DATA SCIENCE FROM SCRATCH

THE #1 DATA SCIENCE GUIDE FOR EVERYTHING A DATA SCIENTIST NEEDS TO KNOW: PYTHON, LINEAR ALGEBRA, STATISTICS, CODING, APPLICATIONS, NEURAL NETWORKS, AND DECISION TREES

STEVEN COOPER

DATA SCIENCE

Data Science from Scratch

The #1 Data Science Guide for Everything A Data Scientist Needs to Know: Python, Linear Algebra, Statistics, Coding, Applications, Neural Networks, and Decision Trees

Steven Cooper

Table of Contents

Legal & Disclaimer

The following document is reproduced below with the goal of providing information that is as accurate and reliable as possible. Regardless, purchasing this eBook can be seen as consent to the fact that both the publisher and the author of this book are in no way experts on the topics discussed within and that any recommendations or suggestions that are made herein are for entertainment purposes only. Professionals should be consulted as needed prior to undertaking any of the action endorsed herein.

This declaration is deemed fair and valid by both the American Bar Association and the Committee of Publishers Association and is legally binding throughout the United States.

Furthermore, the transmission, duplication or reproduction of any of the following work including specific information will be considered an illegal act irrespective of if it is done electronically or in print. This extends to creating a secondary or tertiary copy of the work or a recorded copy and is only allowed with an express written consent from the Publisher. All additional right reserved.

The information in the following pages is broadly considered to be a truthful and accurate account of facts, and as such any inattention, use or misuse of the information in question by the reader will render any resulting actions solely under their purview. There are no scenarios in which the publisher or the original author of this work can be in any fashion deemed liable for any hardship or damages that may befall them after undertaking information described herein.

Additionally, the information in the following pages is intended only for informational purposes and should thus be thought of as universal. As befitting its nature, it is presented without assurance regarding its prolonged validity or

interim quality. Trademarks that are mentioned are done without written consent and can in no way be considered an endorsement from the trademark holder.

Preface

The main goal of this book is to help people take the best actionable steps possible towards a career in data science. The need for data scientists is growing exponentially as the internet, and online services continue to expand.

Book Objectives

This book will help you:

Know more about the fundamental principles of data science and what you need to become a skilled data scientist.

Have an elementary grasp of data science concepts and tools that will make this work easier to do.

Have achieved a technical background in data science and appreciate its power.

Target Users

The book is designed for a variety of target audiences. The most suitable users would include:

- Newbies in computer science techniques
- Professionals in software applications development and social sciences
- Professors, lecturers or tutors who are looking to find better ways to explain the content to their students in the simplest and easiest way
- Students and academicians, especially those focusing on data science and software development

Is this book for me?

This book is for those who are interested in data science. There are a lot of skills that a data scientist needs, such as coding, intellectual mindset, eagerness to make new discoveries, and much more.

It’s important that you are interested in this because you are obsessed with this kind of work. Your driving force should not be money. If it is, then this book is not for you.

Introduction

Data is all around us, in everything that we do. Data science is the thing that makes human beings what they are today. I'm not talking about the computer-driven data science that this book is going to introduce you to, but our brain's ability to see different connections, learn from previous experiences and come to conclusions from facts. This is truer for humans than any other species that have lived on the planet. We humans depend on our brains to survive. Humans have used all of these features to earn out spot in nature. This strategy has worked for all of us for centuries, and I doubt we will be changing anything any time soon. But the brain is only able to take us so far when we are faced with raw computing. The humans can't keep up with all of the data that we are able to capture. Therefore, we end up turning to machines to do some of the work: to notice the patterns, come up with connections, and to give the answers to many different questions.

Our constant quest for knowledge is ingrained in our genes. Using computers to do some of the work for us is not, but it is where we are destined to go.

Welcome to the amazing world of data science. While you were looking over the table of contents, you may have noticed the wide variety of topics that is going to be covered in this book. The goal for *Data Science from Scratch* is to give you enough information about every little section of data science to help you get started. Data science it a big field, so big that it would take thousands of pages to give you every bit of information that makes up data science.

In each chapter, we will cover a different aspect of data science that is interesting.

I sincerely hope that the information in this book will act as a doorway for you into the amazing world of data science.

Roadmap

Chapter one will give you a basic rundown of what data science is. It will go into the importance, the history, and the reasons data science matters so much.

Chapter two will go into everything that you need for data science. This will include the work ethics that are needed to make sure you are successful.
Chapter three will cover the advantages of data science. You will see the reason why so many people love data science.
Chapter four will cover how data science differs from big data, and how the two work together.
Chapter five will go into what a data scientist is and what they do. It will also cover the skills that a person needs to be a good data scientist. It's important for a data scientist to be inquisitive, ask questions, and make new discoveries.
Chapter six will go into the reasons why a data scientist should be familiar with hacking.
Chapter seven will cover the why data scientists need to know how to code. You will also learn about the most common programming languages that data scientists use.
Chapter eight will talk about how a data scientist works with data, such as munging, cleaning, manipulating, and rescaling.
Chapter nine will go in depth about why using Python programming language is so important for a data scientist.
Chapter ten will look at the differences and similarities between data science, analytics, and machine learning.
Chapter eleven will teach you how to use linear algebra for data science.
Chapter twelve will go into the importance and use of statistics for data science.
Chapter thirteen will explain what decisions trees are and how to use them.
Chapter fourteen will explain what neural networks are and they way they are used.
Chapter fifteen will go into the different scalable data processing frameworks and paradigms, such as hadoop.
Chapter sixteen will cover all the applications of data science, such as process management, marketing, and supply chain management.

Code

Besides the sections in chapter seven where we will look at a few other programming languages, all the rest of code will be written in Python script. Python has been developed and has now become a very well respected and widely used language for the data scientists. So much so that it is pretty much the only language that data scientists use.

Whenever code appears in this book, it will be written in italic and will start and end with quotes. The quotes at the beginning and end should not be used when you type your own code, only use the italicized code. All of the codings will be explained so that you aren't confused about what it is supposed to do or how it should be used.

As you dive deeper into data science you will find that there are lots of libraries, toolkits, modules, and frameworks that efficiently use some of the most common, and least common, data science techniques and algorithms. If you do end up becoming a data scientist, you will more than likely become intimately connected to NumPy, with pandas, with sci-kit-learn, and with many other libraries. These are all great tools for data science, but they are also ways for people who know nothing about data science to get started.

This book approaches the world of data science from scratch. This means that we will be starting on the ground floor and working our way up to a better understanding of data science so that you understand all of its many aspects.

It's now time to get started on that journey. Make sure you are ready. It may even help to read the book through once, and then read it through again while working along with it. This will ensure that you fully understand what you're doing, and not just blindly following along.

Data Science and its Importance

“Information is the oil of the 21st century, and analytics is the combustion engine.” – Peter Sondergaard, SVP, Garner Research

An interdisciplinary field, data science uses scientific systems, algorithms, processes, and other methods to gain insight and knowledge from data in different forms, both unstructured and structured. It is a lot like data mining. The concept of data science is to help unify statistics, machine learning, data analysis, and other related methods. That way people will better understand and analyze information with data. It uses different theories and techniques that are drawn from different fields within the context of computer science, information science, statistics, and mathematics.

Jim Gray, a Turing award winner, saw data science as a “fourth paradigm” of science: computational, theoretical, empirical, and driven by data. He also asserted that all parts of science are changing due to the impact of data deluge

and information technology.

Data science became a buzzword when the Harvard Business Review called it "The Sexiest Job of the 21st Century," it became a buzzword. Because of this, it tends to be used to describe predictive modeling, business intelligence, business analytics, or other uses of data, or to make statistics sound more interesting. We're going to make sure that you learn what real data science is, so that you can reap the real benefits.

Data science has popped in lots of different contexts over the past 30 years or so, but it didn't become established until quite recently. Peter Naur used it to describe computer science in the '60s. Naur later started using the term datalogy. Naur then published the *Concise Survey of Computer Methods* in 1974. It referenced data science freely in the way it looked at the common data processing methods that were used in different ways.

The IFCS met in Kobe, Japan for a biennial conference in 1996, and it was here that the phrase "data science" was first included in the conference title. This was after Chikio Hayashi introduced the term at a roundtable discussion.

When he was given the H.C. Carver Professorship at UM, C.F. Jeff Wu, in November 1997, called his inaugural lecture "Statistics = Data Science?" During his lecture, he characterized statistics as a combination of decision making, data analysis and modeling, and data collection. As he concluded his speech, he used the modern non-computer use of data science and suggested that statistics should be renamed data science.

In 2001 William Cleveland introduced data science as its own discipline. He extended statistics so that it would incorporate the changes in data computing. Cleveland gave six areas of technical application that he thought encompassed the field of data science: theory, tool evaluation, pedagogy, data computing, methods and models for data, and multidisciplinary investigations.

The IEEE launched a Task Force on Data Science and Advanced Analytics in 2013. The European Association for Data Science was established in

Luxembourg in the same year. The IEEE had their first international conference in 2014. Later in 2014, The Data Incubator then created a data science fellowship, and the General Assembly created a student-paid boot camp.

What is it Exactly?

At the core of data, science is data. There are troves of raw information that is being streamed in and then stored in data warehouses. There is a lot to learn through mining it. There are advanced capabilities that can be built from it. This means that data science is basically using data in creative ways to add business value. It flows like this:

- Data Warehouse
    - Discovery of Data - Insight and quantitative analysis to help strategic business decision making.
    - Data Product Development– algorithmic solutions in production and operating.
    - Business Value

The main aspect of data science is discovering new results from data. People are exploring at a granular level to understand and mine complex inferences, behaviors, and trends. It's about uncovering hidden information that may be able to help companies make smarter choices for their business. For example:

- Data mines in Netflix are used to look for movie viewing patterns to better understand user's interests and to make decisions on the Netflix series they should produce.
- Target tries to find the major customer segments in its customer base and their shopping behaviors, which helps them to guide messaging to other market groups.
- Proctor & Gamble looks towards time series models to help them to understand future demand and plan production levels.

So how does the data scientist mine all this information? It begins with data exploration. When a data scientist is given a challenging question, they become a detective. They will start to investigate leads, and then try to understand characteristics or patterns in the data. This means they need a lot of analytical creativity.

Then a data scientist can use quantitative techniques to dive a little deeper, such as synthetic control experiments, time series forecasting, segmentation, and inferential models. The purpose of these is to use data to piece together a better understanding of the information.

The use of data-driven insight is what helps to provide strategic guidance. This means that a data scientist works a lot like a consultant, guiding businesses on how they should respond to their findings.

Data science will then give you a data product. Data products are a technical asset that:

1. **Uses data like input.**
2. **Processes the data to get an algorithmically-generated result.**

One of the classic examples of a data product is an engine which takes in user data, and then creates a personalized recommendation based upon that data. The following are some examples of data products:

- The recommendation engine that Amazon uses suggests new items to its users, which is determined by their algorithms. Spotify recommends new music. Netflix recommends new movies.
- The spam filter in Gmail is a data product. This is a behind the scenes algorithm that processes the incoming mail and decides whether or not it is junk.
- The computer vision that is used for self-driving cars is also a data product. Machine learning algorithms can recognize pedestrians, traffic lights, other cars, and so on.

Data products work differently than data insights. Data insights help to provide some advice to help a business executive make smarter decisions. Data products are a technical functionality that encompasses the algorithm, and it is designed to work into the main applications.

The data scientist plays one of the most central roles in coming up with the data product. This means that they have to build out algorithms and test, refine, and technically deploy it into a production system. The data scientist also works as a

technical developer by creating assets that become leverage on a wide scale.

Why It Matters

Big data means nothing if you don't have professional expertise to turn the data into actionable items. Today, there are lots of institutions and organizations in the financial world that are opening their doors to big data to unlock its power. This will increase the value of the data scientist who knows exactly how to drive all the information that is housed in the institution's files.

It's well known that businesses today are full of data. In 2016 alone, McKinsey estimated that the big data initiative within the US healthcare system would cause a $300 to $450 billion reduction in healthcare costs. It could also cause a 12 to 17% reduction of the $2.6 trillion overall healthcare baseline. On the other side, badly used data is believed to be costing the US around $3.1 trillion each year.

It has become clear that value is found in the analysis and processing of data and that is where the data scientist plays an important role. Executives have all heard about the sexy industry of data science and how data scientists are the resident superheroes, but many of them are still not aware of the value of data science for their organizations.

Data science and scientist add value to all businesses in many different ways. Let's look at the eight major areas.

1. **Empowers officers and managers to make better decisions.**

An experienced data scientist will work as a strategic partner and trusted advisor to an institution's management to make sure that the staff is able to maximize their analytical capability. The data scientist will demonstrate and communicate the worth of the analytical products of the institution to help create an improved decision-making process across different levels of the organization, by recording, measuring, and tracking all performance metrics.

2. **Directs action based upon trends that will help define goals.**

Data scientists look at and explore a business's data to recommend and provide different actions that will improve the business's performance and their customer

relations, which will increase profitability.

3. **It will challenge the staff to use their best practices and to focus on important issues.**

One of the big responsibilities of the data scientist is to make sure that staff members understand and are well-versed in the data and its use. They will make sure that the staff is prepared for success by demonstrating the effective use of the system to drive action and insight. After the staff has a good understanding of the data's capabilities, they will shift their focus to addressing the important challenges of the business.

4. **Identify opportunities.**

While they are interacting with the business's analytics system, the data scientists will question the existing assumptions and processes in order to come up with additional methods and analytic algorithms. Their job will require that they are continuously and constantly improving the value that comes from the business's data.

5. **Make decisions with data-driven, quantifiable evidence.**

With data science, the use of data analyzing and gathering through different channels has gotten rid of the need to take high stake risks.

6. **Data scientists test management decisions.**

Half of the battle of running a good business is making decisions and implementing change. The other half is understanding the way that these decisions will affect the company. This is why data scientists are important. It's helpful to have somebody who can measure the important metrics that relate to changes, and quantify their success.

7. **Identifying and refining target audiences.**

From customer surveys to Google Analytics, companies will have at least a single base of customer data that they collect. But if they don't use it well, for example, to find demographics, the data will go to waste.

The data scientist will be able to help identify target audiences through in-depth analysis of multiple data sources. Once businesses have this in-depth

information, they will be able to tailor their products and services to their customer groups so that they improve their profit margins.

8. **They help to recruit the best talent for the organization.**

A recruiter has to constantly review resumes. This can change by using big data correctly. With all the information about talent through job search websites, corporate databases, and social media, data scientists can help find the best candidates for the business's needs.

This means that recruitment won't be as exhausting and time-consuming. By mining the large amount of data that businesses have available, in-housing application and resume processing, and even the fancy data-driven games and tests can be reduced into a more accurate and speedier selection process.

What You Need

“With data collection, ‘the sooner, the better’ is always the best answer.” – Marissa Mayer When it comes to data science, there are three major skill areas that are blended together.

1. **Mathematics expertise**
2. **Technology; hacking skills**
3. **Business/strategy acumen**

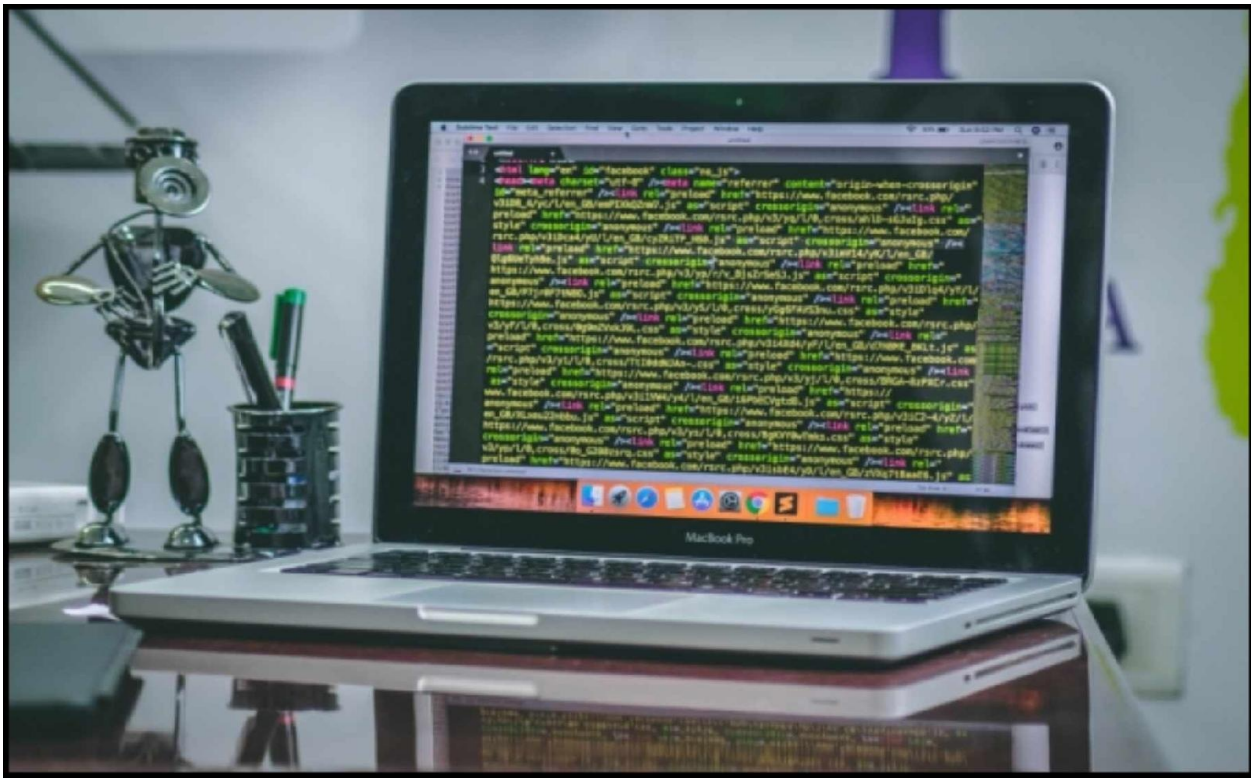
Data science is ***Where*** they all meet. The central point of mining data and creating a data product is to see the data quantitatively. There are correlations, textures, and dimensions in the data that are seen mathematically. Finding a solution through data becomes a brain teaser of quantitative and heuristic technique. To find a solution in a lot of problems involves coming up with an analytic model that is grounded in hard math. Understanding the mechanics underneath those models is crucial for success.

There is also a big misconception that data science only deals with statistics. While statistics do play an important role, it’s not the only math that is utilized. There are two main branches of statistics: Bayesian and classical statistics. When people start talking about statistics, they are most often talking about classical statistics, but understanding both is extremely helpful.

When you get into more machine learning algorithms and inferential techniques you will lean heavily on linear algebra. One example is that a popular way to find hidden characteristics within a set of data is with SCD, which has its grounding in matrix math and doesn’t have as much to do with classical statistics. Overall, it’s extremely helpful for a data scientist to have a pretty good understanding of mathematics in all areas.

When it comes to hacking, we’re not talking about breaking into other people’s computers. We are talking about the programmer subculture that is known as hacking. This is the ingenuity and creativity of using technical skills to create things and to discover new solutions to old problems.

Why is the skill of hacking important? Mainly because data scientists will use technology to help them gather a large amount of data, and then work with complex algorithms to understand it. This will require tools that tend to be more sophisticated than a spreadsheet. Data scientists must understand how to code, prototype fast solutions, and integrate complex data systems. Some of the most common languages that are associated with data science are SAS, R, Python, and SQL. Some of the less commonly used ones are Julia, Java, and Scala. But it's not just having a good understanding of these language fundamentals. A good hacker can creatively work their way through different types of challenges so that they are able to make their code work.



This means that a data science hacker is great at algorithmic thinking, which means that they can break down tough problems and then rework them so that they are solvable. This is crucial because they must work with a lot of algorithmic problems. They need a good mental comprehension of tricky and high-dimensional data control flows. They must be fully clear on how all of the pieces work together to create a cohesive solution.

It’s also important that a data scientist is a tactical business consultant. Since data scientists work closely with data, they can learn things from data that other people can’t. This makes them responsible for translating their observations into shared knowledge and sharing their strategy on how they think the problem should be solved. A data scientist needs to be able to share a clear story. They shouldn’t just throw out data. It needs to be presented in a cohesive discussion of a problem and its solution which uses data insights its basis.

Having business acumen plays just as an important role as having an acumen for algorithms and tech. There must be a clear match between business goals and data science projects. In the end, the value won’t come from the tech, data, and math. It will come from leveraging all this information into valuable results for the business.

