Table of Contents

BOOK 1: Python for Data Science

Master Data Analysis from Scratch, with Business Analytics Tools and Step-by-Step techniques for Beginners. The Future of Machine Learning & Applied Artificial Intelligence

Download the Audio Book Version of This Book for FREE

If you love listening to audio books on-the-go, I have great news for you. You can download the audio book version of this book for **FREE** just by signing up for a **FREE** 30-day audible trial! See below for more details!

Audible Trial Benefits

As an audible customer, you will receive the below benefits with your 30-day free trial:

- FREE audible book copy of this book
- After the trial, you will get 1 credit each month to use on any audiobook
- Your credits automatically roll over to the next month if you don't use them
- Choose from Audible's 200.000 + titles
- Listen anywhere with the Audible app across multiple devices
- Make easy, no-hassle exchanges of any audiobook you don't love
- Keep your audiobooks forever, even if you cancel your membership
- And much more

Click the links below to get started!

>> For Audible US

>> For Audible UK

>> For Audible FR

>> For Audible DE

Introduction

Python and Data Science are two most popular technical terms that you will hear everywhere. The combination of these two terms will provide you an added advantage in current tech era. As it has been proven every year, the application and significance of Python is expanding daily, especially with the data science community and data analytics society.

In this book, we will guide you on how to get started:

Why read on? You will learn how to apply data analysis, which is cooler and advanced than using Microsoft Excel. Secondly, you will also learn how to implement Python data analysis and develop a mindset of an expert data analyst.
The most important, you will learn Machine learning and Python, and how it works in real-world problems. You will see several examples of how the modern methods of a data analyst suit in solving modern problems.

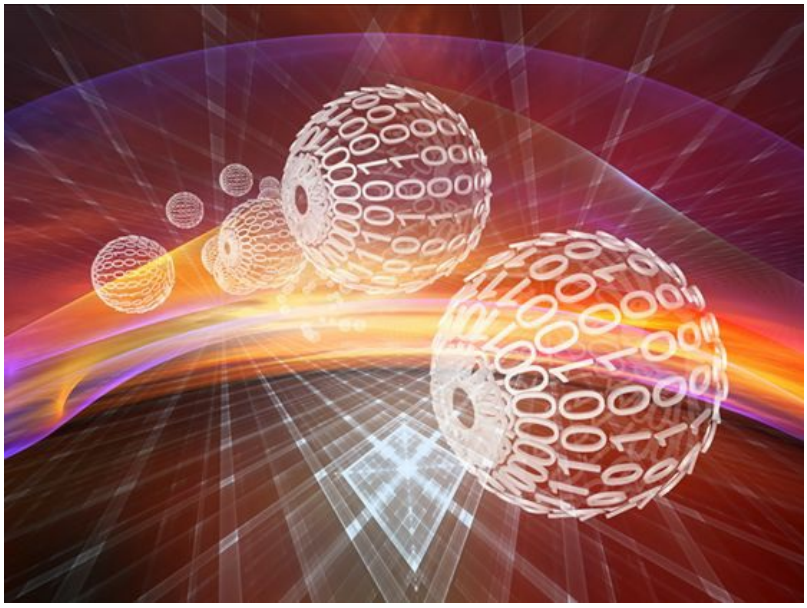
This is vital because a lot of data provides us with numerous opportunities to gain insights and make a useful impact in any field. This phenomenon also delivers new challenges that demand new technologies and methods. Besides, this demands new skills and mindsets to successfully navigate through the problems and successfully tap the biggest potential of the opportunities available.
This book will provide you a great foundation to advance into a complex data analysis with Python.

Python is one of the most common tools used in data analysis. In a certain survey conducted by Analytics India Magazine, it was discovered that 44% of data scientists prefer to use Python than SQL, SAS, and R. Not only that, Python is the only best general-purpose programming language that you can rely on.

