards, 13
- CRISP-DM (cross-industry standard process for data mining), 18
- EDA (exploratory data analysis)
 - defined*, 15–16
 - use cases versus solutions*, 18–19
 - walkthrough*, 17–18
- overlay/underlay, 20–22
- problem-centric approach
 - defined*, 15–16
 - use cases versus solutions*, 18–19
 - walkthrough*, 17–18
- SEMMA (Sample Explore, Modify, Model, and Assess), 18
- microservices architectures**, 5–6
- Migration Analytics**, 425
- mindfulness**, 128–129
- mindset**. *See* mental models
- mirror-image bias**, 110–111
- mirroring**, 69, 110–111
- missing data**, 86
- mlexend package**, 283
- model-driven telemetry (MDT)**, 64
- models**. *See* analytics models, building; mental models
- Monte Carlo simulation**, 202, 271
- moving averages**, 262
- MSE (mean squared error)**, 227
- MSRP (manufacturer's suggested retail price)**, 108
- Mudd, Philip**, 110
- multicollinearity**, 225
- muscle memory**, 102–103

N

- narrative fallacy**, 107–108
- natural language processing (NLP)**, 165–166, 262–263
- negative correlation**, 224
- NETCONF (Network Configuration Protocol)**, 60
- Netflix recommender system**, 191–194
- NetFlow**
 - architecture of, 65
 - capabilities of, 65–66
 - data transport, 94
 - versions of, 65
- Network Configuration Protocol (NETCONF)**, 60
- network functions virtualization (NFV)**, 5–6, 51–52, 365
- network infrastructure analytics use case**, 323–324, 441
 - data encoding, 328–336
 - data loading, 325–328
 - data visualization, 340–344
 - dimensionality reduction, 337–340

- DNA mapping and fingerprinting, 324–325
- environment setup, 325–328
- K-means clustering, 344–349
- machine learning-guided troubleshooting, 350–353
- search challenges and solutions, 331–336
- Network Time Protocol (NTP)**, 87–88
- Network Watcher**, 68
- networking, social**, 133–135
- networking data**, 35–37
 - business and applications data relative to, 42–44
 - control plane
 - activities in*, 41
 - data examples*, 46–47
 - defined*, 37
 - control plane communication, 38
 - data access
 - container on box*, 74–75
 - control plane data*, 67–68
 - data plane traffic capture*, 68–70
 - DPI (deep packet inspection)*, 56
 - external data for context*, 89
 - IoT (Internet of Things) model*, 75–76
 - methods of*, 55–57
 - observation effect*, 88
 - packet data*, 70–74
 - panel data*, 88
 - pull data availability*, 57–61
 - push data availability*, 61–67
 - timestamps*, 87–88
 - data manipulation
 - KPIs (key performance indicators)*, 86–87
 - made-up data*, 84–85
 - missing data*, 86
 - standardized data*, 85
 - data plane
 - activities in*, 41
 - data examples*, 47–49
 - defined*, 37
- data structure
 - JSON (JavaScript Object Notation)*, 82–83
 - semi-structured data*, 84
 - structured data*, 82
 - unstructured data*, 83–84
- data transport, 89–90
 - CLI (command-line interface) scraping*, 92
 - HLD (high-level design)*, 90
 - IPFIX (IP Flow Information Export)*, 95
 - LLD (low-level design)*, 90
 - NetFlow*, 94
 - other data*, 93
 - sFlow*, 95
 - SNMP (Simple Network Management Protocol)*, 90–92
 - SNMP (Simple Network Management Protocol) traps*, 93
 - Syslog*, 93–94
 - telemetry*, 94
- data types, 76–77
 - continuous numbers*, 78–79
 - discrete numbers*, 79
 - higher-order numbers*, 81–82
 - interval scales*, 80
 - nominal data*, 77–78
 - ordinal data*, 79–80
 - ratios*, 80–81
- encoding methods, 85
- management plane
 - activities in*, 40–41
 - data examples*, 44–46
 - defined*, 37
- network virtualization, 49–51
- OpenStack nodes, 39–40
- planes, combining across virtual and physical environments, 51–52
- sample network, 38
- networks, computer**. *See also* **IBN (intent-based networking)**
 - DNA (Digital Network Architecture), 428