Let’s break down the prerequisites for data scientists a bit further.

- **Technical Skills**
    - Adept at working data that is unstructured.
    - Understanding of SAS and other analysis tools.
    - Skills in programming
    - Ability to data process and mine.
    - Skills in statistical analysis.
    - Preferably a Master’s or Ph.D. in engineering, statistics, or computer science.
- **Non-Technical Skills**
    - Great data intuition.
    - Strong communication skills.
    - A strong business acumen.

The Advantages to Data Science

“Hiding within those mounds of data is knowledge that could change the life of a patient or change the world.” – Atul Butte, Stanford Quickly progressing technology around the world results in big data as its byproduct. It can be seen everywhere; from the information within your smartphone and apps to the idea of a car that drives itself. This new modern phenomenon is why data scientists are becoming more necessary. Data science has been labeled as the sexiest career path of the 21st century, and it will only continue to grow and develop throughout the coming years.

Because it is still a novel profession, there are a lot of people that aren’t completely aware of the many possibilities that come with being a data scientist. Those that are interested in this kind of work can look forward to having an outstanding salary and a rewarding career.

Here is a breakdown of where data scientists work:

- 2% of data scientists work in gaming.
- 4% work in consumer goods and retail.
- 4% work in academia.
- 4% work in government.
- 6% work in financial services.
- 7% work in pharmaceuticals and healthcare.
- 9% work in consulting.
- 11% work in a corporate setting.
- 13% work in marketing.
- 41% work in technology.

There are a lot of specific industries that are in high demand for people who are well versed in data science. It’s not surprising that the biggest need is in technology with 41% of the total workforce. Past the more obvious applications in tech, work in financial, healthcare, consulting, marketing, and corporate services also have a high data science need.

When data science is brought into a business, it brings along with it several different benefits. Among those are the following seven.

1. **It will monetize data.**

Facebook turns the data that they get from their subscribers into money, and so can any business. For example, there are a lot of retailer sites that will show you a section that says, “Customers Who Bought This Item Also Bought,” which will show items that is more likely to provide them another sale. This type of creative analysis is what will allow a company to increase its revenue.

2. **It will mitigate company risk.**

A data scientist will analyze client churn patterns and allow a company to react in a proactive manner if they notice that a trend of customers start favoring another business. In order to get the customers back, the business will be able to send out teaser deals discounts to retain customers.

The data scientist will also evaluate the data of other businesses that a company is looking to partner with. This will help to minimize the possible risk. For example, data science could be used to analyze the information from a third-party payment processor about business that they are considering doing working with. They can then use that analysis to assess the creditworthiness of the company.

3. **It will help a company get a better understanding of their customers.**

The behaviors of customers will change with time, and it’s hard to monitor their changes without using data science. For example, websites like Airbnb, which helps hosts and travelers rent and find affordable places to stay, recently looked at the behavior of their consumers during website searches and changed their algorithm engine to give them better results. Because of this change, their reservations and bookings went up. It’s this kind of insight into the actions of consumer that a data scientist can find so that they can improve the business.

4. **It will give businesses unique insights.**

Let’s assume that a data scientist finds out that there is a connection between

winter snowstorms and the sale of Cocoa Krispies. If a grocer had this information, they could strategically place Cocoa Krispies in the store during snowstorms so that they can increase their sales. This would be a rather impossible connection to come up with without data science.

5. **It will help with business expansion.**

A data scientist could end up uncovering new markets that could be interested in a business's service or product. An advertising campaign could be solid, but the data scientist could end up looking it over and could figure out the type of customers gained for a certain initiative so that the business can adjust future campaigns. Data science can be used to find new trends, or it could figure out which inventory items will have a faster impact on revenues.

6. **It will improve forecasting.**

Neural networks and machine learning have been used to mine business data for quite some time to predict future results, and there are a lot of data scientists that have skills in both. For example, a business that deals in car repair could analyze spikes in visits over the past few years so that they can come up with a better schedule for their employees.

7. **It will provide businesses with objective decisions.**

Data is a powerful tool that speaks for itself. Having verifiable and solid data on hand will help a business make better decisions that are based on objectivity. This will take precedence and emotions out of the problem. If emotions, egos, or the tendency to do things the same all the time has caused a business problem in the past, data science is able to help.

Data Science and Big Data

“Data is the new science. Big data holds the answers.” – Pat Gelsinger, CEO, VMware

An unprecedented growth of information generated around the world and on the internet that has resulted in big data. Big data refers to the large group of heterogeneous data that comes from various sources and isn’t typically available in standard database formats that everybody is aware of. This data encompasses all different types of data; unstructured, semi-structured, and structured information that can be found easily throughout the internet.

Big data includes:

- Structured data: transaction data, OLTP, RDBMS, and other structured formats.
- Semi-Structured: text files, system log files, XML files, etc.
- Unstructured data – web pages, sensor data, mobile data, online data

sources, digital audio, and video feeds, digital images, tweets, blogs, emails, social networks, and other sources.

All information and data, no matter its format or type, can be considered big data. The processing of big data will normally start with aggregating data from several sources.

Processing of big data can’t be achieved through traditional Methods. Instead, when it comes to unstructured data, you will need specialized data modeling systems, techniques, and tools to remove information and insights as needed by businesses. Data science will come in as a scientific approach that will apply statistical and mathematical ideas as well as computer tools to process big data. Data science uses different areas like data programming, mining, cleansing, intelligent data capture techniques, mathematics, and statistics to align and prepare big data to find information and insights. Data science can be challenging because of the complexities involved in applying and combing different complex programming techniques, algorithms, and methods to perform intelligent analysis on big amounts of data. Data science has evolved out of big data, but there are plenty of differences between data science and big data.

Key Difference Between Data Science and Big Data

Here are the main differences between data science and big data.

- Organizations have to gather big data to help improve their efficiency, enhance competitiveness, and understand new markets. Data science provides the mechanisms or tools to understand and use big data quickly.
- There is no limit to how much valuable data that can be collected. To use this data, the important information for business decisions has to be extracted. This is where data science is needed.
- People characterize big data by its volume, velocity, and variety, which is often referred to as the 3Vs. Data science provides the techniques and methods to look at the data that is characterized by the 3Vs.
- Big data provides a business with the possibility for better performance. However, finding that information in big data to utilize

its potential to enhance performance is a challenge. Data science will use experimental and theoretical approaches as well as inductive and deductive reasoning. It will unearth the hidden insight from all the complexity of unstructured data, which will support organizations to notice the potential of all the big data.

- Big data analysts perform mining of helpful information from large sets of data. Data scientists use statistical methods and machine learning algorithms to train computers to find information without a lot of programming, and to make predictions based upon big data. Because of this, it's important that you don't confuse data science with big data analytics.
- Big data relates to technology, such as Hive, Java, Hadoop, and so on, and is a distributed computing, and software and analytics tool. This is different than data science, which looks at the strategy for business decisions, data structures and statistics, data dissemination using math, and all other methods mentioned earlier.

Through the differences between data science and big data, you can see that data science is included as part of the concept of big data. Data science is an important part of a lot of different application areas. Data science uses big data to find the most helpful insights through predictive analysis. The results are then used to make better choices. Data science is included as a part of big data, but big data is not included in data science.

The following is a chart to help show the fundamental differences:

- **Meaning:**
  - ***Big Data:***
    - Large volumes of data that can't be handled using a normal database program.
    - Characterized by velocity, volume, and variety.
  - ***Data Science:***
    - Data focused scientific activity.

        - Similar in nature to data mining.
        - Harnesses the potential of big data to support business decisions.
        - Includes approaches to process big data.
- **Concept:**
    - ***Big Data:***
        - Includes all formats and types of data.
        - Diverse data types are generated from several different sources.
    - ***Data Science:***
        - Helps organizations make decisions.
        - Provides techniques to help extract insights and information to create large datasets.
        - A specialized approach that involves scientific programming tools, techniques, and models to process big data.
- **Basis of Formation:**
    - ***Big Data:***
        - Data is generated from system logs.
        - Data is created in organizations – emails, spreadsheets, DB, transactions, and so on.
        - Online discussion forums.
        - Video and audio streams that include live feeds.
        - Electronic devices – RFID, sensors, and so on.
        - Internet traffic and users.
    - ***Data Science:***
        - Working apps are made by programming developed models.
        - It captures complex patterns from big data and developed models.

        - It is related to data analysis, preparation, and filtering.
        - Applies scientific methods to find the knowledge in big data.
- **Application Areas:**
    - ***Big Data:***
        - Security and law enforcement.
        - Research and development.
        - Commerce.
        - Sports and health.
        - Performance optimization.
        - Optimizing business processes.
        - Telecommunications.
        - Financial services.
    - ***Data Science:***
        - Web development.
        - Fraud and risk detection.
        - Image and speech recognition.
        - Search recommenders.
        - Digital advertisements.
        - Internet search.
        - Other miscellaneous areas and utilities.
- **Approach**
    - ***Big Data:***
        - To understand the market and to gain new customers.
        - To find sustainability.
        - To establish realistic ROI and metrics.
        - To leverage datasets for the advantage of the business.
        - To gain competitiveness.
        - To develop business agility.
    - ***Data Science:***

- Data Visualization and prediction.
- Data destroy, preserve, publishing, processing, preparation, or acquisition.
- Programming skills, like NoSQL, SQL, and Hadoop platforms.
- State-of-the-art algorithms and techniques for data mining.
- Involves the extensive use of statistics, mathematics, and other tools.

Data Scientists

“It is a capital mistake to theorize before one has data.” – Sherlock Holmes

Data scientist tend to appear to be wizards who grab their magical crystal balls, chant some mumbo-jumbo, and come up with extremely detailed predictions of what a business’s future may hold. No matter how much you wish it was, data science is not magic. The power that data science brings comes from a complete understanding of algorithms and statistics, hacking and programming, and communication skills. Even more important, data science is all about using these skill sets in a systematic and disciplined way.

A data scientist manages big data. They take a large amount of data points and use their skills in programming, math, and statistics to organize, clean, and massage them. They then use their analytic powers, skepticism of existing assumptions, contextual understanding, and industry knowledge to find hidden solutions for challenges to a business.

The Process of Data Science

Let's say, you just got your first job as a data scientist a Hotshop, a startup located in San Francisco. You're on your first day of work. You're excited to get started crunching data and amaze the people around you with your new discoveries. Where are you supposed to start?

Over lunch, you meet the VP of Sales. As you introduce yourself, you ask "What data challenges should I be looking at?"

She thinks for a moment. You're waited for an answer, the answer that will help you know exactly what you are going to do to have a big impact.

She finally asks, "Would you be able to help us improve our sales funnel and improve our customer conversion rate?"

The first thing that pops into your mind is: What? Is that even a data science problem? She didn't say anything about data. What am I supposed to analyze? What does she mean?

Fortunately, your mentors have told you that this initial ambiguity is a normal situation that data scientists encounter. The only thing you have to do is correctly apply your data science process to figure out what you must do.

When a non-technical boss asks you to figure out a data problem, the description can end up being ambiguous at first. It becomes your job as the data scientist, to change the task into a problem, figure out how you can solve it, and the present your solution to the boss.

This process uses several steps:

- Frame the problem: Who is the client? What are they asking you, exactly, to solve? How are you able to translate the ambiguous request into a well-defined and concrete problem?
- Collect the data that you need to solve the problem. Do you already have access to this data? If you do, what parts of this data can help? If you don't, what data do you need? What resources, such as infrastructure, time, and money, do you need to get the data to a

usable form?

- Process your data: Raw data is very rarely able to be used right out of the box. There will be errors in the collection, missing values, corrupt records, and lots of other challenges you have to take care of. You first have to clean the data to change it into a form that you will be able to analyze.
- Explore the data: After you have the data cleaned, you need a high level of understanding of the information that is contained in it. What are the obvious correlations or trends that you see within the data? What high-level characteristics does it have, and are there any of them that is more important than the other?
- Perform in-depth analysis: This is typically the core of the project. This is where you use the machinery of data analysis to find the best predictions and insights.
- Communicate the results of your analysis: All of the technical results and analysis that you have found isn’t very valuable unless you are able to explain it in a way that is compelling and comprehensible. Data storytelling is a very underrated and critical skill that a data scientist needs to use and build.

Responsibilities of a Data Scientist

On any day a data scientist may have to:

- Recommend the most cost-effective changes that should be made to existing strategies and procedures.
- Communicate findings and predictions to IT and management departments through effective reports and visualizations of data.
- Come up with new algorithms to figure out problems and create new tools to automate work.
- Devise data-driven solutions to challenges that are most pressing.
- Examine and explore data from several different angles to find hidden opportunities, weaknesses, and trends.
- Thoroughly prune and clean data to get rid of the irrelevant information.
- Employ sophisticated analytics programs, statistical methods, and machine learning to get data ready for use in a prescriptive and predictive modeling.
- Extract data from several external and internal sources.
- Conduct undirected research and create open-ended questions.

Different companies will have a different idea of data scientist tasks. There are some businesses that will treat their data scientists like glorified data analysts, or combine the duties with data engineering. There are others that need top-level analytics experts that are skilled in intense data visualizations and machine learning.

As data scientists reach new experience levels or change jobs, the responsibilities they face will change as well. For example, a person that works alone for a mid-sized company may spend most of their day cleaning and munging data. High-level employees that are a part of a business that offers data-based services could have to create new products or structure big data projects on an almost daily basis.

Qualifications of Data Scientists

There are three education options that you will need to look at when considering a career in data science.

1. Graduate certificates and degrees provide recognized academic qualifications, networking, internships, and structure for your resume. This will end up costing you a lot of money and time.
2. Self-guided courses and MOOCs are cheap or free, targeted, and short. They will let you complete your projects within your own timeframe, but they will require you to structure your own career path.
3. Bootcamps are a lot faster and more intense than traditional degrees. They may even be taught by data scientists, but they will not provide you with a degree that has initials after your name.

Academic qualifications are probably more important than you think. It's very rare for a person that doesn't have an advanced quantitative degree to have the skills that a data scientist needs.

Burtch Works, in its salary report, found that 46% of data scientists have a PhD and 88% have a master's degree. For the most part, these degrees are in rigorous scientific, quantitative, or technical subjects, which includes statistics and math – 32%, engineering – 16%, and computer science – 19%.

Many companies are desperate to find candidates that have real-world skills. If you have the technical knowledge, it could trump the preferred degree requirements.

What skills are you going to need to be a data scientist?

1) Technical skills:

- Cloud tools such as Amazon S3.
- Big data platforms such as Hive & Pig, and Hadoop.
- Python, Perl, Java, C/C++
- SQL databases, as well as database querying languages.

- SAS and R languages.
- Unstructured data techniques.
- Data visualization and reporting techniques.
- Data munging and cleaning.
- Data mining
- Software engineering skills
- Machine learning techniques and tools.
- Statistics
- Math

This list is always changing as data science changes.

2) Business Skills:

- Industry knowledge: It’s important to understand how your chosen industry works and how the data is utilized, collected, and analyzed.
- Intellectual curiosity: Data Scientists have to explore new territories and find unusual and creative ways to solve problems.
- Effective communication: Data Scientists have to explain their discoveries and techniques to non-technical and technical audiences in a way that they can understand.
- Analytic problem-solving: Data Scientists approach high-level challenges with clear eyes on what is important. They employ the right methods and approaches to create the best use of human resources and time.

Would You Be a Good Data Scientist?

To figure out whether or not you would make a good data scientist, ask yourself these questions:

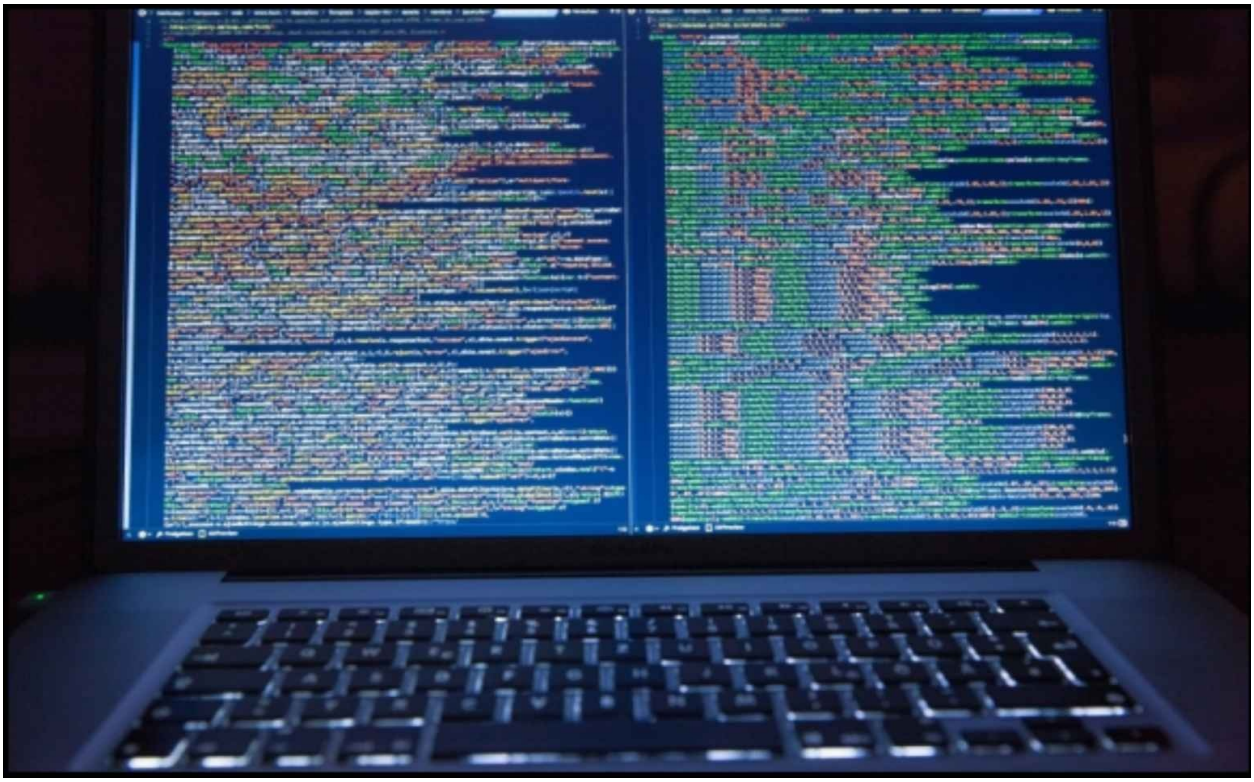
- Are you interested in broadening your skills and taking on new challenges?
- Do you communicate well both visually and verbally?
- Do you enjoy problem-solving and individualized work?
- Are you interested in data analysis and collection?
- Do you have substantial work experience in the areas involved in data science?
- Do you have a degree in marketing, management information systems, computer science, statistics, or mathematics?

If you were able to answer yes to any of these questions, then you will probably find a lot of enjoyment in data science.

It's important that data scientists have knowledge of statistics or math. It's also important that they have a natural curiosity, such as critical thinking and creativity. What are you able to do with the data? What undiscovered information is hidden within the data? You need to have the ability to connect the dots and have a desire to find the answers to these questions you haven't been asked if you notice that there is data that is full of potential.

The Importance of Hacking

“Torture the data, and it will confess to anything.” – Ronald Coase, Economics, Nobel Prize Laureate



According to Drew Conway, a political science Ph.D. student at New York University and a former intelligence community member, skills for a data scientist should be broken into three categories:

1. **Hacking skills**
2. **Statistical and math knowledge**
3. **Substantive expertise**

Having hacking skills is important because data tends to be inside several locations and in different systems, which makes discovering them just a bit challenging.

Data hackers typically have a broad set of technical skills, but they likely won’t be a complete expert in any of the skills, such as:

- Data Munging
- Big data

- Databases
- Dashboarding/Reporting
- Visualization
- Machine learning
- Statistical programming

That's a pretty long list, so how can a person learn all of these things in a decent amount of time? They have to choose one comprehensive technology stack, and then complete everything in the stack.

One such technology stack would be the R-Hadoop stack. R is an open-source and free statistical programming language that was originally based upon the S programming language. There are a few reasons why a person may choose to start with R for data analysis:

- R works great for getting your job done, especially when it comes to the tech industry.
- R is a complete programming language, unlike SPSS or SAS, R isn't only a procedural language.
- R is quite easy to learn and is great for hacking. You won't have to have a lot of experience with programming to be able to get started doing decent work with R.
- It is comprehensive. Nearly any machine-learning or statistical task you are able to dream up has a pre-built library in R.
- It's free. SPSS and SAS tend to be expensive to get started in, and you will normally have to buy new methods if you are interested in trying them out.