As you will be learning the important technical skills needed in Python to complete data science projects, you should focus on practice. Start to search for data from various sources and begin to play around with them. It is highly advised that you apply a lot of operations as possible so that you become familiar with different approaches.

The Python programming language is a great tool for any data scientist because provides effective libraries that carry the tools of the algorithms and models that are necessary for analysis.

Chapter 1: Introduction to Data Analysis



The act of inspecting, cleaning, transforming, and modeling data with the intention to find relevant information and reaching a conclusion to boost the decision-making process is referred to as Data Analysis.

There are different facets and techniques for data analysis. The data analysis in statistics is categorized into exploratory data analysis, confirmatory data analysis, and descriptive statistics. Data requires to be cleaned. Data cleaning is the procedure of fixing the outliers and other inaccurate and unwanted information. There are different types of data cleaning processes to use based on the type of data to be cleaned.
Business intelligence explores the data analysis that extends heavily on aggregation, slicing, disaggregation, dicing, and concentrating on the business information. Predictive analytics refers to the use of statistical models for predictive forecasting. Text analytics describes the application of statistical, structural models, and linguistic models to derive and categorize the information from texts. All these describe different varieties of data analysis.

Python for Data Science

Why Select Python

Python is a simple, clear, and elaborate programming language. That is the reason many scientists and engineers go for Python to implement numerous applications. Probably they prefer handling the main task quickly rather than spend hundreds of hours mastering the nuances of a “complex” programming language.

This lets scientists, engineers, researchers, and analysts to dive into the project more quickly. As a result, they gain important insights into the minimum amount of time and resources. This does not imply that Python is fit and perfect programming language on where to complete data analysis and machine learning. Languages such as R may have advantages and properties Python doesn't have. However, Python is a good starting point, and you may attain the right knowledge of data analysis if you use it in future projects.

Python vs. R

You may have already come across this in Reddit, Stack Overflow, and other forums and websites. You may have also searched for other programming languages because even learning Python or R requires a few weeks and months. It is a huge time investment, and you don't want to make any mistake.

To avoid any confusion, just begin with Python because the usual skills and concepts are easy to transfer to other languages. In some cases, you may need to adopt a whole new approach to thinking. But all in all, understanding how to apply Python in data analysis will provide you a channel to solve many complex problems.

Some may say that R is designed for statisticians, especially when it comes to easy and better data visualization properties. It's also easy to learn, especially if you plan to use it for data analysis. On the flipside, Python is a bit flexible because it goes past data analysis. Most data scientists and machine learning practitioners may have selected Python because the code they write can be integrated into a dynamic and live web application.
While it's open for debate, Python is still a great choice for beginners or anyone who wants to get started with data analysis and machine learning. It is quite easy to learn, and you can dive into full-time programming if you see that this suits you well.

Widespread Application of Data Analysis

There are a lot of packages and tools that enable the usage of Python in Data analysis and machine learning. Some of these packages consist of Numpy, scikit-learn, and Pandas. These tools simplify the data analysis process.

Additionally, graduates from the university can dive into data science because most universities nowadays provide an introductory course in computer science with Python as the main programming language. The change from computer programming and software development can happen so fast because many people already have the correct foundation to begin learning and using programming to tackle real-world data problems.

Another reason for Python's popularity use is that there are numerous resources that will show you how to complete anything. If you've any questions, it is because someone else has requested

that, and another solved it for you. This makes Python more popular because of the presence of resources online.

Clarity

Because of the ease of learning and Python's syntax clarity, experts can concentrate on the crucial features of their projects and problems. For instance, they can apply Numpy, TensorFlow, and scikit-learn to gain insights rather than build everything from scratch.
This generates another level of clarity because professionals can concentrate on the nature of the problem and its effects. They can also come up with effective approaches to handling the problem rather than being bombarded with the many challenges a given programming language presents.

The main aim should always be on the challenge and the opportunities it may bring. It only requires a single breakthrough to change the entire thinking about a given problem, and Python can achieve that because of its ease and clarity.