- IBN (intent-based networking), 11, 428
- NFV (network functions virtualization), 51–52
- overlay/underlay, 20–22
- planes of operation, 36–37
 - business and applications data relative to*, 42–44
 - combining across virtual and physical environments*, 51–52
 - control plane*, 37, 41, 46–47
 - control plane communication*, 38
 - data plane*, 37, 41, 47–49
 - illustrated*, 438
 - management plane*, 37, 40–41, 44–46
 - network virtualization*, 49–51
 - NFV (network functions virtualization)*, 51–52
 - OpenStack nodes*, 39–40
 - sample network*, 38
 - virtualized environment*, 438
- SD-WANs (software-defined wide area networks), 20
- virtualization, 49–51
- networks, neural. *See* neural networks
- neural networks, 11, 252–258
- next-best-action analysis, 193
- next-best-offer analysis, 193
- NFV (network functions virtualization), 5–6, 51–52, 365
- Ng, Andrew, 267
- N-grams, 263
- NLP (natural language processing), 165–166, 262–263
- NLTK, 263
- nlTK package, 283, 328
- noise reduction, syslog telemetry use case, 360–362
- nominal data, 77–78
- normal distributions, 222–223
- normality, tests for, 311–313
- not-invented-here syndrome, 122
- novelty detection, 153–155
- np (numpy package), 313

- NTOP, 68
- NTP (Network Time Protocol), 87–88

- numbers
 - continuous, 78–79
 - discrete, 79
 - higher-order, 81–82
 - interval scales, 80
 - nominal data, 77–78
 - ordinal data, 79–80
 - ratios, 80–81
- numpy package, 283, 313

O

- objects, groupby, 293–296
- observation, 138–139
- observation effect, 88
- Occam's razor, 120
- one-hot encoding, 232–233, 336
- oneM2M, 75
- Open Shortest Path First (OSPF), 41, 61, 357
- open source software, 5–6, 11, 433–434
- OpenNLP, 263
- OpenStack, 5–6, 39–41
- operation, planes of. *See* planes of operation
- operations research, 214
- operators, logical AND, 306
- optimization, business model, 201–202
- optimization use cases, 186–188
- orchestration, 11
- ordinal data, 79–80
- ordinal numbers, 232
- orthodoxies, 139–140
- OSPF (Open Shortest Path First), 41, 61, 357
- outcome bias, 124
- out-group bias, 120
- outlier analysis, 153–155, 307–310, 318–320

Outliers (Gladwell), 99
 overfitting, 219
 overlay, analytics as, 20–22

P

PACF (partial autocorrelation function), 262

packages

fillna, 342–343
 Gensim, 264, 283, 328, 331–332
 importing, 390
 matplotlib, 283
 mlexend, 283
 nltk, 283, 328
 numpy, 283, 313
 pandas, 283, 346, 357–360
 pylab, 283
 scipy, 283
 sklearn, 283
 statsmodels, 283
 table of, 283–284
 wordcloud, 283

packets

file loading, 390
 HTTP (Hypertext Transfer Protocol), 71–72
 IP (Internet Protocol), 390–391
 packet counts, 395–397
 packet protocols, 398
 IPsec (Internet Protocol Security), 73–74
 IPv4, 70–74
 port assignments, 393–394
 SSL (Secure Sockets Layer), 74
 TCP (Transmission Control Protocol), 71–72, 391
 VXLAN (Virtual Extensible LAN), 74