Hadoop is an open-source and free distributed computing framework. Hadoop is typically used for every part of big data: modeling, databases, and analysis. A lot of the top companies use Hadoop, which includes LinkedIn, Facebook, and Twitter. Whenever you hear somebody talking about Hadoop, you will probably end up hearing about MapReduce, which is a framework that gives you the ability to solve large-scale data problems with clusters of commodity computers.

The following are some reasons why Hadoop is perfect system for starting out with big data:

- Hadoop is a perfect system to get your job done, and it seems as if it’s on every job ad for a data scientist.
- Hadoop is comprehensive. Pretty much all big data processing and storage problems can be figured out using the Hadoop system.
- It is quite easy to get started with, even if you don’t have a cluster of computers. Cloudera is a great service to check out with its online trail and a VM that you are able to download completely free.
- Again, it is free.

The R-Hadoop stack gives you the ability to do pretty much anything that you would need to form data hacking, such as:

- Data munging: This means to clean data and then rearrange it in a way that is more useful. Imagine something like parsing unusual date formats, turning columns into rows, getting rid of malformed values, and so on. Hadoop and R have applications to help with this. R is an amazing and easy way to process small to moderate sized sets of data. Hadoop gives you the ability to write out your own programs, and then rearrange and clean all of the large sets of data when you need to.
- Big data: This is the main purpose of Hadoop. Hadoop gives you the ability to process and store essentially an unlimited amount of data on standard commercial hardware. There is no need for a supercomputer. Depending and the size of data, R has a pretty good selection of libraries that you can work directly with, such as data.table.
- Databases: Hadoop has a scalable data warehouses system built on it called Hive that works for ad-hoc SQL-style querying of large sets of data. This was developed at Facebook. HBase, which is used by Twitter, and Cassandra, which is used by Netflix, are other types of database solutions that have been built on Hadoop.

- Dashboarding and reporting: R has the knit package that will give you the ability to create dynamic and beautiful reports. This package is a web framework that is used for creating interactive and stylish web apps.
- Visualization: Using the ggplot2 package, you can create completely customizable and professional looking 2D plots.
- Machine learning: The caret package provides a wrapper for a lot of algorithms, and it makes it super easy to test, train, and tune machine learning models.
- Statistical programming: R comes with a package that is used for data exploration, regression, statistical tests, and pretty much anything else that you could think of.

You can use Hadoop and R on a Windows computer, but they work a lot better and more naturally on a Unix-based system. That system might end up being a bit of a learning curve, but what you get from using a Unix system is amazing, and it will look great on a resume.

Hadoop and R may be able to cover most cases, but there are some situations where you may be looking to use a different feature. For example, Python has a library that will make text mining a lot more scalable and easier than R does. And if you are interested in creating a web app, Shiny may not be flexible enough so you will want to go with a more traditional web framework. For the most part, you should be able to get by Hadoop and R. Now we will be going into more depth about Python because it is more commonly used for data science than R is.

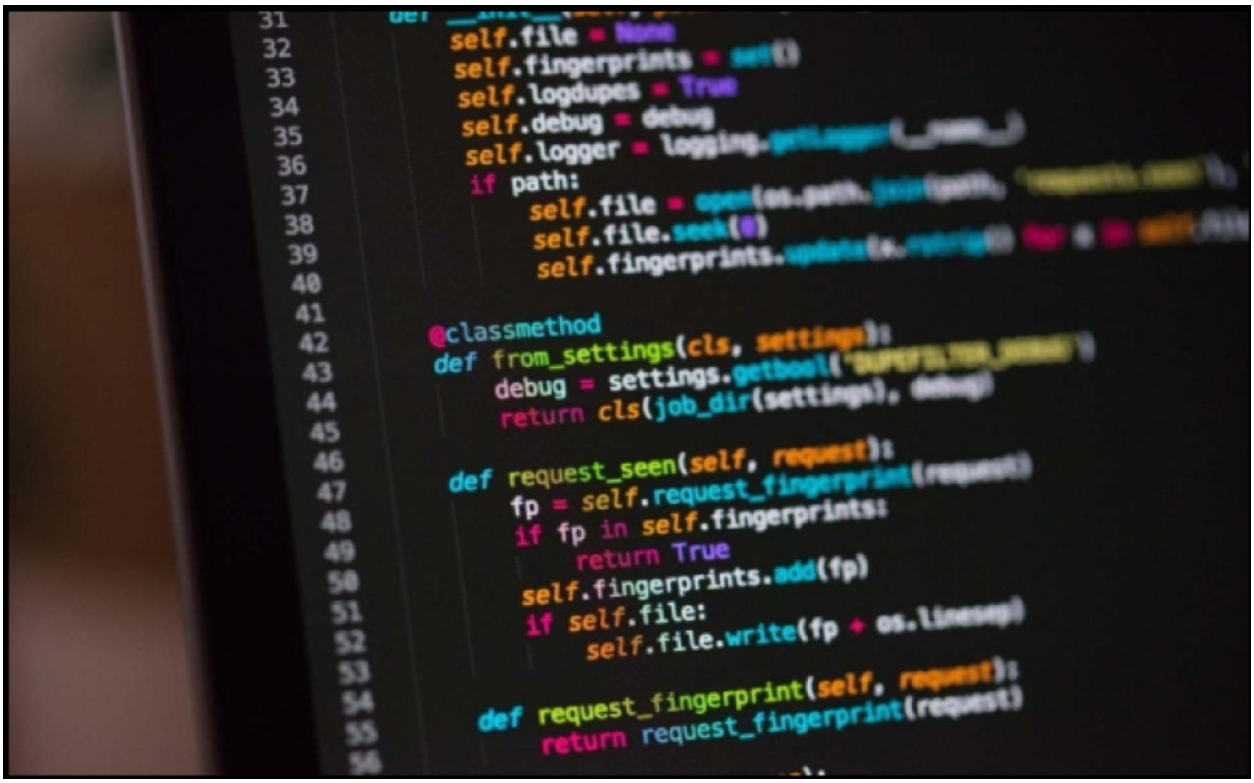
You’re probably wondering why you should stick with learning only one technology stack. A lot of people think that they should use the right tool for the job, and they would be afraid that only learning one would get them stuck in the data science ecosystem. These are all very good points but focusing on a single stack comes with its advantages, especially if you’re just starting out. First off, if you are switching training paths and resources a lot, you will end up wasting a

bunch of time. Secondly, it becomes motivating and useful to focus l on a single technology because when you get good at one thing, it is a faster way of solving problems.

In the references chapter you will see links to help pages from Hadoop, R, and several other technologies we will discuss in this book.

The Importance of Coding

“Having skills in statistics, math, and programming is certainly necessary to be a great analytic professional, but they are not sufficient to make a person a great analytic professional.” – Bill Franks, Chief Analytics Officer at Teradata



The coding skills that you need to get into data science will depend on the area of data science that you may end up working on. If you end up managing databases, it’s important that you know that as more enterprises start using data science, legacy skills, like SQL, will hang around. Larger companies will end up using SQL through their operations.

If you want to do more with the data that you have collected, it may be best if you expand your knowledge of SQL with a focus on skills such as managing, collecting, and storing data. This may seem obvious, but it is worth keeping this in the back of your mind as we look at the trends.

If you plan to use your data to perform visualization, analytics, and modeling,

you will want to make sure you are strong in Java, Python, and R.

R has started to become the lingua franca for the pure data scientist, especially when it is used in scientific research and finance. It works as a procedural language, instead of an object-oriented language such as Java or Python. It will likely require more code to get the job done, but there is more that you can do with it. R also has a granular functionality that a lot of data scientists prefer to use, especially whenever it comes to having to deal with a lot of data.

This being said, Java is extremely scalable and fast. While R is able to present a lot more options when it comes to working with complex data issues, there are a lot of startups, as well as other businesses, who love Java for giving them more bang for their product development and developer training buck.

Python tends to fall in between things. It is able to do a lot, it's scalable, and it's fast. With a skills market that likes to have a good enough for enough uses, the best thing to turn to is Python.

The fresh areas for the most growth will circle around deep learning, AI, and machine learning. The number of people that work in data science who have these skills has doubled over the past three years, and now takes up almost a third of the industry. The great thing about Java, Python, and R is that they all plug into machine learning, so taking the time to learn any of these skills is time well spent.

When it comes specifically to deep learning, Google TensorFlow has changed quickly to get a strong leadership position, and it is followed closely by Keras. A little bit of an interesting note about TensorFlow. It is written in C++ which, until just a few years ago, was the leading language of data science. TensorFlow actually runs on a Python interface that works on a C++ foundation, which means you won't have to understand C++ coding in order to use TensorFlow.

This is the kind of dynamic that is taking place within the world of data science and analytics. As data starts to become ubiquitous, the uses will start to become innumerable, and the amount of code-based solutions will grow exponentially. This will, in turn, drive a market for data science tools that will help to simplify

the coding process.

The main point? View this like it is college: while you may pick out a programming language for your major, it doesn’t hurt to create a working knowledge of several others as your minors. Things in the data science world change fast, and these are very exciting times for data scientist and coders.

Writing Production-Level Code

When it comes to the most sought-after skills for a data scientist, it is to write production-level code. If you were a software engineer before you became a data scientist, this probably doesn't sound all that challenging as you would already have perfected the skill of developing code and deploying it into production. This section is here to help those that are new to writing code and are interested in learning. This is for those that view writing code as a formidable task.

1. **Keep it modular**

This is pretty much just a software design technique that is recommended for software engineers. The idea is that you should break large code into small working parts based on their functionality. This can be broken down into two parts.

One, break up the code into small pieces that each have a specific task to perform.

Two, group the functions into different modules, or Python files, based on their usability. It helps you to stay organized, and it makes the code more maintainable.

Your first step would be to decompose large code into as many easy functions with certain outputs and inputs as possible. Each one of the functions needs to perform one task, such as calculate root-mean-squared error, score a model, replace erroneous values, cleanup outliers in the data, and so on. Try to break up each of these functions a bit further into subtasks and continue doing this until none of these functions are able to be broken down further.

Low-level functions – these are the most basic functions that aren't able to be further broken down. For example, computing Z-score or RMSE of the data. These functions are able to be widely used for implementation and training of any machine learning or algorithm model.

Medium-level functions – these are functions that use one or more medium or low-level function to perform a certain task. For example, a cleanup outlier

works by using computer Z-score function to get rid of the outliers by only keeping the data that is inside certain bounds, or the error function that can use compute RMSE function to find the RMSE values.

High-level functions – these are functions that use one or more low or medium level functions in order to perform a certain task. An example would be that a model training function has to have several functions like one to find random sampled data, a metric function, a model scoring function, and so on.

The last step is to group all of the low and medium level functions that are used for more than a single algorithm into a single python file that can be imported as a module. Then group the other low and medium level functions that can be used for the algorithm into a different file. You need to place the high-level functions into a Python file of their own. This file will dictate each of the steps in the development of your algorithm; from grouping data from various sources to your final model.

There isn't really a fixed rule that you have to follow with the above steps, but it is best that you start these steps and then start to develop your own style once you get a hang of things.

2. Logging and instrumentation

LI are analogous to the black box in all aircraft that records everything that happens in the cockpit. The main purpose for LI is to record any of the useful information from your code while it is being executed. This will help the programmer to debug the code if something goes wrong, and to improve how the code performs, like reducing execution time.

Logging – This records the actionable information like critical failures while the code is running and structured data like the intermediate results that will have to be used later by the code. Multiple log levels like errors, warn, info, and debug are acceptable to see during the testing and development phases. However, these need to be avoided during production.

Logging needs to be minimal and contain only the information that is required for human attention and that needs immediate handling.

Instrumentation – This records all of the other information that was left out during logging that is helpful in validating the code execution steps and to work on performance improvement. In this phase, it is always better to have extra data so that the instrument has as much information as possible.

Validate Execution Steps – You need to record information like steps gone through, intermediate results, and task names. This will help you to validate the results and confirm that the algorithm has gone through all the steps that it should. Invalid results or strange performance of the algorithm may not end up raising a critical error that logging would catch. This is why it is crucial that you record this information as well.

Improve Performance – It's important that you record the time it takes for each subtask, task, and the memory that each variable uses. This will help you to improve your code so that you can make changes to optimize it to run faster and lower memory consumption.

Instrumentation needs to record all of the information that was not logging, and that will help validate that the code's execution steps and work performance is improved. It is always better that you have more data than not enough.

3. Code Optimization

Code optimization means reduced run time and memory usage. The time and space complexity is often shown with O(x), which is also referred to as the Big-O representation. The x is the main term in the time or space taken polynomial. Space and time complexity are the metrics that are used for measuring the algorithms efficiency.

For example, let's assume that there is a nested *for* loop that is size *n* and that it takes around two seconds for it to run, and it is followed by a simple *for* loop that will take around four seconds to run. Your equation for the time consumption would be shown as:

"Time taken ~ 2n^2 + 4n = O(n^2 + n) = O(n^2)"

When it comes to a Big-O representation, you have to drop the non-dominant terms and the coefficients. The scaling factors or coefficients are ignored

because you don’t have as much control over them when it comes to optimization flexibility. It’s important that you remember the coefficient in absolute time taken as it refers to the product of the number of the *for* loops, and the time that it takes for the runs whereas the coefficients within the O(n^2 + n) shows the amount of *for* loops. You need to get rid of the lower order terms. This is why the Big-O for this process is O(n^2).

Your goal is to replace the least efficient parts of your code with a less complex and better alternative. For example, using O(n) is a lot easier than O(n^2). The biggest problem you will face in coding are the *for* loops, and a least common problem that is worse than the *for* loop are recursive functions, meaning (O(branch ^ depth)). You should try to change out the *for* loops with Python functions or modules, which tend to be optimized with C-code that performs the computation, instead of Python so that you get a shorter run time.

4. **Unit Testing**

This will automate code testing when it comes to functionality.

Code has to be able to clear several stages of debugging and testing before it can go into production. There are typically three levels: production, development, and staging. There are some companies where you will find a level before production that will mimic the environment of your production system. You have to make sure that the code is completely free of any obvious problems and it can handle possible exceptions once it goes into production.

In order to find the different issues that could come up you have to test your code against different corner and edge cases, data sets, scenarios, and the like. It’s not efficient for you to carry out all of these processes manually every time you want to test some code. This is the reason you should unit testing will has all of the test case sets and is able to be executed when you need to test your code. It’s important to add in different test cases that will expand your testing results. The unit testing module will work through every test case and will compare the code’s output with what you expected to happen. If the code does not achieve the expected results, the test will fail. It is a pretty good indicator that your code

would end up failing if you put it into production. It's important that you debug your code and then repeat the testing process until your code pasts all of the tests.

Python makes your life a lot easier by using a module called unittest.

5. **Compatibility with ecosystem**

Chances are your code isn't going to be a standalone module or function. You will end up integrating it into your company's code ecosystem, and the code will need to run with the other sections of the ecosystem without experiencing any failures or flaws.

For example, let's assume that you have come up with an algorithm to provide recommendations. This process will normally consist of finding recent data from the company database, updating and generating the recommendations, and storing the data in a database which has to be read by frameworks like web pages to show your recommended items. This works like a chain. You new chain-link has to be able to be locked in with the previous chain links otherwise things will end up failing. Every process has to run the way people expect it to. Every process is going to have to have defined output and input requirements, expected response times, and so on. Whenever your code is faced with a request from other modules for updated recommendations, the code has to be able to return the expected values in a format that is acceptable. If it ends up giving unexpected results, like suggesting milk when the person was looking at stereos, undesired formats, like suggestions that are in text and not pictures, and unaccepted time to respond, this implies that your code isn't working with the system.

The best way to make sure that you avoid this kind of scenario is to discuss with the other people on the team what the requirements are before you start to develop your code. If you don't have a team at your disposal, read your code documentation and the code itself, if you have to, in order to understand all of the requirements.

6. **Version Control**

Git – this is a version control system and is one of the best practices that have come to source code management in recent years. It tracks all of the changes that you make to your computer code. There may be other types of version control or tacking systems, but Git is the one that is most widely used.

In the simplest of terms, it is “modify and commit.” But that may be a bit too simplified. There are a lot of steps to this process like coming up with a branch for development, pushing files to remote branches, pulling files from remote branches, committing changes locally, and many more things, which you should take the time to explore on your own.

Each time that you change your code, instead of saving your file using a different name, you have to commit these changes. This means that you overwrite your old file with the changes that you made along with a key linked to it. Comments are normally written whenever you commit a change to your code.

Let’s assume that you don’t like a change that you made in your last commit, and you want to change your code back to what it previously was. You can easily do this by using the commit reference key. Git is a very powerful and is extremely useful for code maintenance and development.

You may already understand why this is crucial for production systems, and why you have to learn how to use Git. It’s important that you have the flexibility to switch code back to an old version that is stable so that you are prepared if your new version fails.

7. **Readability**

You have to make sure that the code you write is easily understood by others as well, at least when it comes to your teammates. Moreover, you may even find it challenging to understand your code after a few months have passed since you wrote it if you don’t follow proper naming conventions.

Appropriate variable and function names

The function and variable names need to be self-explanatory. When somebody else reads your code, they need to be able to easily figure out what the variables

contain and what the functions do.

It's fine if you have a long name that perfectly lays out what the functionality or role is instead of having a short name like x, y, z, and so on. Those tend to be vague. You should try to keep from exceeding 30 characters for your variable names, and 50 to 60 characters for function names.

The standard code width used to be 80 characters based on outdated IBM standards, which are outdated. Per GitHub standards, the character base is 120.

Setting a page width limit of 1/4th for character names will provide you with 30, which is a good length and doesn't fill up the page. You function names can be a bit longer, but it's best that you don't fill up the whole page, so you if set a limit of ½ of the page width, you will get 60.

For example, if you need to write out the variable of the average age of Asian men, you can write it as *mean_age_Asia* instead of writing *age* or *x*. The same goes for function names s well.

Doc string and comments

Besides having appropriate function and variable names, it's imperative that you place notes and comments when you need to so that they help your readers understand your code.

Doc string – This is module, function, and class specific. It is the first couple of lines in your function definition that describes your functions' role, as well as its' outputs and inputs. You need to add this text between a set of three double quotes.

"def <function_nam>:

"""<docstring>"""

return <output>"

Comments – These can be placed anywhere in your code to tell a reader about the role or action of certain line or section. The need for adding comments will be reduced significantly if you make sure that you give your functions and variables appropriate names. You code should be, at least for the most part, self-

explanatory.

Code review

While this isn't quite a direct step in the process of writing quality production code, having your code reviewed by your peers is helpful in improving your skills in coding.

Nobody is going to write completely flawless computer code unless they have been coding for over ten years. There is always going to be room for some improvement. Chances are, you can find somebody that is better than you. It will all depend on the number of hours a person invests in learning, practicing, and improving their coding skills.

You may find yourself in a situation where you are the most experienced in coding within your team, and you can't allow somebody outside of your team to read your code. In that case, share your code with your team, and ask them for feedback. Even if you are the best coder in your team, other members might find something that you have done wrong. Fresh eyes may be able to catch mistakes. Code review is very important when it happens during the early stage of your career. This step will help to improve your skills as a coder. The following are the steps you should take to make sure that your code is successfully reviewed.

1. Once you have finished writing your code go through debugging, development, and testing. This will help you make sure that you haven't made any mistakes. Then you can ask your peers to review your code.
2. Forward your teammates a link to your code. Make sure you don't ask them to review a lot of scripts at one time. Send them one, once they finish, send them another. The comments they provide you for your first script may work for the others. If applicable, make sure that you make those changes to all of your scripts before you send them another one.
3. Each time you send out an iteration, give them a week or so to read it and test it. Make sure that you give them all of the necessary

information to test your code such as limitations, sample inputs, and so on.

4. Meet everybody and get their suggestions. It’s important to remember that you don’t have to use all of their suggestions. Pick the ones that you believe will make your code better.
5. Repeat this process until you and the team are satisfied. Now you need to fix and improve your code during the first three to four, iterations. Otherwise, it may seem like you don’t know what you’re doing.

The rest of this chapter is going to introduce you to seven programming languages that data scientists can use.

We’ll go deeper in Python in a later chapter, but for now here’s a quick overview.

Python

Python was created by Guido van Rossum in the late 1980s and is an interpreted high-level programming language that is typically used for general-purpose programming.

The design of Python makes it more readable, notably by making use of more whitespace. It gives a programmer the chance to make clear programs on small and large scales. Van Rossum is still the main author of Python.