Types of Data Analysis

Descriptive Analysis

Data science is related to information retrieval and data collection approaches with the aim of reconstituting past events to determine patterns and identify insights that help determine what happened and what made it happen. For instance, looking at sales figures by region to identify customer preferences. This section requires that you remain familiar with statistics and data visualization approaches.

Predictive Analysis

Data science is a means to determine the chances that some events are currently taking place or will occur in the future. In the following case, the data scientist considers past data to determine explanatory variables and create statistical models that can be used in other data points. For instance, trying to predict the probability that a certain credit card transaction is fraudulent in real-time. This section is usually related to the machine learning field.

Prescriptive Analysis

In this case, data analysis is considered as a way to make informed decisions, or probably data-driven decisions. The focus should be to search for multiple options and applying simulation approaches and optimize the results. For instance, maximizing the supply chain by aiming to reduce the operating costs.
Typically, descriptive data science aims to address the question of what does the data tell me. On the other hand, the predictive analysis answers the question of why is the data behaving in this

way, and prescriptive analysis respond to the question of how you optimize the data toward a given goal.

Why Data Analysis Is on the Rise?

There are several factors that lead to the rise of data science.

First, the amount of data gathered increases at an exponential rate. According to a certain IBM Marketing Cloud research, around 2.5 quintillion bytes are generated daily. This is around 2.5 billion bytes, but only a small fraction of this data is ever analyzed, leaving a lot of missed opportunities on the table.

Secondly, we are experiencing a cognitive revolution that began some years ago. Nearly every industry is going for AI technology, which comprises of natural language processing (NLP) and machine learning. Although these sections existed for some time, they have recently achieved renewed attention to the point that they are now popular courses in colleges. It is now clear that, if they are to survive, companies require to be more agile, move faster, and shift into digital businesses, and as the time available for decision-making reduces, they must be fully data-driven.

If you also include the fact that AI algorithms require high-quality data, you can start to understand the vital role played by data scientists.

Another reason is that with the growth of cloud technologies and the rise of platforms as a Service (PaaS), access to computing engines and storage has never been easier. Running big data workloads is now available to smaller organizations or any person with a credit card. In turn, this is driving the growth of innovation across the board.

Because of the following reasons, there is no question that data science is here with us and that its development will continue for a long time. Still, we cannot ignore the fact that it hasn’t achieved its full potential and generated the right results, especially in enabling companies to transform into data-driven organizations. In most cases, the problem lies in achieving that next step, which is to convert data science and analytics into a key business activity that provides clear-sighted and intelligent business decisions.

Summary of the Data Science Process

Sticking to a structured method in data science helps you to increase your chances of success in a data science project at the lowest cost. It also makes it possible to handle a project as a team, with every team member concentrating on what they do best. However, this procedure might not be helpful for every project type or be the only means to implement the right data science.

The typical data science procedure comprises of six steps through which you will iterate.

Here is a short introduction to each of the steps.

1. The first step is to set a research goal. The main focus here is to ensure all the stakeholders know the what, how, and why of the project.

2. The next step is data retrieval. You need to have data ready for analysis, so the following step involves searching for the best data and getting access to the data from the data owner. The result is that raw data probably demands polishing and transformation before it is usable.

3. Once you have the raw data, the next thing is to prepare it. This involves transforming the data from a raw type into data that's directly usable in your models. To achieve this, you will identify and correct various types of errors within the data, integrate data from various sources, and change it. If you complete this step successfully, you can move to data visualization and modeling.

4. The fourth step is data exploration. The purpose of this step is to attain a deep understanding of the data. You will search for patterns, deviations, and correlations depending on visual and descriptive techniques. The insights you derive from this step will allow you to begin modeling.

5. Lastly, you will get to the most interesting part. And that is data modeling. This is the time when you try to make the insights or predictions outlined in your project charter. Now is the time to generate heavy guns, but remember that research has taught us that always a mix of simple models tends to overcome a complex model. If you have completed this phase, you're almost done.