pairwise ANOVA (analysis of variance), 317

pandas package, 283

apply, 346
 fillna, 342–343
 log analysis with, 357–360

panel data, 88, 225–226

parsimony, law of, 120, 152

partial autocorrelation function (PACF), 262

partnerships, Cisco, 433

part-of-speech tagging, 263

pattern mining, 243–244

pattern recognition, 190

PCA (principal component analysis), 233–234

 network infrastructure analytics use case, 339–340

 syslog telemetry use case, 372–373

Pearson's correlation coefficient, 225, 236

perceptrons, 252

perspectives, gaining new, 130–131

phi, 262

physical environments, combining planes across, 51–52

pivoting, 142

planes of operation, 36–37

 business and applications data relative to, 42–44

 combining across virtual and physical environments, 51–52

 control plane

activities in, 41

communication, 38

data examples, 46–47

defined, 37

 data plane. *See also* data plane analytics use case

activities in, 41

data examples, 47–49

defined, 37

 illustrated, 438

 management plane

activities in, 40–41

data examples, 44–46

defined, 37

 network virtualization, 49–51

 NFV (network functions virtualization), 51–52

 OpenStack nodes, 39–40

- sample network, 38
- virtualized environments, 438
- planning, capacity, 180–181**
- platform crashes, statistics use case for, 288–299**
 - apply method, 295–296
 - box plot, 297–298
 - crash counts by product ID, 294–295
 - crash counts/rate comparison plot, 298–299
 - crash rates by product ID, 296–298
 - crashes by platform, 292
 - data scaling, 298
 - dataframe filtering, 290–292
 - groupby object, 293–296
 - horizontal bar chart, 289–290
 - lambda function, 296
 - overall crash rates, 292
 - router reset reasons, 290
 - simple bar chart, 289
 - value_counts function, 288–289
- Platform for Network Data Analytics (PNDA), 433**
- platforms, Cisco analytics solutions, 433**
- plots**
 - box, 221–222
 - cluster scatterplot, 410–411
 - defined, 220
 - platform crashes example, 297–299
 - Q-Q (quartile-quantile), 220, 311–312
 - software crashes example, 300–305
- PNDA (Platform for Network Data Analytics), 433**
- polynomial regression, 247**
- population variance, 167**
- ports**
 - assignments, 393–394
 - mirroring, 69
 - per-host port analysis, 403
 - profiles, 407–408
 - full*, 413–419
 - source*, 419–422
 - SME port clustering, 407–413
 - cluster scatterplot*, 410–411
 - host patterns*, 411–413
 - K-means clustering*, 408–410
 - port profiles*, 407–408
- positive correlation, 224**
- post-algorithmic era, 147–148**
- post-hoc testing, 317**
- preconceived notions, 107–108**
- Predictably Irrational* (Ariely), 108**
- prediction of trends, use cases for, 190–191**
- Predictive Analytics* (Siegel), 148**
- predictive maintenance use cases, 188–189**
- predictive maturity, 8**
- preemptive analytics, 9**
- preemptive maturity, 8**
- PrefixScan, 244**
- prescriptive analytics, 9**
- priming effect, 109, 151**
- principal component analysis (PCA), 233–234**
 - network infrastructure analytics use case, 339–340
 - syslog telemetry use case, 372–373
- proactive maturity, 8**
- probability, 228**
- problem-centric approach**
 - defined, 15–16
 - use cases versus solutions, 18–19
 - walkthrough, 17–18
- process, analytics, 437**
- profiles, port, 407–408**
 - full, 413–419
 - source, 419–422
- pro-innovation bias, 121**
- psychology use cases, 209–210**
- publisher/subscriber environment, 29**
- pub/sub bus, 29**
- pull data availability**
 - CLI (command-line interface) scraping, 59, 92