Python supports structured and object-oriented programming, and many of its features will support aspect-oriented and functional programming as well as metaobjects. There are a lot of paradigms that are supported through extensions, which includes logic programming and design by contract.

Python has dynamic typing, and it uses a combination of cycle-detecting garbage collectors and reference counting to manage memory.

Instead of building all of the functionality into the core, Python was made in a way to make it highly extensible. Having this compact modularity has caused it to be a popular means of adding in interfaces to existing applications that are programmable. Van Rossum wanted to create a small core language with a large library and make an easy interpreter. His desire came from how frustrated he was with ABC, which used a very different approach.

SQL

This is a standard language that is used for manipulating and accessing databases. SQL means Structured Query Language. It became a standard of ANSI in 1986, and ISO in 1987.

SQL is able to:

- Set permissions on views, tables, and procedures.
- Create views in a database.
- Make stored procedures in a database.
- Make new tables in a database.
- Make new databases.
- Delete records from databases.
- Update records in databases.
- Insert records in databases.
- Retrieve data from databases.
- Execute queries against databases.

While SQL is a standard for ANSI and ISO, there are several versions of this language. For it to be compliant with the standard of ANSI, it has to support the major commands like WHERE, SELECT, INSERT, UPDATE, and DELETE.

The Relational Database Management System, RDBMS, is the basis for SQL It also works as the basis for most modern database systems like Microsoft Access, MySQL, Oracle, DB2, IBM, and MS SQL Server. The RDBMS data is stored within tables. These tables are a collection of data entries that are related and made up of rows and columns.

R

R is a programming language and environment used for graphics and statistical computing. It's a GNU project, which works like an S language and environment that was created at Bell Laboratories by John Chambers and his colleagues. R is sometimes seen as different implementation of S. There are a lot of important differences between the two, but for the most part, S code can run unchanged through R.

R gives a user a large variety of statistical and graphical techniques, such as clustering, classification, time-series analysis, classical statistical tests, linear and nonlinear modeling, for example. S language is used more often as a way to research statistical methodology, and R gives you an Open Source route to work with the activity.

One of the biggest strengths of R is the ease with which creating a well-designed plot can be done, including mathematical formula and symbols when needed. R has thoroughly taken care of the defaults for minor choices in the graphics, but the user still has full control.

The R environment includes:

- A well-developed, effective, and simple programming language that includes output and input facilities, user-defined recursive functions, loops, and conditionals.
- Graphical facilities that are needed for data analysis and displays it on-screen or on a hard copy.
- A large, integrated, and coherent collection of intermediate tools for data analysis.
- A suite of operators that are used for calculations on arrays, especially matrices.
- Effective data storage and handling.

SAS

SAS stand for Statistical Analysis System. It is a software suite that was created for advanced analytics, predictive analytics, data management, business intelligence, and multivariate analyses.

SAS was first created at NC State between 1966 and 1976. It was further developed throughout the 1980s and 1990s when new statistical procedures, more components, and JMP were added.

SAS is able to retrieve, mine, manage, and alter data from several different sources, and analyze the information.

In 2002 Text Miner software was added. This software analyzes text data such as patterns in emails. In a free version that was introduced in 2010 for students, media analytics for social media monitoring was added.

Java

Java works as a computer programming language, and is object-oriented, class-based, concurrent, and is designed to use as little implementation effort as possible. It is made so that developers are able to “write once, run anywhere.” This means that Java code can be run on all platforms that support Java without having to recompile it.

James Gosling, at Sun Microsystems, was the original developer of Java. It was originally released in 1995 as one of the core parts of Sun Microsystems. A lot of its syntax is derived from C++ and C, but it doesn’t have very many low-level facilities.

There are five main goals that Java was created to achieve. It must be:

1. familiar, simple, and object-oriented.
2. secure and robust.
3. portable and architecture-neutral.
4. high-performance execution.
5. dynamic, interpreted, and threaded.

Scala

Scala mixes functional and object-oriented programming into a single high-level and concise language. The static types of Scala help remove bugs in complex applications. It also has JavaScript and JVM runtimes, which will allow you to build high-performance systems that have easy access to the large library ecosystem.

The Scala language uses several different features, such as:

- String interpolation
- Implicit classes
- Universal traits and value classes

Scala uses futures, which provides you with a way to reason running operations in parallel that is non-blocking and efficient. Futures work as a placeholder object for a value that you don’t have yet. Typically, the future’s value is provided concurrently and can be used at the same time.

Julia

Julia is another programming language that is high-level and high-performance and is used for numerical computing. It provides a sophisticated compiler, extensive math library, numerical accuracy, and distributed parallel execution. The base library, which is mainly written in Julia, will also use mature and best-of-breed open source Fortran and C libraries to help with string processing, signal process, random number generation, and linear algebra.

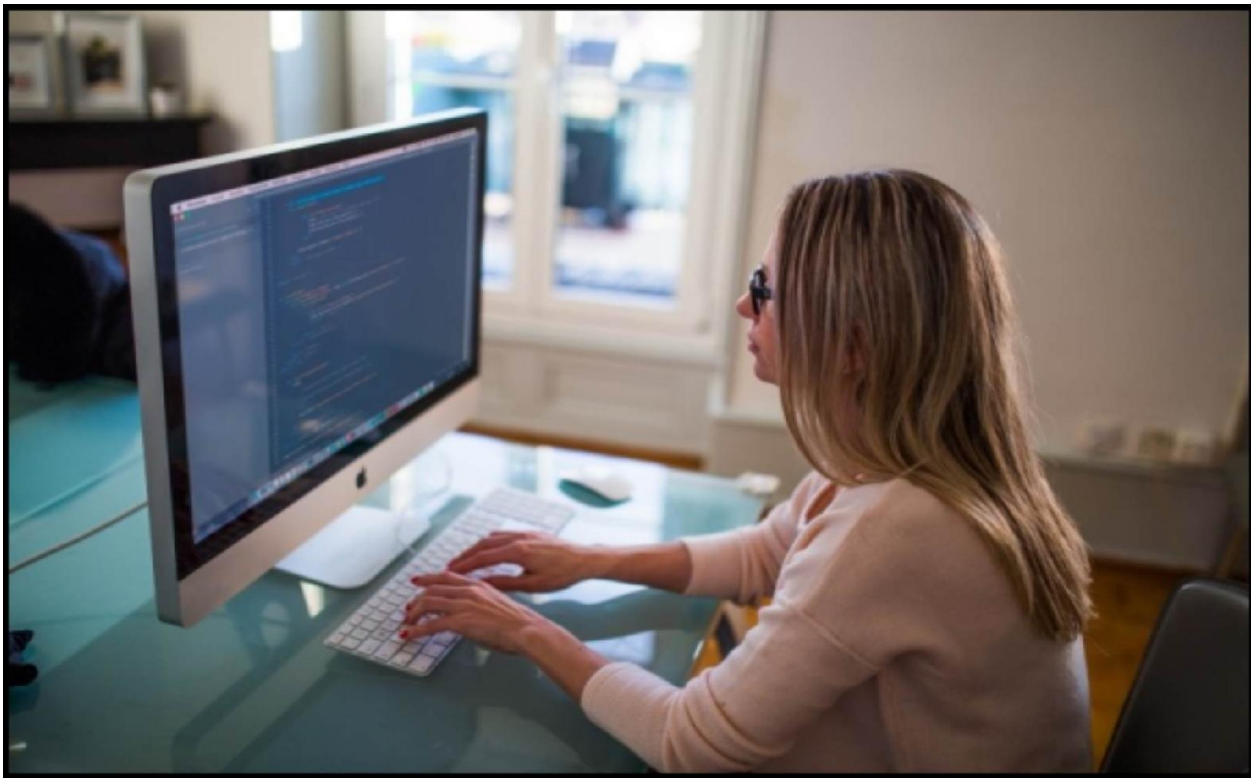
Programs with Julia are created around several dispatches, which will allow user-defined and built-in functions that can be overloaded for several argument combinations.

Some of the main features of Julia include:

- An open source and free MIT license.
- Support for Unicode, which includes UTF-8.
- Extensible and elegant promotions and conversions for numeric and other types.
- Automatic generation of specialized and efficient code for several types of arguments.
- User-defined types which are quick and compact.
- Coroutines.
- It was created from distributed and parallelism computation.
- Powerful shell-like capabilities for managing processes.
- The ability to directly call C functions.
- The ability to call Python functions.
- Lisp-like macros, as well as other metaprogramming facilities.
- A built-in package manager.
- Great performance that is almost like languages such as C
- Dynamic type systems.
- Multiple dispatches.

How to Work with Data

“No great marketing decisions have ever been made on qualitative data.” – John Sculley



Once you have your data, you have to work with it. In this chapter, we will go over the different ways to work with data such as cleaning, munging, rescaling, and manipulating.

Data Cleaning and Munging

Data cleaning means that you get rid of any information that doesn't need to be there and clean up any mistakes. Your approach is going to depend on the data you have gathered, and what your end goal is.

Here is a general guideline on how to clean data.

Does what you're reading make sense?

First, you have to look over your data. If your dataset is fairly large, look at the top 20 rows, the bottom 20 rows, and then another random 20-row sample. The following are some questions that you need ask yourself about the data.

- **Does this data make sense?**
- **Does the data you're looking at match the column labels?**
    - Example: Do you have names in the column labeled name, addresses in your column for addresses, phone numbers in the numbers column? Or do you have different data in the columns.
- **Does your data follow the rules for its field?**
    - Example: Are the names written in alphabetical letters, Josh, or does it contain numbers like?
    - Example: Is the phone number 10 digits, 3331234567, or not, 1234567?
- **After you compute summarizes statistics for your numerical data, does it make sense?**
    - Example: When you have time elapsed data, is your minimum value negative?
    - Example: If you are working with annual wages, is the max value an outlandish number?
- **How many of the values are nulls? Is the number of nulls acceptable? Can you find a pattern to the null values?**
- **Do you see duplicates, and are they okay?**

If you have enough information, you need to correct the values.

Once you have noticed all the problems within your data, start to think about ways to fix it. There are some corrections that will be obvious, such as a name being written like J4osh. You know it should be Josh.

However, there could be corrections that aren't as obvious. Let's say that your data deals with more than 500,000 people and somebody has the name of Brick. You may think the name is a type and that it should say Rick. You could make the change to this if you have the information that it is, in fact, an error, such as you know the person that the data belongs to. But unless you do have a lot of confidence, you probably shouldn't change that type of difference, or you could end up making a typo.

Try to figure out if the data issues have any logic to them and make corrections needed. If your system only allows phone numbers with numbers and no special characters, but your data has phone numbers written like so (333)123-4567, you will be able to use Excel or Python to find and get rid of the dashes and parentheses.

If you are able to, you can use this logic to fill in data for the null values. However, if you're not able to find logic, there are different ways that you could address filling in any missing data. One of the common ways to do this is by using the median or average of the column to fill in the blank spots, which is quite easy but could affect your data distribution. There are also some complicated, but statistically solid, methods to fill in your blank spots, like MICE. You may even want to create a column of metadata that indicates the values that are imputed and the values that are organic. No matter which method you choose, if you have a huge number of nulls for a column, it may be in your best interest to forgo using it in your analysis.

Make use of the tools that makes sense.

When you decide to think about the tool that you need to reach your goal the right tool can end up saving you time from having to do things over and over again. Each tool has its own pros and cons, so here is a bit of a rule of thumb.

You can use Excel when:

- Defining the logical pattern for cleaning your data is hard to figure out, and you have to clean it manually.
- You have a logical pattern in cleaning the data, and you can easily clean it with Excel functions.
- The job needs to be done quick and easy.
- You have less than a million records.

You could use Python or other scripting languages when:

- You have a logical pattern for cleaning, but it would be hard to do with Excel functions.
- The job will be done on a repeating basis.
- You have to document the process.

The tools that you are more comfortable with will also play a role in this.

Start communicating with your source.

If you discover that you can't read something in your data, don't be afraid to call somebody or email them. For example, if you were given data that dealt with information from every county in a state, and you had to have the county names, but all the data gives you are number codes, you should call the person who sent you the data. They can shed some light on how they organized their information. This is even more so true if you are the client of the data source because you are guaranteed to receive clear information. Communicating with them will save you a lot of time and heartache.

This is one of the main, and most basic, ways of cleaning your data, but it isn't the only approach you could take. The more experienced you become, the more creative you can become in cleaning your data.

Data munging, sometimes known as data wrangling, is the process of manually changing or mapping data in one raw form into a different form that will gives you a more convenient use of your data through semi-automated tools.

This could mean that you modify the values into a column in a particular way or grouping several columns together. The need for data munging is typically from

poorly presented or collected data. Data that has been entered manually normally have a lot of errors and the data that is collected through websites are normally optimized to be shown on websites and not aggregated and sorted.

The next steps we're going to go through will be looking at how to mung data. It is important that you don't try to follow along with these steps until you do the Python setup that we will go through in the next chapter. The setup in the next chapter will get you t where you need to be to start these steps.

1. **Look for missing values in your dataset.**

You can continue using your own data set for this, but for the purpose of the tutorial, I will be using random data information so that I can be more direct with my explanation. We're going to start by looking at Cabin. When you first look at the variable, it will leave you with the impression that there may be too many nulls in the set of data. Let's check how many nulls are in the dataset.

"sum(df ['Cabin'].isnull())"

This command will tell you the amount of missing values, or the values that are null. With this, you should get a return of 687, which means there are a lot of missing values and you should drop this variable.

Now we need to look at Ticket. The variable Ticket looks like it has a mixture of text and numbers, and there doesn't seem to be any information. We should drop Ticket too.

"df = df.drop([' Ticket', 'Cabin'], axis=1)"

2. **Filling in the missing age values.**

There are a lot of different ways for you to fill in the missing age values. The simplest would be to replace it with the mean, and can be done like this:

"meanAge = np.mean(df.Age)

df.Age = df.Age.fillna(meanAge)"

The other thing you could do is to create a supervised learning model that could predict ages based on all the other variables, and it would then use age as well as the other variables to predict survival.

Since we are trying to learn data munging, it would be best if we take an

approach that is somewhere between these two extremes. The main hypothesis would that the information in pclass, name, and gender combined is able to give us the information we need to figure out the missing age values.
The following are the steps that you need to take to work this hypothesis: **Step 1: Extract information from Name**
We are going to create a function that will extract the information in Name so that it is written as such: Family_Name, Salutation. First Name
"def name_extract(word):
return word.split(',') [1].split('.') [0].strip()"
With this code, it would change "Jain, Mr. Kunal" to Mr. and "Jain, Miss. Jenika" to Miss. Then we can apply this function so that it will change the whole column.
"df2 = pd.DataFram({ 'Salutation' :df['Name'].apply(name_extract)})"
After we have the salutations, we can look at how they are distributed. We will now use the groupby after we merge the DataFrame df2 with DataFrame df: *"df = pd.merge(df, df2, left-index = True, right_index = True) # merges on index temp1 = df.groupby('Salutation').PassengerId.count() print temp1"*
The output you should get is:
Capt 1
Col 2
Don 1
Dr 7
Jonkheer 1
Lady 1
Major 2
Master 40
Miss 182
Mlle 2
Mme 1
Mr 517

Mrs 125
Ms 1
Rev 6
Sir 1
The Countess 1
dtype: int64
From the information, you can tell that there are four main salutations: Master, Mr, Miss, and Mrs. The other salutations are a lot less common. This means that we will group all of the remaining salutations into one salutation named others. To do this, we can use the same approach that we did to get all of the salutations.
“def group_salutation(old_salutaion):
If old_salutation == ‘Mr’:
Return(‘Mr’)
Else:
If old_salutation == ‘Mrs’:
Return(‘Mrs’)
Else:
If old_salutation == ‘Master’:
Return(‘Master’)
Else:
If old_salutation == ‘Miss’:
Return(‘Miss’)
Else:
Return(‘Others’)
Df3 = pd.DataFram({‘New_Salutation’ :df[‘Salutation’] .apply (group_salutation)}) Df = pd.merge(df, df3, left_index = True, right_index = True) Temp1 = df3.groupby(‘New_Salutation’).count() Temp1
Df.boxplot(column= ‘Age’, by = ‘New_Salutation’)”
You should then receive a list of new salutations that would look like this:
Master 40

Miss 182

Mr 517

Mrs 125

Others 27

Step 2: Create a simple grid

We now want to make a Pivot table that will show us the median values of the class, gender, and age. We will define a function that will give use the values of our chosen cells while also filling in all of the missing ages.

“ table = df.pivot_table(values= ‘Age’, indext=[‘New_Salutation’], columns=[‘Pclas’, ‘Sex’], aggfunc=np.median) # Define function to return value of this pivot_table Def fage(x):

Return table[x[‘Pclass’]][x[‘Sex’]][x[‘New_Salutation’]]

 # Replace missing values

Df[‘Age’].fillna(df[df[‘Age’].isnull()].applu(fage, asix=1), inplace=True)”

This will give you a pretty good way to add in the missing age values.

How should you treat the outliers in the distribution of fare?

As we figured the means of fare match up fairly well with Pclass. However, there are some extreme outliers. There is one point of data that probably grabs your attention, the fare of 512 for class one. Chances are this is an error. There are several ways to change the data. Replace it with the mean or median of class one, or you could also change the value with the second highest value, which will relate closer to the other points of data.

You can decide which one to chose and replace the respective values. The commands work similarly to the ones we’ve already went through.

You now have a set of data that you can use to build a predictive model.

Data Manipulation

Data manipulation is where you take your data and change it to try and make it easier to read, or to make it more organized. For example, you could organize a log of data in alphabetical order, which would make the entries a lot easier to read. Data manipulation is typically used for web server logs so that there are more website owners that can view the most popular pages and traffic sources. People in accounting fields or other jobs that tend to work a lot with numbers will often manipulate data so that they can figure out their costs of products, how well the merchandise is selling each month or week, potential tax obligations, or trends in sales. Stock market analysts will typically use data manipulation so that they can predict trends within the stock market, and how their stocks could end up performing in the future.

Computers are also able to use data manipulation to display information to the users in a way that is more meaningful, based on the software programs code, data formatting, or web page that is defined by the user.

Data Rescaling

When it comes to rescaling data, you will multiply every member of a set of data by a constant j. This means that you transform each number y to f(y), where f(y)=jy, and j and y are real numbers. Rescaling is meant to change the data spread and the position of the data points. What will stay unchanged is the distribution shape and the curve attributes.

How will the variance, standard deviation, and mean change when you rescale data?

The standard deviation, median, and mean are rescaled using the same constant. Your data is multiplied by the scaling constant (y) in order to figure out your new standard deviation, median, or mean.

- F(sd) = y*sd
- F(m) = ym
- F(med) = y*med

The variance (sd^2) is changed by multiplying the squared scaling constant.

- F(sd^2) = y^2 * sd^2

Please note that the z-scores and percentile values aren't changes by rescaling. Since these numbers are calculated through a ratio, the scaling constant ends up getting canceled out.