6. The last step of the data science model is the results presentation and automating the analysis. One focus of a project is to change a process or make better decisions. You might still need to convince the business that your findings will, in fact, change the business process as required. This is the point where you can shine in your influence role. The importance of the following step is obvious on a strategic and tactical level. Some projects require you to complete the business process over and over again. Hence, automating the project saves time.

The reality is that this is not going to be a straightforward process. In most cases, you will regress and iterate between the various phases.

Following the above six steps pays off based on a higher project success ratio and an increased

effect of research results. This process helps you attain a well-defined research plan, a great understanding of the business question, and clear deliverables before you begin looking at the data. The first steps allow you to concentrate on achieving high-quality data as input for your models. This way, your model will do well in the later stages.

Another advantage of sticking to a structured approach is that you work a lot more in a prototype model while you look for the best model. When creating a prototype, you will probably attempt multiple models and won't concentrate heavily on problems such as program speed or writing code standards. This enables you to concentrate on generating business value.

Not every project is created by the business itself. Insights derived at the time of analysis or the arrival of new data can generate new projects. When the data science team comes up with an idea, work has already been conducted to present a preposition and identify a business sponsor.

Dividing a project into smaller stages will enable employees to work together as a team. It is impossible to be an expert in everything. You don't need to learn how to upload all the data to all the various databases, determine an optimal data scheme that works not only for your application but also for other projects within your company, and then monitor all the statistical and data mining techniques.

Prerequisite and Reminders

By now, you need to be familiar with the Python syntax plus other basic things in Python. Some of the basic things you need to be conversant include functions, variables, loops, and lists. You don't need to be a specialist, but it's important to have the right knowledge, so the rest become smoother.

You don't need to make it hard because programming revolves around telling the computer what is supposed to be done. Then, the computer should be able to understand and successfully implement your instructions. You may only need to write several lines of code to suit your application.

Plus, a lot of things that you will implement in Python for data analysis are already pre-built for you. So, you may only need to copy and run the code. But remember that mastering Python programming is still important. This will also provide you with confidence because you understand how something operates.

Do You Need Some Expertise in Mathematics?

Data analysis implies working with numbers and mining useful insights from them. But do you really need to be an expert on numbers and mathematics?
Successful data analysis using Python always demands great skills and knowledge in math,

programming, and the field you are working on. This implies that you don’t have to be an expert in any of them.

Don’t allow a lot of experts to fool you because many of them are fake or just inexperienced. What you really need to know is what's the next thing to do so you can successfully complete your projects. You won’t be an expert in anything once you read all the chapters here. But this is enough to provide you a better knowledge about Python and data analysis.

When we consider the mathematical expertise, it is probably right that you’re familiar with mean, standard deviation, and other common terms in statistics. While extending further into data analysis, you may come across calculus and linear algebra. If you have the time and interest to explore them, you can always do anytime, or later. This may or may not provide you with the advantage of the specific data analysis project you’re working on.

Again, remember that it is about solving problems. The main focus should be on how to handle the challenge and successfully overcome it. This should apply to all fields in business and science. Don’t allow the myths to distract you. Concentrate on the key concepts, and you will be fine.

Chapter 2: Python Review

In this chapter, we shall look at the Python programming language basic stuff. This chapter tries to teach you Python in brief. It's so much as a cheatsheet, so it will only remind you of some basic features to start you off. Typically, if you really want to master a language, you need to commit time and program it for a while. This chapter assumes that you're already familiar with Python programming and will, therefore, skip the majority of the non-language–specific stuff. The important keywords will be pointed out so you can easily spot them.
We shall concentrate on Python 3 because that is the version you need to use. All the examples we shall provide to you are written in Python 3.

Properties

First, Python is a strongly typed language.