- NETCONF (Network Configuration Protocol), 60
- SNMP (Simple Network Management Protocol), 57–59
- unconventional data sources, 60–61
- YANG (Yet Another Next Generation), 60
- pull methods, 28–29
- push data availability
 - IPFIX (IP Flow Information Export), 64–67, 95
 - NetFlow, 65–66, 94
 - sFlow, 67, 95
 - SNMP (Simple Network Management Protocol) traps, 61–62, 93
 - Syslog, 62–63, 93–94
 - telemetry, 63–64, 94
- push methods, 28–29
- p-values, 227, 314–317
- pylab package, 283
- pyplot, 395
- Python packages. *See* packages

Q

- Q-Q (quartile-quantile) plots, 220, 311–312
- qualitative data, 77–78
- queries (SQL), 82
- questioning
 - CARESS technique, 137
 - example of, 135–137
 - “Five whys”, 137–138

R

- race bias, 97–98
- radio frequency identification (RFID), 210–211
- random forest, 250–251
- ratios, 80–81
- RCA (root cause analysis), 184
- RcmdrPLugin.temis, 263
- reactive maturity, 7–8

- recency bias, 111
- recommender systems, 191–194
- reconciling data, 29
- recurrent neural networks (RNNs), 254–256
- regression analysis, 101–102, 246–247
- reinforcement learning, 173, 212–213
- relational database management system (RDBMS), 82
- Remote SPAN (RSPAN), 69
- reset_index function, 414
- retention use cases, 202–204
- retrieval of information
 - algorithms, 263–264
 - use cases, 185–186
- reward functions, 186
- RFIS (radio frequency identification), 210–211
- ridge regression, 247
- right skewed distribution, 310
- RNNs (recurrent neural networks), 254–256
- roles
 - analytics experts, 25
 - analytics infrastructure model, 24–25
 - business domain experts, 25
 - data domain experts, 25
 - data scientists, 25
- root cause analysis (RCA), 184
- RSBMS (relational database management system), 82
- RSPAN (Remote SPAN), 69
- R-squared, 227
- Rube Goldberg machines, 151–152
- rules, association, 240–243

S

- Sample Explore, Modify, Model, and Assess (SEMMA), 18
- Sankey diagrams, 199
- SAS, Cisco's partnership with, 433
- scaling data, 298

- scatterplots, 410–411
- scheduling use cases, 194–195
- scipy package, 283
- scrapping, CLI (command-line interface), 59
- SDA (Secure Defined Access), 428
- SDN (software-defined networking), 61, 365
- SD-WANs (software-defined wide area networks), 20
- searches, network infrastructure analytics use case, 331–336
- seasonality, 261
- Secure Defined Access (SDA), 428
- Secure Sockets Layer (SSL), 74
- security signatures, 214
- segmentation, customer, 160
- self-leveling wireless networks, 186
- SELs (system event logs), 62
- semi-structured data, 84
- SEMMA (Sample Explore, Modify, Model, and Assess), 18
- sentiment analysis, 266–267
- sequential pattern mining, 243–244
- sequential patterns, 197
- service assurance
 - analytics infrastructure model with, 33
 - defined, 11–12
 - Service Assurance Analytics, 425
 - use cases for, 195–197
- service-level agreements (SLAs), 11–12, 196
- The Seven Habits of Highly Successful People* (Covey), 10
- severities, syslog, 359–360
- sFlow, 67, 95
- Shapiro-Wilk test, 311
- Siegel, Eric, 148
- signatures, security, 214
- Simple Network Management Protocol. *See* SNMP (Simple Network Management Protocol)
- simulations, 271
- Sinek, Simon, 148
- singular value decomposition (SVD), 265
- six hats thinking approach, 132–133
- sklearn package, 283
- SLAs (service-level agreements), 11–12, 196
- slicing data, 286
- small numbers, mental models and, 117–118
- smart meters, 189
- smart society, 213–214
- Smarter, Faster, Better* (Duhigg), 99
- SME analysis
 - dataframe and visualization library loading, 394
 - host analysis, 399–404
 - IP address packet counts, 395–397
 - IP packet protocols, 398
 - MAC addresses, 398
 - output, 404–406
 - time series counts, 395
 - timestamps and time index, 394–395
 - topology mapping information, 398
- SME port clustering, 407–413
 - cluster scatterplot, 410–411
 - host patterns, 411–413
 - K-means clustering, 408–410
 - port profiles, 407–408
- SMEs (subject matter experts), 1–2
- SNMP (Simple Network Management Protocol), 28–29
 - data transport, 90–92
 - pull data availability, 57–59
 - traps, 61–62, 93
- social filtering solution, 191
- soft data, 150
- software
 - crashes use case, 299–305
 - box plots*, 300–305
 - dataframe filtering*, 300
 - dataframe grouping*, 299–300
 - defect analysis use cases, 178–179
 - open source, 5–6, 11
- software-defined networking (SDN), 61, 365