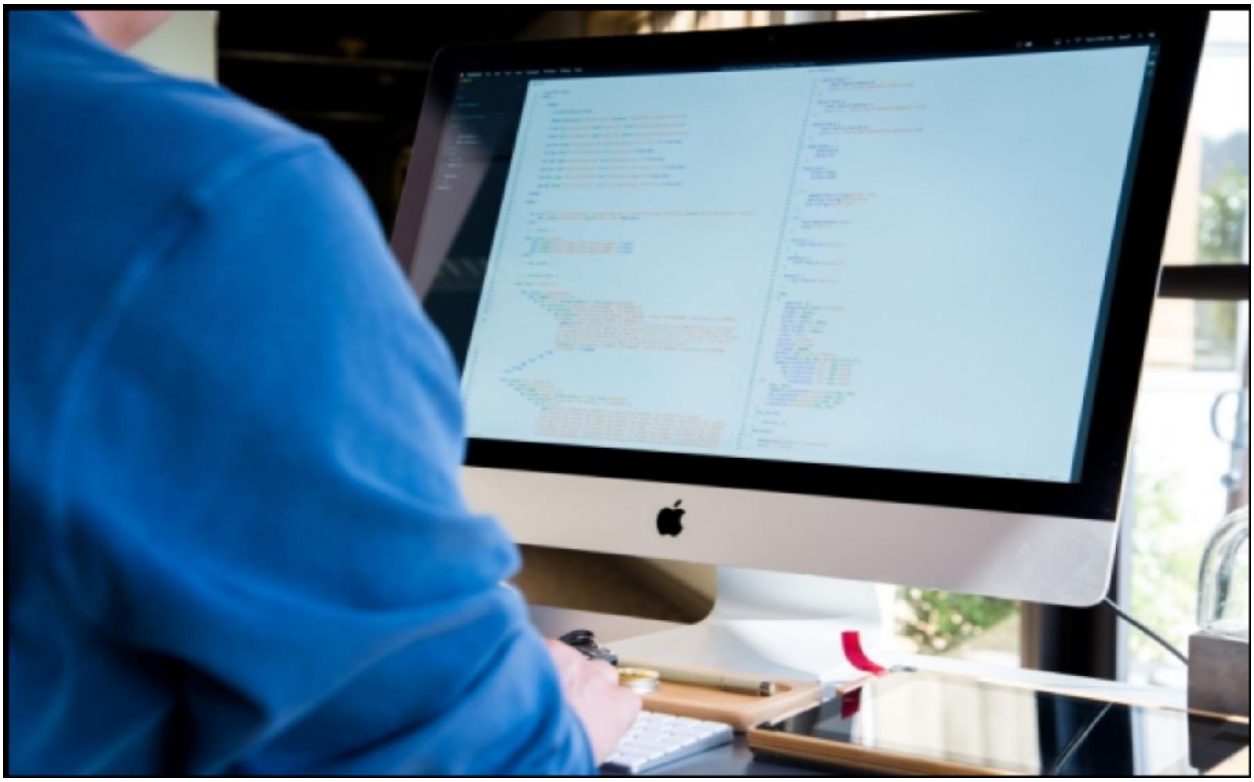
Scaling is typically used if you are looking to change your data measurement types. This could be from millimeters to kilometers or from square inches to acres. You can also use this to compare two different datasets that are typically incomparable because they typically use different scales.

Here's an example: Let's assume that teacher Y grades their students using a 100-point system, which teacher Z, who teaches the same subject, grades their students using a 170 point system. We can assume that the difficulty of the homework and tests are standardized between both classes. To figure out which one of the classes are doing better, you could rescale class Y by mapping the members of the set of data a to f(a) where f(a)=1.7a.

- To change a student's grade in class Y of a 99 to class Z's grad, you would take 99*1.7= 153.
- If there were a mean of 86, it would change to 86*1.7= 146.2.
- You could rescale a standard deviation of 15 to 15*1.7= 25.5.
- If a student had grades that put them in 99th percentile before you scaled it, they would still have a grade in the 99th percentile after you scaled it.

Python

“Abstraction is one of those notions that Python tosses out the window, yet expresses very well.” – Gordon McMillian



Python has become very popular language for data analysis. To help you see why it is such a popular option, here are a few points:

- It could end becoming the common language for data science and for the production of analytics products in a web-based setting.
- Extremely easy language to learn.
- Has a great online community.
- It is open source and free to install.

That being said, it does have its drawbacks:

- Its language is interpreted and not complied. This means that it could end up taking up more CPU time. However, because it saves time during programming by being easy to learn, it is still the better

option.

One of the most debated topics when it comes to Python is between version 2.7 or 3.4. If you are a beginner, you will come across this debate. When you get down to it, there isn't actually a right or wrong choice. It will depend on what you need and your situation. We'll go through some pointers of each to help you figure out which you would prefer.

Python 2.7 has:

- A great support community. This is especially important when you are just starting out. This version of has been in use for over 15 years.
- There is a large number of third-party libraries available. Even though most of the libraries have been made to support 3.x, there are still a large number of modules that will only work with the 2.x versions. If you think you will be using Python for any specific applications such as web-development where you would be using a lot of external modules, you would be better sticking with 2.7.
- There are some features of 3.x versions that have backward compatibility so that they are able to be used with 2.7.

Python 3.4 has these benefits:

- It is faster and cleaner. The developers of Python have fixed some of the glitches and a few drawbacks so that it can create a stronger foundation for future use. Initially these things may not seem that relevant, but eventually, they will become important.
- Python. 2.7 was the last version of the 2.x Python family. Eventually, you will have to change to the 3.x versions. Python 3 has introduced stable versions over the last five years, and they are going to continue doing this.

There isn't really a clear winner of the two, but the more important thing is that you should learn the Python language.

Installing Python

There are two ways that you can install Python:

1. Python can be downloaded directly from the project site, and then you can install the individual libraries and components you want.
2. You could also install and download a package, which will come with libraries already installed. One option would be to download Anaconda. You could also use Enthought Canopy Express.

The second method will give you a hassle-free installation, so it is the best option for beginners. The problem with doing it this way is that you will have to wait for the complete package to be upgraded, even if you just want to get the latest the version of one of the libraries. This wouldn't be a problem unless you start performing cutting-edge statistical research.

After you have installed Python, you will find several options for picking your environment. There are three main common options:

1. iPython notebook, which is similar to markdown in R.
2. IDLE, which is the default environment.
3. Terminal or Shell based.

Picking your environment will depend on what it is that you will need. I recommend iPython Notebooks. It will provide you with a lot of amazing documenting features while you are writing your code, and you can run the code in blocks instead of line by line. For the rest of this chapter, I will be referring to iPython environment.

To start out with you can use Python as a calculator like this: *" In [1]: 2 + 3 Out [1] : 5 "*

Let's go over a few notes:

- The iPython notebook can be started by typing "ipython notebook" in your terminal/cmd, depending on your OS.
- Name your iPython notebook by selecting the name – UntitledO.
- The interface will give you In [*] for the input and Out [*] for the

output.

- A code can be executed by hitting “ALT + Enter” or “shift + enter” if you want add in another row.

Before we really get into the problem-solving aspects of Python, let’s step back a bit and make sure that you understand all of the basics. Conditional constructs, data structures, and iteration form the crux of all programming languages. When it comes to Python, it includes if-else, while-loop, for-loop, dictionaries, tuples, strings, lists, and so on. Let’s go through a few of these.

Python Libraries and Data Structures

The following are some of the data structures that you can use in Python. It is important that you are familiar with these so that you can use them in the appropriate way.

- Lists – This is one of Python's most versatile data structures. You can simply define a list by writing out commas that are separated by different values in square brackets. The lists could have different types of items, but usually, the items will be the same type. The lists in Python are individual and mutable elements of a list can to be changed.

The following are list examples and how to access them.

" In [1] : squares_lists = [0, 1, 4, 9, 16, 25]
In [2] : square_list
Out [2] : [0, 1, 2, 9, 16, 25]
In [3] : squares_list[0]
Out [3] : 0
In [4] : squares_list [2 : 4]
Out [4] : [4, 9]
In [5] : squares_list [-2]
Out [5] : 16 "

- Strings – You can define a string with single, double, or triple inverted commas. Strings that are enclosed with triple quotes are able to span over several lines and are often used within docstrings, which is Python's way to document functions. A backslash is used as an

escape character. These strings in Python are often immutable, so you are unable to change up the parts of the string.

“ In [6] : greeting = ‘Hello’
Print greeting[1]
Print len(greeting)
Print greeting + ‘World’
In [7] : stmt = r ‘\n is a newline character by default. ‘
Print stmt
In [8] : greeting[1 :] = ‘1’ “

- Tuples – You can show a tuple by a group of values that are separated by commas. These are also immutable, and their outputs will be encompassed by parentheses. This is so that the nested tuples will be correctly processed. Even though these are immutable functions, they are still able to house mutable data if you need them to.

Because tuples can’t be changed, and they are immutable, it makes them a lot faster when they are processed, especially if they are compared to lists. This means that if it is unlikely for your list to change, you should think about using tuples instead of using a list.

“ In [9] : tuple_example = 0, 1, 4, 9, 16, 25
In [10] : tuple_example
Out [10] : (0, 1, 4, 9, 16, 25) In [11] : tuple_example[2]
Out [11] : 4
In [12] : tuple_example[2] = 6 “

- Dictionary – This is an unordered group of key: value pairs. It requires that your keys are unique, which means within a single

dictionary. Using a pair of braces with make and empty dictionary.

“ In [20] : extensions = (‘Dale’ : 9073, ‘Page’ : 9128, ‘Small’ : 9223, ‘Nate’ : 9330) extensions Out [20] : { ‘Dale’ : 9073, ‘Nate’ : 9330, ‘Page’ : 9128, ‘Small’ : 9223}
In [21] : extensions (‘Matt’) = 9150 extensions Out [21] : {‘Dale’ : 9073, ‘Matt’ : 9150, ‘Page’ : 9128, ‘Small’ : 9223, ‘Nate’ : 9330}
In [22] : extensions.keys {}
Out [22] : (‘Small’, ‘Page’, ‘Dale’, ‘Matt’, ‘Nate’) “

Conditional and Iteration Constructs

Like the majority of other programming languages, Python uses a FOR-loop, which is one of the most commonly used methods for iteration. The syntax is simple.

“ for I in [Python Iterable]:

Expression(i) “

In the section that says python iterable, you can fill in a tuple, list or any other data structures. Here is an example that will give you the factorial of a number.

“ fact=1

For I in range(1, N + 1):

*Fact *= i “*

When it comes to a conditional statement, you will use these to execute code fragments that are based upon a condition. The if-else is the most commonly used construct, and you can use this syntax: *“ if [condition]:*

*_execution if true_*

Else:

execution if false “

You could use this syntax if you want to print out whether the number S was odd or even.

“ if S%2 == 0:

Print ‘Even’

Else:

Print ‘Odd’ “

Now that you have a decent understanding of some Python fundamentals, we can dive a little be further. Think about having to perform these types of tasks:

1. Access web-pages
2. Make statistical models
3. Plot histograms and bar charts
4. Find the root of quadratic equations

5. Multiply two matrices

If you are trying to come up with code from scratch, it will end up becoming a nightmare, and you will end up leaving Python after a day or two. But we're not going to worry about those things at the moment. There are a lot of predefined libraries that you can directly add to your code and make things a lot easier. Here's an example, if you take our factorial example from earlier, that can be changed into a single step.

" math.factorial (S) "

In order for that to work, you will have to make sure that you have the math library. Now would be a good time to explore the libraries.

Python Libraries

This first thing you have to know how to do when it comes to libraries is to import them into your environment. There are a few different ways you can do this.

“ import math as m

*From math import * “*

With the first line, you will have defined a name m to library math. It is now possible to use different functions from your math library by using the name that you gave it: m.factorial().

In the second line, you will import the whole name with math, meaning that you are able to directly use factorial() without having to refer to math.

The following libraries are those that you will need for data analysis and scientific computations:

- NumPy – This stands for Numerical Python. The best feature that comes with NumPy is n-dimensional array. This library will give you some basic linear algebra functions, advanced random number capabilities, Fourier transforms, and applications for integrating with other low-level programming languages such as C++, C, and Fortran.
- SciPy – This stands for Scientific Python. This is an add-on to NumPy. This is an extremely useful library when it comes to several different high-level engineering and science modules such as Sparse and optimization matrices, linear algebra, and discrete Fourier transform.
- Matplotlib – This is used to plot lots of different graphs, beginning with histograms to heat plots to line plots, and so on. With ipython notebook, you can use the Pylab feature to access these plotting features. If you don’t pay attention to the inline option, then the pylab function will change your ipython environment to something similar to Matlab. Latex commands are also able to be used to add math into

your plot.

- Pandas – This is used for manipulations and structured data operations. This is used when there is a lot for data preparation and munging. This is still a fairly new addition and has played a big part in boosting Python's usage among the data science world.
- Scikit Learn – This is used for machine learning. This was built onto matplotlib, NumPy, and Scipy. This library has plenty of helpful tools for statistical modeling and machine learning, which includes dimensionality reduction, clustering, regression, and classification.
- Statsmodels – This is used for statistical modeling. This is a Python module that will let its users explore data, perform statistical tests, and estimate statistical models. You can find a large list of things like result statistics, plotting functions, statistical test, and descriptive statistics, and these can be used for different data types.
- Seaborn – This can be used for data visualization. This library helps to make informative and attractive statistical graphics. Matplotlib is the basis of this library. It aims to make visualization an important part of understanding and exploring data.
- Bokeh – This is used to create interactive plots, data applications, and dashboards on regular web browsers. It gives the user the power to create concise and elegant graphics in the D3.js style. It also has the ability for high-performance interactivity with streaming or large datasets.
- Blaze – This is for improving the capabilities of Pandas and Numpy to stream and distribute data sets. This can be used to gain access to data from several different sources. Combined with Bokeh, Blaze is able to act like a powerful tool to create effective dashboards and visuals on huge amounts of data.
- Scarpy – This is for web crawling. This is a helpful framework for getting data set patterns. It can begin at a home URL and then dig

throughout the pages in the site to find information.

- SymPy – This is for symbolic computation. This library has a lot of capabilities that range from calculus to basic symbolic arithmetic, quantum physics, discrete mathematics, and algebra. Another one of its useful features is its ability to format the results of LaTeX code computations.
- Requests – This is used to access the web. This works a lot like the standard library of Python urllib2, but it makes things a lot easier to code. There are some small differences with urllib2, but for the beginner, requests tends to be more convenient.

Exploratory Analysis with Pandas

Panda is a very useful data analysis library for Python. It has played a large part in upping the use Python among the people in the of data science community. Now we are going to use Panda to read our first set of data. Before we start to work with the data, you need to understand that there are two data structures that Pandas has, DataFrames and Series.

You can understand Series as a one dimensional indexed and labeled array. This can be used to access individual elements of your series with labels.

Dataframes work very similar to an Excel workbook. You will see column names which refer to the columns, and you will see rows that you can access by using row numbers. The main difference between the two is that the row numbers and column names are referred to as row and column index when it comes to dataframes.

Dataframes and series are what create the core data model for Panda. Your set of data will first be read into the dataframe, and then you can perform various operations to the columns.

At this point, you need your set of data. This can be any data set that you want to play around with on Python.

To start out with, run the iPython interface in Inline Pylab mode with this prompt that you will type into your terminal or windows command line.

“ ipython notebook - - pylab = inline “

This will give you an iPython notebook that is open in a pylab environment, which will provide with some useful libraries. Here you can also plot your data inline. This is why this is such a great environment for interactive data analysis.

In order to figure out if your environment has been loaded correctly, you can type on this command:

“ plot(arrange(5)) “

In this tutorial you will be using these libraries:

- Pandas

- Matplotlib
- Numpy

You will not have to import the numpy and matplotlib libraries because you are using the pylab environment. I would still place them in my code in case you choose to use a different environment.

Once you have imported your library, you will need to read your set of data by making use of the read_csv() function. This is how it should look:

“ import pandas as pf

Import numpy as np

Import matplotlib as plt

Df = pd.read_csv(“enter the location of your data here”) “

After your set of data has been read, you can look at a few of the top rows by using the following code:

“ df.head(10) “

This will give you a print out of ten rows. You can look at as many rows as you want to by printing your set of data.

Next, we will take a look at the summary of numerical fields with this function:

“ df.describe() “

This function will give you the standard deviation, mean, count, max, quartiles, and min in its output. Let’s say that we get an output of information that looks like this:

1. Loan amount is 614 to 592 with 22 missing values.
2. Loan amount term is 614 to 600 with 14 missing values.
3. The credit history is 614 to 564 with 50 missing values.
4. There are around 84% of applicants that have credit history because the mean of the credit history is 0.84.
5. The distribution of applicant income looks to be in line with expectations. The same is true for co-applicant income.

With the information that you get from this output, you will be able to tell if there is a skew in your data by looking the mean and the median. When you

have non-numerical values, you can look towards the frequency of the distribution to figure out if it makes sense. You can print out a frequency table by using this command:

“ def[‘fill in name’].value_counts() “

You can use dfname[‘column_name’] as a basic way to index in order to access a certain column of your dataframe. You can even view a list of columns.

Now we can look at the distribution of different variables since we have a basic understanding of data characteristics. Begin by picking something with a numeric variable in your data to work with.

You can plot a histogram for your numerical value with:

“ df[‘variable name’].hist (bins=50) “

With a histogram, you will be able to tell if there are any extreme values. If this isn’t clear in your data, you can change the bin value so that you can see the distribution more clearly. Now you can look at the information in a box plot so that you can understand the distribution a bit better.

“ df.boxplot (column= ‘variable name’) “

This will confirm for you if there are extreme values or outliers. If you are dealing with variables like salary and education, these two factors may be the reason why you have such outliers. This is where you can create another box plot that separates the two values, such as education and salary.

“ df.boxplot (column = ‘variable name’, by = ‘variable name’) “

This view should show you if there is a difference between the two variables. This will help you when it comes to analyzing your data.

Now we can start to look at the category variables in depth. For this, we are going to use an Excel-style cross-tabulation and pivot table.

“ temp1 = df[‘variable name’].value_counts(ascending=True)

Temp2 = df.pivot_table(values= ‘Loan_Status’, index= [‘variable name’], aggfunc=lam

Bda x: x.map({ ‘Y’ : 1, ‘N’ : 0}). Mean())

Print ‘name of your table:’

Print temp1
Print ‘\n name of your second table:’
Print temp2 “
You should get a pivot table that is similar to Microsoft Excel. You can also plot this information like a bar chart using the following code for matplotlib.
“ import matplotlib.pyplot as plt
Fig = plt.figure(figsize = (8, 4))
Ax1 = fig.add_subplot(121)
Ax1.set_xlabel (‘ variable label’)
Ax1.set_ylabel(‘ variable label’)
Ax1.set_title(“variable name”)
Temp1.plot (kind= ‘bar’)
Ax2 = fig.add_subplot(122)
Temp2.plot (kind = ‘bar’)
Ax2.set_xlabel(‘ variable label’)
Ax2.set_ylabel(‘ variable lable’)
Axe.set_title(“variable name”) “
This will show you the relationship between your selected variables and how one affects the other. For example, if we were looking at the chances of a person getting a loan, you would like to get a plot that tells you that the odds of them getting a loan improve if they have a valid credit history.
You can also combine this information into a stacked chart.
“ temp3 = pd.crosstab (df[‘ variable label’], df[‘variable label’])
Temp3.plot (kind= ‘bar’, stacked=True, color= [‘red’, ‘blue’], grid=False) “
You have now created two basic classification algorithms, one that is based on a single variable, and another with two stacked variables.
You also know how to do an exploratory analysis with Python using the Pandas library. Hopefully your love for the Pandas library has grown, and hopefully, you can see how Pandas can help you analyze your sets of data.

Creating a Predictive Model

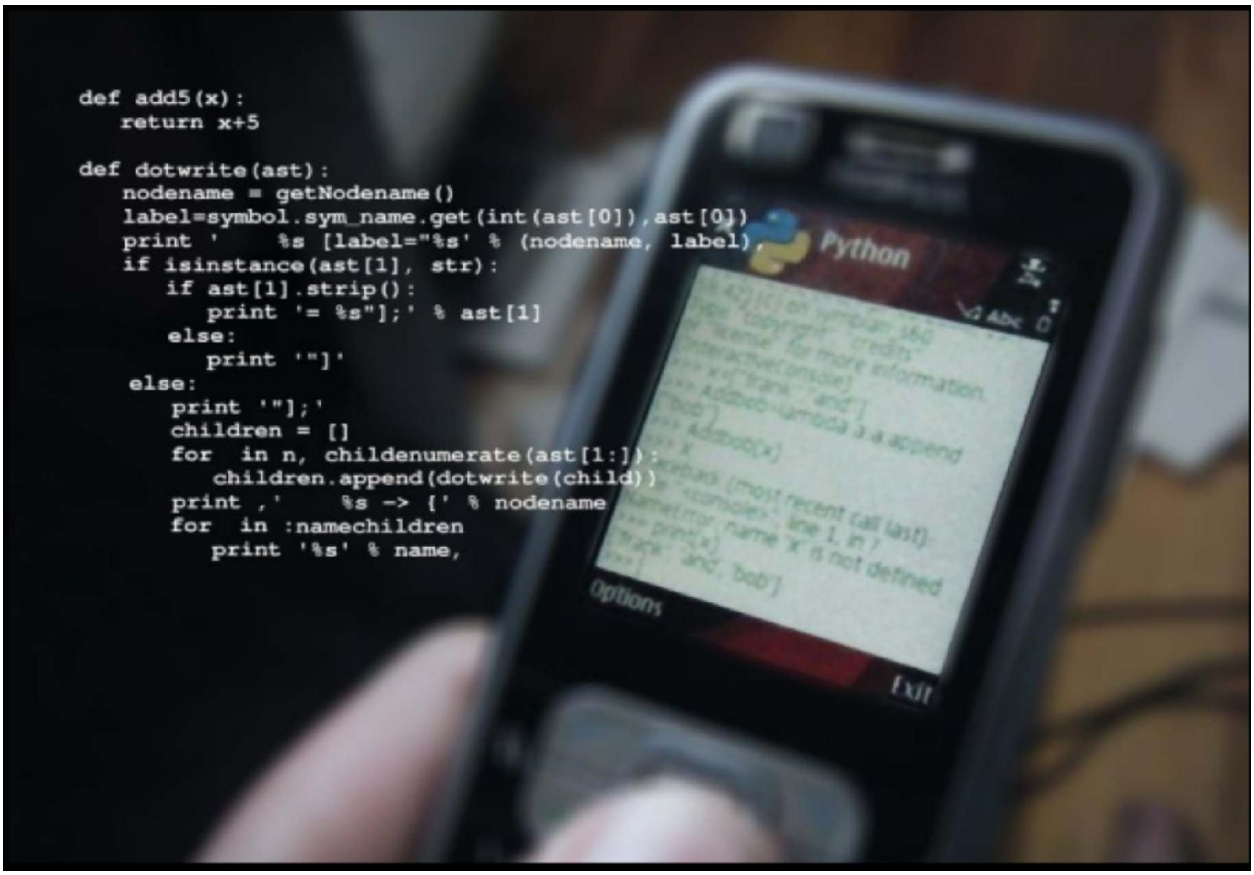
You should do the data munging chapter before you start creating your predictive model. You want to make sure that it is useful data. Skicit-Learn is the most popular library for creating a predictive model. This library requires that all of your inputs be numeric. This means you need to change all of your variables into numerals with this:

“ from sklearn.preprocessing import LabelEncoder
Var_mod = [‘labels of non-numeric variables separated with ‘ ‘, ‘ ‘]
Le = LabelEncoder()
For I in var_mod:
Df[i] = le.fit_transform(fd[])
Df.dtypes “

Now you will import the modules that you need. Then you will have to define a classification function that will take the model input to figure out the cross-validation and accuracy scores.