- software-defined wide area networks (SD-WANs), 20
- solution design, 150, 274
 - breadth of focus, 274
 - operationalizing as use cases, 281
 - time expenditure, 274–275
 - workflows, 282
- sorting dataframes, 326–327
- source IP address packet counts, 396
- source port profiles, 419–422
- SPADE, 244
- SPAN (Switched Port Analyzer), 69
- Spanning Tree Protocol (STP), 41
- Spark, 28–29
- SPC (statistical process control), 189
- Spearman's rank, 225, 236
- split function, 368
- SQL (Structured Query Language), 29, 82
- SSE (sum of squares error), 227
- SSL (Secure Sockets Layer), 74
- standard deviation, 167, 222–223
- standardizing data, 85
- Stanford CoreNLP, 263
- Starbucks, 110
- Start with Why* (Sinek), 148
- stationarity, 261
- statistical analysis, 440. *See also* statistics use cases
 - ANOVA (analysis of variance), 227
 - Bayes' theorem, 228–230
 - box plots, 221–222
 - correlation, 224–225
 - defined, 220
 - longitudinal data, 225–226
 - normal distributions, 222–223
 - probability, 228
 - standard deviation, 222–223
 - statistical inference, 228
- statistical process control (SPC), 189
- statistics use cases, 153, 285
 - anomalies and outliers, 153–155
 - anomaly detection, 318–320
 - ANOVA (analysis of variance), 305–310
 - data filtering*, 305–306
 - describe function*, 308
 - drop command*, 309
 - groupby command*, 307
 - homogeneity of variance*, 313–318
 - outliers, dropping*, 307–310
 - pairwise*, 317
- benchmarking, 155–157
- classification, 157–158
- clustering, 158–160
- correlation, 160–162
- data loading and exploration, 286–288
- data transformation, 310
- data visualization, 163–165
- descriptive analytics, 167–168
- NLP (natural language processing), 165–166
- normality, tests for, 311–313
- platform crashes example, 288–299
 - apply method*, 295–296
 - box plot*, 297–298
 - crash counts by product ID*, 294–295
 - crash counts/rate comparison plot*, 298–299
 - crash rates by product ID*, 296–298
 - crashes by platform*, 292–294
 - data scaling*, 298
 - dataframe filtering*, 290–292
 - groupby object*, 293–296
 - horizontal bar chart*, 289–290
 - lambda function*, 296
 - overall crash rates*, 292
 - router reset reasons*, 290
 - simple bar chart*, 289
 - value_counts function*, 288–289
- software crashes example, 299–305
 - box plots*, 300–305
 - dataframe filtering*, 300
 - dataframe grouping*, 299–300
- time series analysis, 168–169
- voice, video, and image recognition, 170