“ from sklearn.linear_model import LogisticRegression
From sklearn.cross_validation import KFold
From sklearn.ensemble import RandomForestClassifier
From sklearn.tree import DecisionTreeClassifier, export_graphviz
From sklearn import metrics
Def classification_model(model, data, predictors, outcome):
Model.fit(data [predictors], data [outcome])
Predictions = model.predict(data [predictors])
Accuracy = metrics.accuracy_score(predictions, data[outcome])
Print “accuracy : %s” % ‘{0: .3%}” .format(accuracy)

Kf = KFold(data.shapre [0], n_folds=5)
Error = []
For train, test in kf:
Train_predictors = (data[predictors]. Iloc[train, :])
Train_target = data [outcome].iloc[train]
Model.fit(train_predictors, train_target)
Error.append(model.score(data [predictors]. Iloc [test, :], data [outcome].iloc [test]))
Print "Cross-Validation Score : %s" % "{0: .3%}" .format(np.mean(error))
Model.fit(data[predictors], data[outcome]) "



Now we can now make a logistic regression model. One way to do this is to take all of the variables in the model, but this could end up causing problems. You

should come up with some intuitive hypothesis based on what you know about your data. If we go back to the loan example, your odds of being approved increase if:

1. **You have a good credit history.**

2. **You and the co-applicant have high incomes.**

3. **You have a higher education level.**

Now you can make a model.

“ outcome_var = ‘variable label’
Model = LogisticRegression()
Predictor_var = [‘variable label’]
Classification_model (model, df, predictor_var, outcome_var) ”

You can also add in more variables. When you do this, you can expect the accuracy to increase, but this tends to be more challenging. The cross-validation and accuracy scores aren’t going to be impacted very much by the less important variables.

You can also create a decision tree. A decision tree will typically provide you with a more accurate reading than the logistic regression model.

“model = DecisionTreeClassifier()
Predictor_var = [list of variables separated by ‘ ‘ , ‘ ‘]
Classification_model(model, df, predictor_var, outcome_var) “
Lastly, we can make a random forest which helps to solve classification problems.

“ model = RandomForestClassifier(n_estimators=100)
Predictor_var = [list of variable names separated by ‘ ‘, ‘ ‘]
Classification_model(model, df, predictor_var, outcome_var) “
If you end up getting an accuracy score of 100%, the chances are you have overfitted, and you can fix this in two ways.

1. **Lower the number of predictors.**

2. **Tune your parameters.**

Try only using the top five variables for your model, change the parameters just a bit.

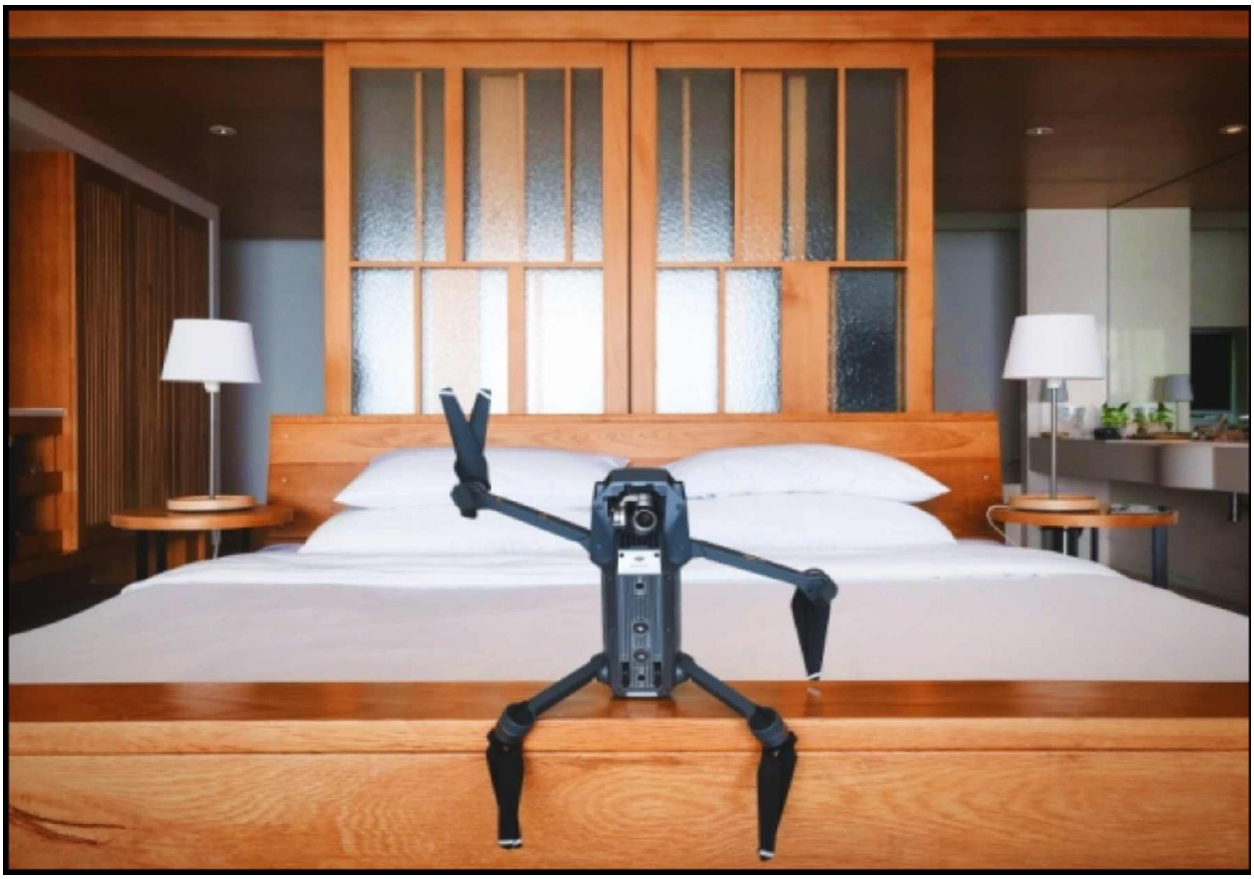
“ model = RandomForestClassifier(n_estimators=25, min_samples_split=25, max_depth=7, max_features=1)
Predictor_var = [variable labels separated by ‘ ‘, ‘ ‘]
Classification_model(model, df, predictor_var, outcome_var)

You should see your accuracy score going down, and your cross-validation score improve. This means that you model is generalizing well. You can’t really repeat random forest models. Different runs will give you slightly different variations because of the randomization. But all of your outputs should still be in the ballpark range.

Machine Learning and Analytics

“Without big data, companies are blind and deaf, wandering out onto the web like deer on a freeway.” – Geoffrey Moore

When it comes to data science, you will likely hear the words machine learning and data analytics a lot. There are a lot of people that get these terms confused, and they’re not sure which is which. Here we are going to look at how machine learning and data analytics compares to data science.



Machine learning is the practice of using algorithms to learn from data and forecast possible trends. The traditional software is combined with predictive and statistical analysis to help find the patterns and to get the hidden information that was based upon the perceived data. Facebook is a great example for machine learning implementation. Their machine learning algorithms collect information for each user. Based upon a person’s previous behavior, their

algorithm will predict the interests of the person and recommend notifications and articles on their news feed.

Since data science is a broad term that covers several disciplines, machine learning works into data science. There are various techniques used in machine learning such as supervised clustering and regression. But, the data that is used in data science may not have come from a machine or any type of a mechanical process. The biggest difference is that data science covers a broader spectrum and doesn't just focus on statistics and algorithms but will also look at the entire data processing system.

Data science can be viewed as an incorporation of several different parent disciplines, including data engineering, software engineering, data analytics, machine learning, business analytics, predictive analytics, and more. It includes the transformation, ingestion, collection, and retrieval of large quantities of data, which is referred to as Big Data. Data science structures big data, finding the best patterns, and then advising business people to make the changes that would work best for their needs. Machine learning and data analytics are two tools of the many that data sciences use.

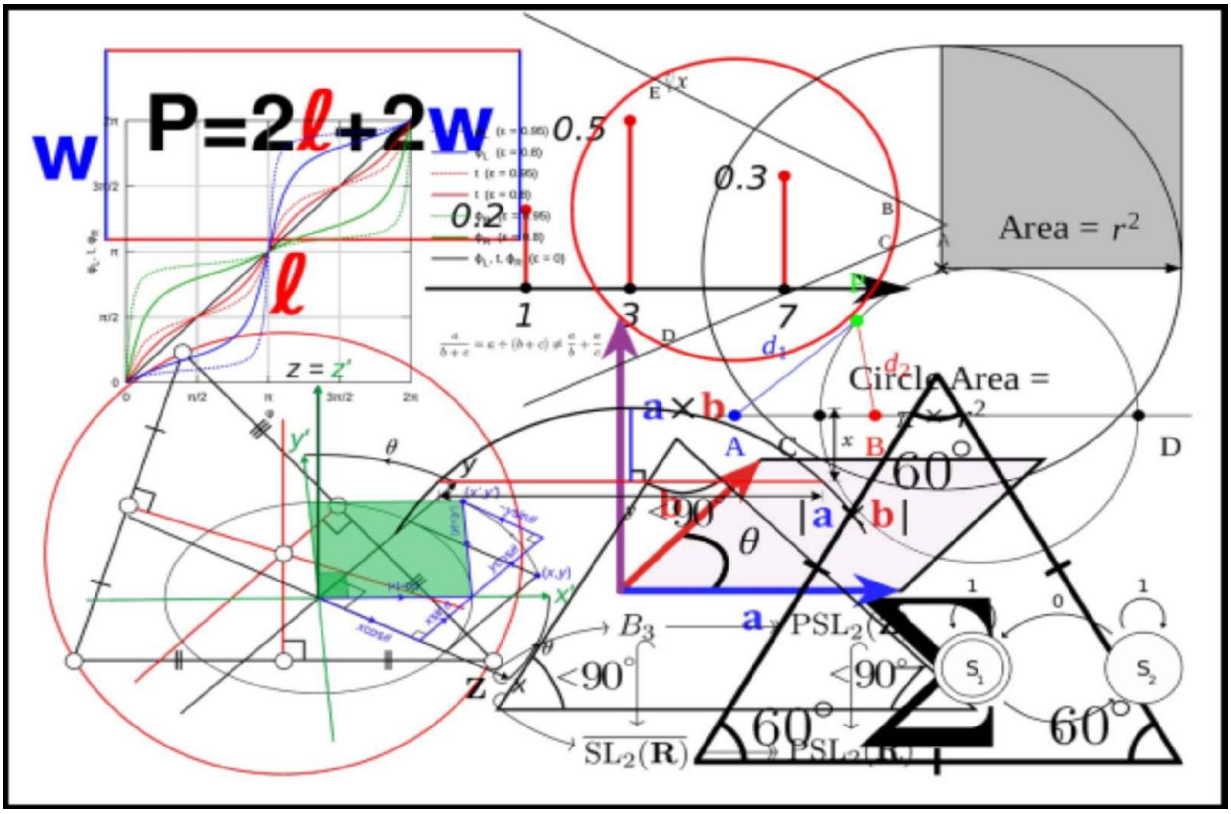
A data analyst is someone who is able to do basic descriptive statistics, communicate data, and visualize data. They need to have a decent understanding of statistics, and good understanding of databases. They need to be able to come up with new views and to perceive data as a visualization. You could even go as far as to say that data analytics is the most basic level of data science.

Data science is a very broad term that encompasses data analytics and other several related disciplines. Data scientists are expected to predict what could happen in the future using past patterns. A data analyst has to extract the important insights from different sources of data. A data scientist will create questions, and the data analyst will find the answers to them.

Machine learning, data science, and data analytics are some of the most booming areas of employment at the moment. Having the right combination of skills and experience could help you get a great career in this trending arena.

Linear Algebra

“My favorite subjects at school were algebra and logic: making a big problem into something small.” – Mario Testino



Linear algebra is a mathematics branch that handles vector spaces. It underpins a huge amount of data science techniques and concepts, which means that it is important learn as much as possible.

Vectors

Vectors are objects that you can add together to make new vectors, and they can be multiplied by scalars to make new vectors as well. Vectors are points located in a finite space. While you may not view your data as a vector, they are great ways to represent numeric information.

If you are dealing with ages, heights, and weights of a large group of people, you could treat this data like three-dimensional vectors: age, weight, height. If you are teaching a class that has four exams throughout the semester, you could treat these grades like a four-dimensional vector: test1, test2, test3, test4.

One of the easiest from-scratch approaches is to show your vectors as a number list. This list of three numbers will correspond to a single vector in your three-dimensional space, and so on:
“ height_weight_age = [70,
170,
40]
Grades = [95,
80,
75, 62] “

A problem that comes with this approach is that you are going to want to do arithmetic on your vectors. Since Python lists don’t work as vectors, and as such don’t give you any tools for vector arithmetic, you will have to create these types of tools yourself. Let’s see how that would work.

To start out, you will need to add in two vectors. Vectors will add component-wise. This means that if there are two vectors, a and b, and they have the same

length, they have a sum that has a first element of a[0] + b[0], and a second element of a[1] + b[1], and so on. If they don't have the same length, then they can't be added in.

If you were to add in the vectors [2, 3] and [3, 2] you would get [2 + 3, 3 + 2] or [5, 5].

This can easily be used by zipping all of the vectors together and then make use of a comprehension to add in all of the corresponding elements.

" def vector_add(a, b):
Return [a_i + b_i
For a_i, b_i in zip (a, b)] "

In a similar manner, you can subtract your two vectors by getting rid of the corresponding elements.

" def vector_substract(a, b):
Return [a_i – b_i
For a_i, b_i in zip (a, b)] "

There may be times when you need to sum a vector list. This means that you will want to make a new vector which is the sum of all the first elements, and the second vector is the sum of the second elements, etc. The simplest way for you to figure this out is to add a vector at a time
" def vector_sum (vectors) :
Result = vectors[0]
For vector in vectors [1:] :
Result = vector_add(result, vector) Return result "

When you really think about what we are doing, we are only reducing the vector list with vector_add. This means that we are able to rewrite this using higher-order function, such as:

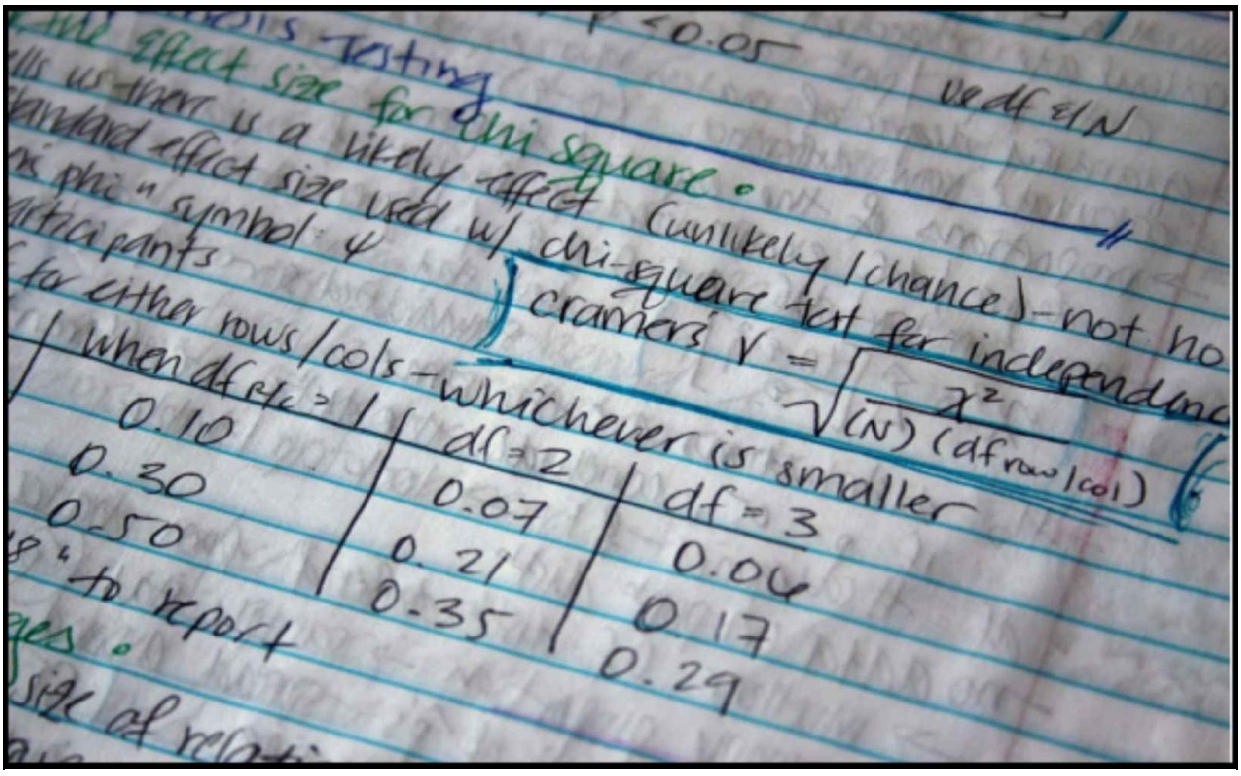
“ def vector_sum(vectors) :

Return reduce(vector_add, vectors) “

Or you could:

“ Vector_sum = partial(reduce, vector_add) “

This last one is probably cleverer instead of helpful. Next, you will also have to multiply your vector by your scalar, which can be done by multiplying every vector element by this number.



“ def scarlar_multiply (c, a):

*Return [c * a_i for a_i in a] “*

This is going to give you the ability to compute the componentwise means of your same-sized vector lists.

“ def vector_mean(vectors):
N = len(vectors)
Return scalar_multiply(1/n, vector_sum (vectors)) “

One of the lesser known tools is to use the dot product. This product is created through the sum of two vectors and their componentwise products.

“ def dot(a, b):
*Return sum(a_i * b_i*
For a_i, b_i in zip(a, b)) “

This product will measure how far vector a will extend in vector b’s direction. One example would be if b = [1, 0] then dot (a, b) is only the first element of a. A different way to do this is by saying it is the length of the vector you would see if you were to project point a on point b.

When you use this, it is quite easy to find a vector’s sum of squares.

“ def sum_of_squares (a):
Return dot (a, a) “

And this can then be used to figure out the length or magnitude
“ import math
Def magnitude(a):

Return math.sqrt(sum_of_square(a)) “

At this point you now have the pieces you need to figure out space between your two vectors, as you can see in this equation:

$$\sqrt{(a_1 - b_1)^2 + \cdots + (a_n - b_n)^2}$$

“ def squared_distance(a, b) :
Return sum_of_squares(vector_subtract (a, b)) Def distanc(a, b) :
Return mathsqrt(squared_distance(a, b)) “

You can write the equivalent to get a clearer image of what we’re looking at.

“ def distanc(a, b) :
Return magnitude(vector_substract (a, b)) “

This is a pretty good amount of information to help you get started with vectors. It’s important that you take the time to study them even further if you are still unsure how it works.

Note: When it comes to using lists as vectors, it works well for exposition, but not for performance. When it comes to production code, you want to use NumPy library. This library includes all of the high-performance array class, and it includes all of the arithmetic operations.

Matrices

A two-dimensional set of numbers is considered a matrix. The matrices will be represented as lists of lists. Each of the inner lists will have the same size and will represent a matrices row. If K is a matrix, then K[c] [d] would be the element in the c row and d column. Mathematical convention dictates that matrices are represented by capital letters. You can see this here:

“ K = [[1, 2, 3],
[4, 5, 6]]
L = [[1, 2],
[3, 4],
[5, 6]] “

Note: When it comes to mathematics, the first row of a matrix would be labeled “row 1” and the first column would be names “column 1.” Since we are writing our matrices as Python lists, which are indexed at zero, our first matrix row will be labeled “row 0” the first column will be labeled “column 0.”

Since we are using list-of-lists representation, our matrix K will have len(K) rows and len(K[0]) columns. Here we will consider the shape.

“ def shap(K) ;
Num_rows = len(K)
Num_cols = len(K[0]) if K else 0
Return num_rows, num_cols “

If you have a matrix with c rows and d columns, it is called c X d matrix. You are able to view the rows of a c X d matrix as length c’s vector, and every column is the vector length d.

“ def get_row(K, c) :
Return K[c]
Def get_column (K, d):
Return [K_c [d]
For K_c in K] “

You will also want to make a matrix based on the shape and a function to create the elements. The can be done through a nested list comprehension.

“ def make_matrix(num_rows, num_cols, entry_fn) : Return [[entry_fn (c, d)
For d in range (num_cols)]
For c in range (num_rows)] “

By using this function, you can create a five by five identity matrix that has a 1s on the diagonal and elsewhere would be a 0s.

“ def is_diagonal (c, d) :
Return 1 if c == d else 0
Identity_matrix = make_matrix (5, 5, is_diagonal)
These matrices end up being important for many different reasons.

A matrix can be used to represent a set of data that consists of several vectors by simply looking at each of the vectors as row for your matrix. An example would be if you have the ages, heights, and weights for 1,000 people, you are able to place them in a 1,000 X 3 matrix.

“ data = [[70, 170, 40],
[65, 120, 26],
[77, 250, 19],

 # …
] “

You are also able to use c X d matrix to show a linear function that will map your c-dimensional vectors to your d-dimensional vectors. There are a lot of concepts and techniques that will involve these types of functions.

The third thing you can do with matrices is to use them to represent binary relationships. One representation of an edge of a network is to show them as a collection pair (c, d). But another way you could do this is make a matrix K like K[c] [d] is one of the nodes c and d are connected and if not they are zero.

In the former representation you would have:
“ relationships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4), (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)] “

This could also be shown as:

“ relationships = [[0, 1, 1, 0, 0, 0, 0, 0, 0, 0], [1, 0, 1, 1, 0, 0, 0, 0, 0, 0], [1, 1, 0, 1, 0, 0, 0, 0, 0, 0], [0, 1, 1, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 1, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0, 0, 1, 0], [0, 0, 0, 0, 0, 1, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1, 1, 0, 1], [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]] “

If you don’t have many connections, then this wouldn’t be a very efficient representation because you will more than likely have a lot of stored zeros. However, when you use a matrix representation it will be a lot quicker to check if your two nodes are connected. To do this you would only to do a matrix lookup instead of having to inspect every edge.

“ relationships [0] [2] == 1

relationships [0] [8] == 1 “

If you are looking to find connections that a node has, you would have to inspect column or row that corresponds with the node.

“ friends_of_five = [c
For c, is_friend in enumerate(relationships[5]) If is_friend] “

Previously you may have added a connections list to all of the node objects to speed up the process, but when it comes to evolving a large graph that would end up being a bit too expensive, and it would be hard to maintain.

Statistics

“Statistics is the grammar of science.” – Karl Pearson

When it comes to data science, it is important that you have a good understanding of statistics so that you are able to convey the message that you need to.

Discrete Vs. Continuous

We are going to be looking at discrete variables. Discrete variables are variables that come from a limited set. They can also include numbers with decimals depending on your variable set, but this rule has to be established. For example, if you have the number 3.578 representing the number of medical procedures that person has had in their life, that's not possible. Even if this was just the average, it is still misleading.

You can't come out with the odds are that person has had 3.578 medical procedures in their life. They would have either had three or four. If you were looking at procedures, you would see numbers like this:

- Numbers of procedures
    - 1
    - 2
    - 3
- Odds of having that number of procedures in a year
    - 25%
    - 25%
    - 50%

When you look at continuous variables, they can't be visualized in a table. Instead, these variables have to be given in a formula as there are an infinite number of variables. An example of an input variable could be 2, 2.9, 2.99, 2.999, 2.9999 … n.

Examples of these variables could be age, weight, and so one. A person isn't just 32. They are typically 32 years old, 195 days, 2 hours, 1 second, 4 milliseconds. Technically, these variables could represent any single moment in time, and every interval contains infinite intervals.

Statistical Distributions

Poisson Distribution

$$p(x;\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!} \quad \text{for } x = 0,\ 1,\ 2,\ \cdots$$

A Poisson distribution equation is used to figure out how many events could happen during a continuous interval of time. One example would be the number of phone calls that could happen during a certain time, or the number of people that could end up in a queue.

This is actually a fairly simple equation to remember. The symbol is known a lambda. This is what represents the average amount of events that happen during a certain interval of time.

An example for this distribution equation is to figure out the loss in manufacturing sheets of metal with a machine that has X flaws that happen per yard. Let's say that the error rate is two errors per yard of metal. Now figure out what the odds are that two errors would occur in a single yard.

Binomial Distribution

This is one of the most common and the first taught distribution in a basic statistics class. Let’s say our experiment is flipping a coin. Specifically, the coin is flipped only three times. What are the odds that the coin will land on heads? Using combinatorics, we know that there are 2^3 or eight different results combinations. By graphing the odds of getting 3 heads, 2 heads, 1 heads, and 0 heads. This is your binomial distribution. On a graph, this will look just like a normal distribution. This is because binomial and normal distributions are very similar. The difference is that one is discrete and the other is continuous.

PDFs and CDFs

Probability Density Function

If you have ever taken a basic statistics class, you know this function better than you think. Remember standard deviations? How about when you calculated the odds between the standard and average deviation? Did you realize that you were using a calculus concept known as integrals? Now think about the space under the curve.

With this, we can assume that the space under the curve could be from negative infinity to positive infinity, or it could be a number set like the sides of a die. But the value under the curve is one so you would be calculating the space under two points in the curve. If we go back to the sheet metal example, trying to find the odds that two errors occur is a bit of a trick question. These are discrete variables and not continuous.

A continuous value would be zero percent.

Since the value is discrete, the integer will be whole. There wouldn't be any values between one and two, or between two and three. Instead, you would have 27% for two. If you wanted to know a value between two and three, what would the answer be?

PDF and the cumulative distribution function are able to take on continuous and discrete forms. Either way, you want to figure out how dense the odds are that fall under a range of points or a discrete point.

Cumulative Distribution Function

This function is the integral of the PDF. Both of these functions are used to provide random variables. To find the odds that a random variable is lower than a specific value you would us the cumulative distribution function.

The graph shows the cumulative probability. If you were looking at discrete variables, like the numbers on a die, you would receive a staircase looking graph. Every step up would have 1/6 of the value and the previous numbers. Once you reach the sixth step, you would have 100%. This means that each one

of the discrete variables has a 1/6 change of landing face up, and once it gets to the end the total is 100%.

Testing Data Science Models and Accuracy Analysis

ROC Curve Analysis

Data science and statistics both need the ROC analysis curve. It shows the performance of a model or test by looking at the total sensitivity versus its fall-out rate.

This plays a crucial role when it comes to figuring out a model's viability. However, like a lot of technological leaps, this was created because of war. During WWII they used it to detect enemy aircraft. After that, it moved into several other fields. It has been used to detect the similarities of bird songs, accuracy of tests, response of neurons, and more.

When a machine learning model is run, you will receive inaccurate predictions. Some of the inaccuracy is due to the fact that it needed to be labeled, say, true, but was labeled false. And others need to be false and not true.

What are the odds that the prediction is going to be correct? Since statistics and predictions are just supported guesses, it becomes very important that you are right. With an ROC curve, you are able to see how right the predictions are and using the two parable figure out where to place the threshold.

The threshold is where you choose if the binary classification is false or true, negative, or positive. It will also make what your Y and X variables are. As your parables reach each other, your curve will end up losing the space beneath it. This shows you that the model is less accurate no matter where your threshold is placed. When it comes to modeling most algorithms, the ROC curve is the first test performed. It will detect problems very early by letting you know if your model is accurate.

Some Algorithms and Theorems

There are Boolean, basic deduction, neural networks, decision trees, clustering algorithms, classification algorithms, and on and on.

Bayes Theorem

This is one of the more popular ones that most computer minded people need to understand. You can find it being discussed in lots of books. The best thing about the Bayes theorem is that it simplifies complex concepts. It provides a lot of information about statistics in just a few variables.

It works well with conditional probability, which means that if this happens, it will play a role in the resulting action. It will allow you to predict the odds of your hypothesis when you give it certain points of data.

You can use Bayes to look at the odds of somebody having cancer based upon age, or if spam emails are based on the wording of the message.

The theorem helps lower your uncertainty. This was used in WWII to figure out the locations of U-boats and predict how the Enigma machine was created to translate codes in German.

K-Nearest Neighbor Algorithm

This is one of the easiest algorithms to learn and use, so much so that Wikipedia refers to it as the “lazy algorithm.”

The concept of the algorithm is less statistics based and more reasonable deduction. Basically, it tries to identify the groups that are closest to each other. When k-NN is used on a two-dimensional model, it will rely on Euclidian distance.

This only happens if you are working with a one norm distance as it relates to square streets, and that cars are only able to travel in a single direction at a time. The point I’m making that the models and objects in this rely on two dimensions, just like the classic xy graph.

k-NN tries to identify groups that are situated around a certain number of points. K is the specified number of points. There are certain ways to figure out how big

your k needs to be because it is an inputted variable that the data science system or user has to pick.

This model is perfect for feature clustering, basic market segmentation, and finding groups that are among specific data points. The majority of programming languages will let you implement in a couple of code lines.

Bagging or Bootstrap Aggregating

Bagging will involve making several models of one algorithm like a decision tree. Each one of them will be trained on different bootstrap sample. Since this bootstrapping will involve sampling with replacement, some of your data won't be used in all of the trees.

The decision trees that are made are created with different samples, which will help to solve the problem of sample size overfitting. Decision trees that are created in this way will help to lower the total error since the variance will continue to lower with ever tree that is added, without increasing the bias.

A random forest is a bag of decision trees that use subspace sampling. There is only one selection of the trees features that is considered at the split of each node, which removes the correlation of the trees in your forest.

These random forests also have their own built-in validation tool. Since there is only a percentage of this data that gets used for every model, the error of the performance can be figure out using only 37% of the sample that was left by the models.

This was only a basic rundown of some statistical properties that are helpful in data science. While some data science teams will only run algorithms in R and Python libraries, it's still important to understand these small areas of data science. They will make easier abstraction and manipulation easier.

Decision Trees

“Data is a precious thing and will last longer than the systems themselves.” – Tim Berners-Lee

For banks to figure out if a they should offer a person a loan or not, they will often work through a list of questions to see if the person would be safe to give the loan to. These types of questions could start are simply like, “What kind of income do you have?” If the answer is between $30 and $70,000, they will continue onto the following question. “How long have you worked at your current job?” If they say one to five years, it will continue onto their next question. “Do you make regular credit card payments?” If they yes, then they will offer them a loan, and if they don’t they won’t get the loan. This is the most basic decision tree.

A decision tree is pretty much just a non-parametric machine learning modeling technique that is used for classification and regression problems. In order to find the solutions, a decision tree will create a hierarchical and sequential decision that variables of the outcome based on data.

And this means what?

Hierarchical refers to the model that is defined by a question series that will lead to a label or value once it has been applied to an observation. After it is set up, this model will work like a protocol using a bunch of “if this happen then that will happen” conditions that will give a certain result from the data that was added.

A method that is non-parametric means that there won’t be an underlying assumption concerning the distribution of the data or errors. This basically means that you model will be created by using observed data.

Decision trees that use a discrete value set for the target variable are classification trees. With these types of trees, the nodes, or leafs, are representations of class labels, and the branches show the feature conjunctions

that lead to the class. Decision trees that have target variables that are taking continuous value, which is typically numbers, are referred to as Regression Trees. Together, these two types of decision trees are called CART.

Each one of these models is a case of a Directed Acyclic Graph. All of the graphs have nodes that show a decision point about the top variable given the edges and predictor that are between each node. If we continue with the loan scenario, $30 to $70,000 would represent an edge, and “Years at present job” would be a node.

The main goal of the decision tree is to make the best choice once you reach the end of a node so it will need an algorithm that does that. This is known as Hunt’s algorithm, which works recursive and greedily. Greedily means it makes the best choice and recursive means that it split big questions into smaller ones. The choice of splitting a node is decided according a purity metric. Nodes are considered 100% impure if a node is split 50/50, and it is considered 100% if all of the data is a part of one class.

To make sure that the model is optimized you have to reach a max purity and stay away from impurity. You do this by using the Gini impurity, which will measure how often a random element ends up being labeled wrong if the distribution randomly labeled it. This is figured out by adding the odds, pi, of a node with the label, I, being picked then multiplied by the odds of a mistake in categorization. The goal is to make sure you reach 0 where it should be pure.

The other metric that it will use is information gain. This is for deciding the feature that you split at each tree step. This can be figured out using this equation.

“Information Gain = Entropy(parent) – Weight Sum of Entropy(Children)”

This is a pretty good model, but it presents a problem because it results in a model that will only stop once all of the information is distributed into a single attribute or class. At the cost of bias, the model’s variance is huge and will end up leading to over fitting. This can be fought by setting a max depth of your tree, or by setting an alternative to specify the minimum amount of points that will be

needed to decide to split.

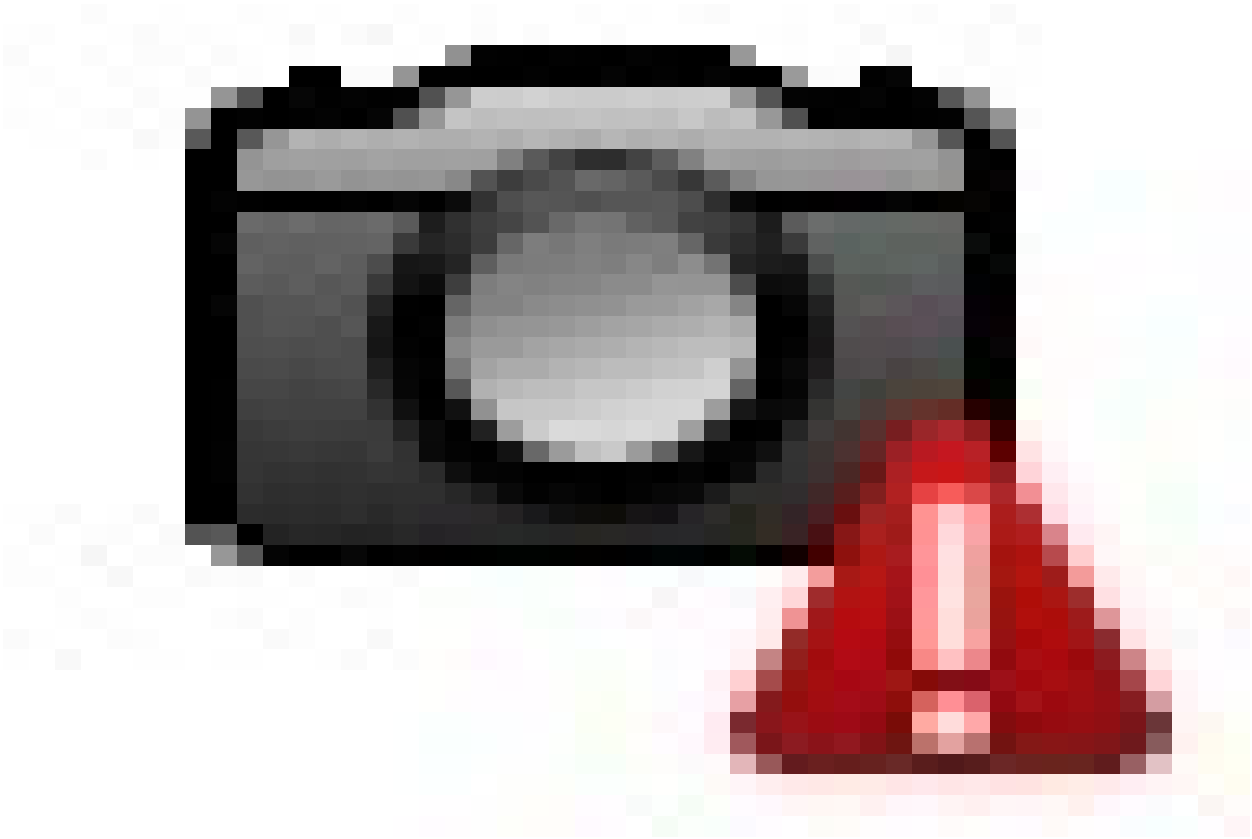
With decision, trees comes advantages and disadvantages. On the down side, they are greedy algorithms that are locally optimized where a return to the global tree isn’t guaranteed. On the positive side, they are super simply to understand because they have a visual representation that doesn’t require all that much data.

Neural Networks

“Data matures like wine, applications life fish.” – James Governor
Neural networks, which are sometimes referred to as Artificial Neural Networks, are a simulation of machine learning and human brain functionality problems. You should understand that neural networks don’t provide a solution for all of the problems that come up but instead provide the best results with several other techniques for various machine learning tasks. The most common neural networks are classification and clustering, which could also be used for regression, but you can use better methods for that.

A neuron is a building unit for a neural network, which works like a human neuron. A typical neural network will use a sigmoid function. This is typically used because of the nature of being able to write out the derivative using f(x), which works great for minimizing error.

Sigmoid function: f(x) = 1/1+e^-x
Neurons are then connected in layers in order for a single layer to communicate with other layers which will form the network. The layers that are within the input and output layers are known as hidden layers. The outputs of a layer are sent to the inputs of a different layer.



For a neural network to learn, you have to adjust the weights to get rid of most of the errors. This can be done by performing back propagation of the error. When it comes to a simple neuron that uses the Sigmoid function as its activation function, you can demonstrate the error as we did below. We can consider that a general case where the weight is termed as W and the inputs as X.

With this equation, the weight adjustment can be generalized, and you would have seen that this will only require the information from the other neuron levels. This is why this is a robust mechanism for learning, and it is known as back propagation algorithm.

To practice this, we can write out a simple JavaScript application that uses two images and will apply a filter to a specific image. All you will need is an image you want to change and fill in its filename where it says to in the code.

```
“ import Jimp = require(“jimp”);
Import Promise from “ts-promist”;
Const synaptic = require(“synaptic”); Const _ = require(“lodash”);
Const Neuron = synaptic.Neuron,
   Layer = synaptic.Layer, Network = synaptic.Network, Trainer =
synaptic.Trainer,
   Architect = synaptic.Architect; Function getImgData(filename) {
   Return new Promise((resolve, reject) => {
Jimp.read(filename).then((image) => {
Let inputSet: any = [];
Image.scan(0, 0, image.bitmap.width, image.bitmap.height, function (x, y, idx) {
Var red = image.bitmap.data[idx + 0]; Var green = image.bitmap.data[idx + 1];
inputSet.push([re, green]);
});
Resolve(inputSet);
   }).catch(function (err) {
Resolve([]);
   });
});
}
Const myPerceptron = new Archietect.Perceptron(4, 5); Const trainer = new
Trainer(myPerceptron); Const traininSet: any = [];
getImgData(‘ imagefilename.jpg’). then((inputs: any) => {
   getImageData(‘imagefilename.jpg’).then((outputs: any) => {
for (let i=0; I < inputs.length; i++) {
trainingSet.push({
```

input: _.map(inputs[i], (val: any) => val/255), output: _.map(outputs[i], (val: any) => val/255) });
}
Trainer.train(trainingSet, {
Rate:.1,
Interations: 200,
Error: .005,
Shuffle: true,
Log: 10,
Cost: Trainer.cost.CROSS_ENTROPY
});
Jimp.read('yours.jpg').then((image) => {
Image.scan(0, 0, image.bitmap.width, image.bitmap.height, (x, y, idx) => {
Var red = image.bitmap.data[idx + 0]; Var green = image.bitmap.data [idx + 1]; Var out – myPerceptron.activate([red/255, green/255);
*Image.bitmap.data[idx + 0] = _.round(out[0] * 255); Image.bitmap.data[idx + 1] = _.round(out[1] * 255); });*
Console.log('out.jpg');
Image.write('out.jpg');
}).catch(function (err) {
Console.error(err);
});
});
});

Scalable Data Processing

“Talented data scientists leverage data that everybody sees; visionary data scientists leverage data that nobody sees.” – Vincent Granville, Executive Data Scientist & Co-Founder of Data Science Central

Processing frameworks compute a systems’ data through ingesting it into the system or reading non-volatile storage. Computing over data is where you extract insight and information from a large amount of data points.