- statsmodels package, 283
- status-quo bias, 122
- Stealthwatch, 6, 65, 427
- Steltzner, Adam, 202
- stemming, 263
- stepwise regression, 247
- stop words, 263, 329
- STP (Spanning Tree Protocol), 41
- strategic thinking, 9
- streaming data, 30
- structure. *See* data structure
- Structured Query Language (SQL), 29, 82
- subject matter experts (SMEs), 1–2
- Sullenberger, Chesley “Sully”, 99–100
- Sully*, 99–100
- sum of squares error (SSE), 227
- sums-of-squares distance measures, 167
- sunk cost fallacy, 122
- supervised machine learning, 151, 246
- support vector machines (SVMs), 258–259
- survivorship bias, 118–119
- SVD (singular value decomposition), 265
- SVMs (support vector machines), 258–259
- swim lanes configuration, 161
- Switched Port Analyzer (SPAN), 69
- switches, virtual, 69–70
- syslog, 62–63, 93–94
- syslog telemetry use case, 355, 441
 - data encoding, 371–373
 - data preparation, 356–357, 369–371
 - high-volume producers, identifying, 362–366
 - K-means clustering, 373–375
 - log analysis with pandas, 357–360
 - machine learning-based evaluation, 366–367
 - noise reduction, 360–362
 - OSPF (Open Shortest Path First) routing, 357
 - syslog severities, 359–360
 - task list, 386–387

- transaction analysis, 379–386
 - apriori function*, 381–382
 - data preparation*, 379
 - dictionary-encoded message lookup*, 380–381
 - groupby method*, 380
 - log message groups*, 382–386
 - tokenization*, 381
- word cloud visualization, 367–369, 375–379

System 1/System 2 intuition, 102–103

system event logs (SELs), 62

T

tables, contingency, 267–268

tags, data transport, 93

Talent Is Overrated (Colvin), 103

Taming the Big Data Tidal Wave (Franks), 147

task lists

- data plane analytics use case, 423–424
- syslog telemetry use case, 386–387

TCP (Transmission Control Protocol)

- packet data, 71–72
- packet format, 391

tcpdump, 68

telemetry, 441

- analytics infrastructure model, 27–28
- architecture of, 63
- capabilities of, 64
- data transport, 94
- EDT (event-driven telemetry), 64
- MDT (model-driven telemetry), 64
- syslog telemetry use case, 355
 - data encoding*, 371–373
 - data preparation*, 356–357, 369–371
 - high-volume producers, identifying*, 362–366
 - K-means clustering*, 373–375
 - log analysis with pandas*, 357–360

- machine learning-based evaluation*, 366–367
- noise reduction*, 360–362
- OSPF (*Open Shortest Path First*) routing, 357
- syslog severities*, 359–360
- task list*, 386–387
- transaction analysis*, 379–386
- word cloud visualization*, 367–369, 375–379
- term document matrix**, 336
- term frequency-inverse document frequency (TF-IDF)**, 232
- terminology**, 7
- tests**, 219, 220
 - F-tests, 227
 - Levene's, 313
 - normality, 311–313
 - post-hoc testing, 317
 - Shapiro-Wilk, 311
- Tetration**, 6, 430–431
- text analysis**, 256–262
 - information retrieval, 263–264
 - NLP (natural language processing), 262–263
 - nominal data, 77–78
 - ordinal data, 79–80
 - sentiment analysis, 266–267
 - topic modeling, 265–266
- TF-IDF (term frequency-inverse document frequency)**, 232
- thinking**
 - innovative, 127–128, 439
 - associative thinking*, 131–132
 - bias and*, 128
 - breaking anchors*, 140
 - cognitive trickery*, 143
 - crowdsourcing*, 133–134
 - defocusing*, 140
 - experimentation*, 141–142
 - inverse*, 204–206
 - inverse thinking*, 139–140
 - lean thinking*, 142
 - metaphoric thinking*, 130–131
 - mindfulness*, 128–129
 - networking*, 133–135
 - observation*, 138–139
 - perspectives*, 130–131
 - questioning*, 135–138
 - quick innovation wins*, 143–144
 - six hats thinking approach*, 132–133
 - unpriming*, 140
 - strategic, 9
- Thinking Fast and Slow* (Kahneman), 102
- thinking hats approach**, 132–133
- thrashing**, 122
- tilde (~)**, 291–292, 370
- time index**
 - creating from timestamp, 357–358
 - data plane analytics use case, 394–395
- time series analysis**, 168–169, 259–262
- time series counts**, 395
- time to failure**, 183–184
- TimeGrouper**, 395
- timestamps**, 87–88
 - creating time index from, 357–358
 - data plane analytics use case, 394–395
- tm**, 263
- tokenization**, 263, 328
 - syslog telemetry use case, 371
 - tokenization, 381
- topic modeling**, 265–266
- traffic capture, data plane**, 68–69
 - ERSPAN (Encapsulated Remote Switched Port Analyzer), 69
 - inline security appliances, 69
 - port mirroring, 69
 - RSPAN (Remote SPAN), 69
 - SPAN (Switched Port Analyzer), 69
 - virtual switch operations, 69–70
- training data**, 219
- transaction analysis**
 - explained, 193, 197–199