Processing engines and frameworks are used for computing the data in your system. While there isn’t really a definition that sets engines and frameworks apart, it’s helpful to define the former as the component that is responsible for working with the data, and the latter is a set of components that are created to do the same thin.

Apache Hadoop can be seen as a processing framework, and MapReduce would be seen as the processing engine. Often times you can swap out engines and frameworks, or you can use them together. Apache Spark is a framework that is able to be connected to a Hadoop to replace MapReduce. The fact that the components are able to work together is the main reason big data systems tend to be flexible.

While these types of systems that take care of this part of the data life cycle tend to be complex, the goals across a broad level are similar. They operate over data so that they can improve the surface patterns, understanding, and create insights into the interactions.

To make things simpler, we will combine some of these processing frameworks by the data that they are created to handle. There are some systems that will handle data in batches, while there are others that process data in a more continuous stream while it is flowing through the system. Then there are others that handle the data in both ways.

Batch Processing Systems

Batch processing works by looking at a large and static set of data and then returns the results at another time once the computation has been completed.
The data that is normally processed in batch is:

- Large: batch operations tend to be the only option when it comes to process large amounts of data.
- Persistent: data is typically backed by some form of storage that is permanent.
- Bounded: batch sets of data represent a certain group of data.

Batch processing tends to be well-suited for calculations where you have to have access to whole set of records. For example, when you are calculating averages and totals, sets of data should be treated holistically and not as a group of individual records. These types of operations will require that state is kept for the complete time of the calculation.
Tasks that need a large amount of data are typically best handled with a batch operation. Batch systems are created expecting large amounts of data and will have the resource that they need in order to handle them. Since batch processing is amazing at taking care of large volumes of data, it tends to be used with data that is historical.
The trade-off for all of this large quantity handling means that it takes a longer time to compute. Because of this batch processing isn’t always good to use in certain situations where the time it takes plays an important role.

Apache Hadoop

This is a processing framework that provides batch processing. Hadoop is the first framework that ended up gaining a fair amount of significant traction within the open-source world. After several presentations and papers from Google about the way they were dealing with large amounts of data, Hadoop started to use components and algorithm stacks so that they could make processing large amounts of data easier.

The modern version of Hadoop is made up of a lot of layers or components that all work together in order to process batch data: **MapReduce**: this is the native batch processing engine.

YARN: this stands for Yet Another Resource Negotiator, and is a cluster coordinating component for this stack. It makes sure that the underlying scheduling and resources are managed and coordinated. YARN is what makes things possible to be able to run more diverse workloads through this cluster than could be done with earlier iterations by working like an interface for the resources.

HDFS: this is the distributed file system that uses node clusters to coordinate replication data storage. HDFS makes sure that data stays available no matter the possible host failures. This is used by data sources in order to store the intermediate results and perfect the final results.

Hadoop's processing functionality comes from MapReduce. The processing technique follows the reduce, map, and shuffle algorithm using the key-value pairs. This procedure will involve:

- Reading the set of data from HDFS.
- Dividing the set of data up into chunks that is distributed through the nodes that are available.
- Applying the computation for each of the nodes to the data subset.
- Redistributing the intermediate results so that they are grouped by key.

- Reduce the value of the keys by combining and summarizing the results that the individual nodes calculated.
- Writing the final results into the HDFS.

Stream Processing Systems

This system computes the data as it goes into the system. This will require different processing models than the batch processing system. Instead of defining the operation that needs to be applied to the whole dataset, these processors will define the operations that have to be applied to the individual pieces of data as it goes through the system.

In stream processing, the datasets are seen as unbounded. There are few implications:

- The processing is based on events, and it doesn't end until somebody stops it. Results will then be shared immediately and will always be updated, and new data comes in.
- The working set of data tends to be more relevant and is limited to only one thing at a time.
- The total set of data is defined the how much data has come into the system thus far.

Stream processing systems are able to handle pretty much an unlimited amount of data, but they are only able to process one, or very few items at a single time, with very little status maintained in between the records. While there are a lot of systems that provide methods to maintain some state, this type of processing is optimized for better functionality with very few side effects.

Apache Storm

This is a stream processing framework that's main focus is extreme low latency and is probably one of the best options when it comes to work that needs almost real-time results. It is able to handle large amounts of data and provide results with less latency than other types of applications.

Storm works by using DAG (Directed Acyclic Graphs) in a framework known as topologies. This framework describes the different steps that have to be taken for every piece of data that goes into the system.

These topologies are made up of: **Bolts** – these are the processing steps that make up the streams, applies the operation, and provide the results. There are bolts on every spout, and they connect to each other to create the processing that is needed. A final bolt output can be used at the end of the topology for a connected system.

Spouts – the sources of the data travel to the edge of the topology. This could end up being queues, APIs, and so on that creates the data that needs to be operated on.

Streams – this is the unbounded data that will continuously flow into the system.

The purpose of Storm is to define discrete and small operations that use these components and then create a topology. Storm provides an at-least-once guarantee, which means that it guarantees that each message will be process at least one time, but you can find some duplicates when there is a failure. Storm doesn't guarantee messages will processed in the order that they came in.

Apache Samza

This processing framework is very close in nature to Apache Kafka messaging system. Kafka can be used for most stream systems, and Samza is made so that it specifically takes advantage of Kafka's guarantees and architecture. Kafka is there to provide Samza with fault tolerance, state storage, and buffering.

For resource negotiation, Samza uses YARN. This means that there will have to be a Hadoop cluster, but this also means that there will be the features of YARN present.

Samza uses the semantics of Kafka to define how their streams will be handled. Kafka uses these types of concepts when they are dealing with data: **Consumers** – these are the parts that read from a Kafka topic.

Producer – these are the components that write to a Kafka topic.

Brokers – these are the nodes that make of the clusters.

Partitions – the topics are distributed evenly among nodes by dividing the incoming messages into partitions.

Topics – all of the data that enters into the Kafka system is known as topics.

Hybrid Processing Systems

There are some frameworks that are able to handle stream and batch workloads. These make the diverse requirements simple by letting the same or similar APIs and components to be used for each data type.

The way this is done will vary between the different frameworks like we will see in Flink and Spark. This is pretty much the function of how the processing paradigms come together and the assumptions that end up being made about the data's relationship between the unfixed and fixed data.

While you may have a project that fits closely for a specific system, a hybrid system is there to try and offer a basic solution for the processing. They not only give you a method for processing the data, but they will have their own tooling, integrations, and libraries for things like interactive querying, graph analysis, and machine learning.

Apache Spark

This is a framework that is batch processing with stream processing ability. It is built with a lot of the same principles of MapReduce, and it focuses mainly on the processing speed of its workload by providing full processing and in-memory computation.

Spark can be used standalone, or it can connect with Hadoop as a MapReduce alternative.

Spark will process all of the data in-memory and will only interact with the storage layer when it starts to load the data. At the end, it will provide the final results. Everything else is managed through memory.

Spark works faster when it comes to disk-related tasks because of their optimization that is able to be reached by looking at the entire task beforehand. It is able to do this by making DAGs, which show the operations that need to be done, the data to be worked on, and the relationship between them, which will give the processor better chance of coordinating work.

Stream gets it stream processing from Spark Streaming. Spark by itself is made for batch-oriented work. Spark has implemented a design known as micro-batches. This strategy was created to treat streams of data as if it were little batches of data that it can handle by using its batch engine’s semantics.

Apache Flink

This is a stream framework which can handle batch data. It sees the batches and streams of data that has defined boundaries, so it ends up treating batch processing as a stream processing subset.

Their stream-first approach is called the Kappa architecture, which is in contrast to the better-known Lambda. Kappa works by treating everything as streams, simplifies the model, and very recently it has become possible because of the sophistication of stream engines.

Flink's model works with incoming data as it comes in, just like a true stream. It uses a DataStream API to work on the unbounded data that comes in. Its basic components are: **Sink** – this is where the stream comes out of the system.

Sources – this is where the streams enter the system.

Operators – these are functions that operate on streams of data to create new streams.

Streams – these are the unbounded and immutable sets of data that travel through the system.

The batch processing for Flink works similarly to an extension of its stream model. Instead of reading from a continuous stream it read as a bounded set of data off of the persistent storage. The runtime for both of these processing models are exactly the same for Flink.

In the end, Hadoop is a great option if you have batch-only workload and the batches aren't time-sensitive. Storm is a good option for stream-only workloads because it provides low latency processing. Smaza uses Kafka and YARN so that it is able to be flexible and straightforward.

If you have mixed workloads, your best choices are Flink and Spark. Spark has a large number of tools and libraries. Flink will provide you with great stream processing that supports batch processing.

Finding the best fit for you is up to you to figure out. It will depend upon the data that you have to process, and how bound by time you are.

Data Science Applications

“The big technology trend is to make systems intelligent, and data is the raw material.” – Amod Malviya, CTO at flipkart

Even after everything in this book, some people will still be driven to ask, “Is data science really popular or is it going to be a one-time opportunity?” Different people will likely have different viewpoints. Instead of starting a debate, we’re going to take a safer approach. We are going to look at the applications that affect the layman’s life, and where data science plays a large role.

Internet Search

Chances are when you hear the word search your first thought is Google. But there are actually several other search engines out there such as duckduckgo, AOL, Ask, Big, and Yahoo. Every search engine out there uses some form of a data science algorithm to provide their uses the best results for their search query in less than a second. Think about this. Google processes over 20 petabytes of

data every single day. If there wasn't any data science, Google would not be as good as it is today.

There may be a lot of companies that try to make data-based decisions; Google uses a rigorous statistical analysis and scientific testing that is more commonly found in university labs. Google did what they called People Analytics so that they could better understand how humans worked so that they can improve their services and their hiring process.

Digital Advertisements

The entire digital marketing spectrum is the biggest application for data science. Beginning with the display banners on different sites to the digital billboards that you see in airports, nearly every single one of them uses data science algorithms to run properly.

This is why digital ads are able to receive better CTR than the traditional ads. These ads are able to be targeted based up the behavior of the user.

Data science helped to improve digital ads by helping people find the best bid price depending on the use, and the network that they are bidding on. The best thing is that they can spend their advertising budget more efficiently.

They are also able to detect fraud. This is mainly click based fraud performed by malicious bots. They can examine the statistical properties that pertain to the clicking behavior of the regular users versus the click properties from the malicious bots, and then they can find algorithms to combat the ever changing strategies of these bots.

They also group their customers into different interest segments based on their browsing history, but they also use third party data to find the relevant clusters where the customers share similar properties. When they find customers that have the same properties, the advertisers target these audiences and prioritize the high value audiences.

Recommender Systems

Recommender systems are tools that were made for interacting with complex and large amounts of information spaces and prioritize items in these spaces that

will likely interest the user. This area was created in 1995, and has grown enormously in the variety of problems addressed.

Think about the suggestions Amazon gives you. They help you to find relevant products from billions of others, but that also improve your experience. There are a lot of companies out there that use this system to promote suggestions that align with their user's internet. The giants of the internet like imdb, Linkdin, Netflix, Google Play, Twitter, Amazon, and several more use this type of system to make their user's experience better. The recommendations you see are based upon your previous searches.

The purpose of recommender systems could include:

- Assistance in exploration
- Assistance is discovery
- Assistance in comparison
- Assistance in decision making

Recommender systems can also be helpful in calculating the behavior of these users:

- Loyal users and unpopular items
- Loyal users and popular items
- Non-loyal users and unpopular items
- Non-loyal users and popular items

Image Recognition

Whenever you upload a picture to Facebook of you and your friends, you start to get suggest of who you should tag in the photo. This automatic suggestion uses a face recognition algorithm. In the same way as the WhatsApp, you are able to scan a barcode while in your phones' web browser. Google also provides you with the chance to search for images by uploading one. It makes use of image recognition and then gives you related results.

Hiring humans to manual tag libraries of music and movies is a daunting task, but it's impossible when you're talking about the challenges of teaching a navigation system in a driverless car to distinguish pedestrians crossing the road

from other vehicles, or filtering, tagging, and categorizing the millions of user-upload videos and pictures that show up every day on social media.
Neural networks are commonly used for analyzing images, but conventional neural networks tend to be more expensive. It would take 900 inputs and more than half a million parameters to process a small image. That's why people tend to use convolutional neural networks for better image recognition skills.

Speech Recognition

Cortana, Google Voice, and Siri are the best examples of speech recognition. Whether you can type or not, you can speak your message, and it will be changed to text.
Data science helps by assisting the talk and speech applications by recognizing voice messages in a more effective manner, and then produces accurate text outputs in response. Deep learning is one of the most reliable techniques for coming up with an exact and accurate speech recognition result.
One of the traditional ways was to use a simple neural network to produce the text output, but the problem is that the speech wasn't able to be entered in the same speed and pitch. Since everybody's input voice varies, the system needed to be powerful enough to detect the right words independently. That's when people started sampling different voices. Then they created the recurrent neural network.

Gaming

Activision-Blizzard, Nintendo, Sony, Zynga, and EA Sports have been the leaders in the gaming world and brought it to the next level through data science. Games are now being created by using machine learning algorithms which are able to upgrade and improve playing as the player moves through the game. When you are playing a motion game, the computer analyzes the previous moves to change the way the game performs.
Players interact with their games in several ways, like the device used, in-game purchases, social media gaming usage, dedication level, playing style, and amount of time spent playing. The huge amount of gaming styles, along with the

popularity of subscription and in-game revenue opportunities, means that all of the gaming companies have to tailor their advertising to maximize their revenue. Data analytics will allow these companies to collect, cleanse, format, and model the data so that they can get a clear picture of the way their users interact with their games. With time, a profile for their players will emerge, which will enable companies to offer highly-specific products based upon their users gameplays. These are only a few applications for data science, there are many, many more out there.

Conclusion

You have officially made through this comprehensive guide to data science. There was a lot of information in this book, and I hope that it will help you in a career in data science. The best way to proceed is to read back through this book and practice all of the different guides. Focus on the parts that you find more difficult until they are no longer difficult.

The goal of data science is to help improve the decision-making process for different businesses, which is done by basing decisions on different insights that has been extracted from large sets of data. Data science as a field encompasses a certain set of principles, algorithms, problem definitions, and process for finding useful patterns from a large set of data.

Today, the decision-making process of data science is used in almost every area of modern society. Some of the ways that data science could affect the daily life of humans includes figuring out which ads should be presented online; which friend, movie, and book connections you are shown; which emails end up being sent to your spam folder; what offers you end up getting when you renew cell service; how much your health insurance premiums cost you; the timing and sequencing of traffic lights; how the drugs you could take were designed; and the locations where your city's police are targeting.

The growth of the data science industry across society has been driven by social media and big data, the quickening of computer power, the huge reduction in computer memory cost, and the creation of powerful methods for data modeling and analysis. All of these factors together mean that it hasn't ever been easier for businesses to process, gather, and store data. Along with this, innovations and the bigger applications for data science means that the ethics of using the data and individual privacy is an even bigger problem.

When it comes to coding, the best thing you can do is code. Try out all of the different coding languages and get familiar with them all just in case you have to

use them. The most commonly used language is Python, so take extra care in learning it.

As you have probably figured out, there is a pretty big debate over which programming language a data scientist should use, especially when you are just learning about data science. There are a lot of people who think that programming language R is the best, which is wrong. There are even a few that think Scala and Java are the best and they are also wrong. Python, in my opinion, is the obvious best option.

If you're still not convinced, let's review some of the best features of Python that makes it the best for learning and doing data science:

- It comes with a bunch of data science-related libraries.
- It is a pretty simple code to learn and to understand, so it perfect for beginners.
- It is completely free (you're sold, I'm sure.)

Python may not be the favorite programming language among all data scientists, but it's the best to start with. Data scientists could see other programming languages as more pleasant, better-designed, or more fun to code with, and yet they still end up using Python when they start a new data science project.

Similarly, the best way to make sure that you understand mathematics is by practicing math. Data science, and this book is not inherently math based so you may not have to worry about "doing math." That said, you can't go into a data science career without knowing something about math, especially linear algebra, statistics, and probability. This means that in the areas that are appropriate you need to really dive into different mathematical equations, mathematical axioms, and mathematical intuition. Try not to let the math involved scare you away from data science because it really is only a small part of it.

While math and statistics may seem boring, they really are amazing tools, especially when it comes to data science. Statistics can be used to help explain things from the idiocy of participating in the lottery to DNA testing. Statistics can be used to identify the factors that are associated with things like heart

disease and cancer. They can also help people spot cheating on standardized tests. Statistics can also be used to help you win game shows.

Besides learning the information found in this book, it's important that you realize the future of data science. Jobs in data science have grown from 2.3 million in 2015 to 2.9 million in 2018.

The combination of cheap and fast computation with statistical methods has allowed for lots of new methods such as machine learning. This doesn't even consider the cheaper and more reliable ways we have to store data today. That means we're storing even more of it. This is why businesses want to find statisticians who can code or programmers that understand stats, as well as the desire for new tools to help store and process data.

While the tools used for the job could change in the future, the need for these types of people isn't going to go anywhere. The need for a data scientist isn't going anywhere. While Python could be a distant memory in ten year (doubt it) that's just how programming language works. Somebody will come up with something more efficient, and the data scientist will have to learn how to use it.

In the future, we will likely see new sources of data. While the most common datasets normally include clickstream data or sales and purchase data, more data scientists will start asking for sensor-generated data from vehicles, retail environment, manufacturing lines, and offices.

There will be new tools that will make this work easier to do. This can be seen with BI tools and open source libraries in the Python and R communities. There are now algorithms, which you would have had to code from scratch ten years ago, available through "from sklearn.neighbors import LSHForest".

Lastly, we will see quantitative methods and data science become more distributed throughout several roles instead of only being concentrated in a single department or role.

I hope that through this book I gave you a sense that playing around with data can be fun because it is actually fun.

About the author

Steven Cooper is a data scientist and worked as a software engineer at multiple startups. Now he works as a freelancer and helping big companies in their marketing and statistical analysis using machine learning and deep learning techniques.

Steven has many years of experience with coding in Python and has given several seminars on the practical applications of data science, machine learning, and deep learning over the years. In addition he delivers training and coaching services that help technical professionals advance their careers.

He loves to write and talk about data science, machine learning, and Python, and he is very motivated to help people developing data-driven solutions without necessarily requiring a machine learning background.

When not writing or programming, Steven enjoys spending time with his daughters or relaxing at the lake with his wife.

References

Flink

https://flink.apache.org/ - this is the main website for Flink.

Hadoop

https://www.cloudera.com/downloads.html - these are great intros into what Hadoop can do.

http://hadoop.apache.org/ - this is the main site for hadoop.

Java

https://www.codecademy.com/learn/learn-java - a course to learn how to use Java.

Julia

https://julialang.org/ - this is the main website for Julia.

Python

https://www.python.org/ - this is the main website for Python.

R

https://www.pluralsight.com/search?q=R – this is an interactive, free, and short introduction to R.

https://www.coursera.org/specializations/statistics - this is a free or paid online course that will teach you how to use R for statistics.

https://www.r-project.org/ - this is the main website for R.

Samza

http://samza.apache.org/ - this is the main website for samza.

SAS

https://www.sas.com/en_us/home.html - this is the main website for SAS.

Scala

https://www.scala-lang.org/ - this is the main website for Scala.

Spark

https://spark.apache.org/ - this is the main website for spark.

SQL

https://www.codecademy.com/learn/learn-sql - this is a course to teach you SQL.

Storm

http://storm.apache.org/ - this is the main site for storm.