- syslog telemetry use case, 379–386
 - apriori function*, 381–382
 - data preparation*, 379
 - dictionary-encoded message lookup*, 380–381
 - groupby method*, 380
 - log message groups*, 382–386
 - tokenization*, 381
- transformation, data, 310
- translation, language, 11
- Transmission Control Protocol (TCP), 391
- transport of data, 89–90
 - analytics infrastructure model, 26–28
 - CLI (command-line interface) scraping, 92
 - HLD (high-level design), 90
 - IPFIX (IP Flow Information Export), 95
 - LLD (low-level design), 90
 - NetFlow, 94
 - other data, 93
 - sFlow, 95
 - SNMP (Simple Network Management Protocol), 90–92, 93
 - Syslog, 93–94
 - telemetry, 94
- traps (SNMP), 61–62
- trees, decision
 - example of, 249–250
 - random forest, 250–251
- trends, prediction of, 11–12, 190–191
- troubleshooting, machine learning-guided, 350–353
- truncation, 263
- TrustSec, 427
- Tufte, Edward, 163
- Tukey post-hoc test, 317
- tunnel vision, 107
- types, 76–77
 - continuous numbers, 78–79
 - discrete numbers, 79
 - higher-order numbers, 81–82
 - interval scales, 80

- nominal data, 77–78
- ordinal data, 79–80
- ratios, 80–81

U

- UCS (Unified Computing System), 62
- unconventional data sources, 60–61
- underlay, 20–22
- Unified Computing System (UCS), 62
- unpriming, 140
- unstructured data, 83–84
- unsupervised machine learning
 - association rules, 240–243
 - clustering, 234–239
 - collaborative filtering, 244–246
 - defined, 151, 234
 - sequential pattern mining, 243–244
- use cases, 439
 - algorithms, 3–4
 - autonomous applications, 200–201
 - benefits of, 147–149, 273–274
 - building
 - analytics infrastructure model*, 275–276
 - analytics solution design*, 274
 - code*, 280–281
 - data*, 276–278
 - data science*, 278–280
 - environment setup*, 282–284
 - time expenditure*, 440
 - workflows*, 282
 - business model analysis, 200–201
 - business model optimization, 201–202
 - churn and retention, 202–204
 - control plane analytics, 441
 - data plane analytics, 389, 442
 - assets*, 422–423
 - data loading and exploration*, 390–394
 - full port profiles*, 413–419
 - investigation task list*, 423–424
 - SME analysis*, 394–406

- SME port clustering*, 407–413
 - source port profiles*, 419–422
- defined, 18–19, 150
- development, 2–3
- dropouts and inverse thinking, 204–206
- engagement models, 206–207
- examples of, 32–33
- fraud and intrusion detection, 207–209
- healthcare and psychology, 209–210
- IT analytics, 170
 - activity prioritization*, 170–173
 - asset tracking*, 173–175
 - behavior analytics*, 175–178
 - bug and software defect analysis*, 178–179
 - capacity planning*, 180–181
 - event log analysis*, 181–183
 - failure analysis*, 183–185
 - information retrieval*, 185–186
 - optimization*, 186–188
 - prediction of trends*, 190–191
 - predictive maintenance*, 188–189
 - recommender systems*, 191–194
 - scheduling*, 194–195
 - service assurance*, 195–197
 - transaction analysis*, 197–199
- logistics and delivery models, 210–212
- machine learning and statistics, 153
 - anomalies and outliers*, 153–155
 - benchmarking*, 155–157
 - classification*, 157–158
 - clustering*, 158–160
 - correlation*, 160–162
 - data visualization*, 163–165
 - descriptive analytics*, 167–168
 - NLP (natural language processing)*, 165–166
 - time series analysis*, 168–169
 - voice, video, and image recognition*, 170
- network infrastructure analytics, 323–324, 441
 - data encoding*, 328–331, 336–337
 - data loading*, 325–328
 - data visualization*, 340–344
 - dimensionality reduction*, 337–340
 - DNA mapping and fingerprinting*, 324–325
 - environment setup*, 325–328
 - K-means clustering*, 344–349
 - machine learning-guided troubleshooting*, 350–353
 - search challenges and solutions*, 331–336
- operationalizing solutions as, 281
- packages for, 283–284
- reinforcement learning, 212–213
- smart society, 213–214
- versus solutions, 18–19
- statistics, 153, 285, 440
 - anomalies and outliers*, 153–155
 - anomaly detection*, 318–320
 - ANOVA (analysis of variance)*, 305–310
 - benchmarking*, 155–157
 - classification*, 157–158
 - clustering*, 158–160
 - correlation*, 160–162
 - data loading and exploration*, 286–288
 - data transformation*, 310
 - data visualization*, 163–165
 - descriptive analytics*, 167–168
 - NLP (natural language processing)*, 165–166
 - normality, tests for*, 311–313
 - platform crashes example*, 288–299
 - software crashes example*, 299–305
 - time series analysis*, 168–169
 - voice, video, and image recognition*, 170
- summary table, 215
- syslog telemetry, 355
 - data encoding*, 371–373
 - data preparation*, 356–357, 369–371

high-volume producers, identifying, 362–366
K-means clustering, 373–375
log analysis with pandas, 357–360
machine learning-based evaluation, 366–367
noise reduction, 360–362
OSPF (Open Shortest Path First) routing, 357
syslog severities, 359–360
task list, 386–387
transaction analysis, 379–386
word cloud visualization, 367–369, 375–379

V

validation, 219
 value_counts function, 288–289, 396, 400, 403
 values, key/value pairs, 82–83
 variables, dummy, 232
 variance, analysis of. *See* ANOVA (analysis of variance)
 vectorized features, finding, 338
 video recognition use cases, 170
 views, dataframe, 329–330, 347
 Viptela, 20
 Virtual Extensible LAN (VXLAN), 74
 virtual private networks (VPNs), 20
 virtualization
 network, 49–51
 NFV (network functions virtualization), 51–52, 365-cip

planes of operation, 51–52, 438
 virtual switch operations, 69–70
 VPNs (virtual private networks), 20
 VXLAN (Virtual Extensible LAN), 74
 voice recognition, 11, 170
 VPNs (virtual private networks), 20
 VXLAN (Virtual Extensible LAN), 74

W

Wald, Abraham, 118–119
 What You See Is All There Is (WYSIATI), 118
 whys, “five whys” technique, 137–138
 Windows Management Instrumentation (WMI), 61
 Wireshark, 68
 wisdom of the crowd, 250
 WMI (Windows Management Instrumentation), 61
 word clouds, 367–369, 375–379
 wordcloud package, 283
 workflows, designing, 282
 WYSIATI (What You See Is All There Is), 118

X-Y-Z

XGBoost, 252
 YANG (Yet Another Next Generation), 60
 Yau, Nathan, 163
 Yet Another Next Generation (YANG), 60
 zero price effect, 123