Getting help

Help in Python is usually available within the interpreter. If you want to learn how an object operates, all you need to do is call **help (<0bject>)!** Additionally, the most important are dir (), which displays all the object's methods, and **<object>._doc_,** which displays all the documentation string.

Let's set some expectations here, so you know where you're going. This is also to introduce the limitations of Python, data analysis, data science, and machine learning (and also the key differences). Let's start.

Data Analysis vs. Data Science vs. Machine Learning

Data Analysis and Data Science are almost the same because they share the same goal, which is to derive insights from data and use it for better decision-making.

Often, data analysis is associated with using Microsoft Excel and other tools for summarizing data and finding patterns. On the other hand, data science is often associated with using programming to deal with massive data sets. In fact, data science became popular as a result of the generation of gigabytes of data coming from online sources and activities (search engines, social media). Being a data scientist sounds way cooler than being a data analyst. Although the job functions might be similar and overlapping, it all deals with discovering patterns and generating insights from data. It's also about asking intelligent questions about the nature of the data (e.g., Are data points form organic clusters? Is there really a connection between age and cancer?). What about machine learning? Often, the terms data science and machine learning are

used interchangeably. That's because the latter is about "learning from data." When applying machine learning algorithms, the computer detects patterns and uses "what it learned" on new data. For instance, we want to know if a person will pay his debts. Luckily, we have a sizable dataset about different people who either paid his debt or not. We also have collected other data (creating customer profiles) such as age, income range, location, and occupation. When we apply the appropriate machine learning algorithm, the computer will learn from the data. We can then input new data (new info from a new applicant) and what the computer learned will be applied to that new data. We might then create a simple program that immediately evaluates whether a person will pay his debts or not based on his information (age, income range, location, and occupation). This is an example of using data to predict someone's likely behavior.

Possibilities

Learning from data opens a lot of possibilities, especially in predictions and optimizations. This has become a reality, thanks to the availability of massive datasets and superior computer processing power. We can now process data in gigabytes within a day using computers or cloud capabilities. Although data science and machine learning algorithms are still far from perfect, these are already useful in many applications such as image recognition, product recommendations, search engine rankings, and medical diagnosis. And to this moment, scientists and engineers around the globe continue to improve the accuracy and performance of their tools, models, and analysis.

Drawbacks of Data Analysis & Machine Learning

You might have read from news and online articles that machine learning and advanced data analysis can change the fabric of society (automation, loss of jobs, universal basic income, artificial intelligence takeover). In fact, society is being changed right now. Behind the scenes, machine learning and continuous data analysis are at work especially in search engines, social media and e-commerce. Machine learning now makes it easier and faster to do the following:

- Are there human faces in the picture?
- Will a user click an ad? (Is it personalized and appealing to him/her?)
- How to create accurate captions on YouTube videos? (recognize speech and translate into text)
- Will an engine or component fail? (preventive maintenance in manufacturing)
- Is a transaction fraudulent?
- Is an email spam or not?

These are made possible by the availability of massive datasets and great processing power. However, advanced data analysis using Python (and machine learning) is not magic. It's not the solution to all problems. That's because the accuracy and performance of our tools and models heavily depend on the integrity of data and our own skill and judgment. Yes, computers and algorithms are great at providing answers. But it's also about asking the right questions. Those

intelligent questions will come from us humans. It also depends on us if we'll use the answers being provided by our computers.

Accuracy & Performance

The most common use of data analysis is in successful predictions (forecasting) and optimization. Will the demand for our product increase in the next five years? What are the optimal routes for deliveries that lead to the lowest operational costs? That's why an accuracy improvement of even just 1% can translate into millions of dollars of additional revenues. For instance, big stores can stock up certain products in advance if the results of the analysis predict an increasing demand. Shipping and logistics can also better plan the routes and schedules for lower fuel usage and faster deliveries. Aside from improving accuracy, another priority is on ensuring reliable performance. How can our analysis perform on new data sets? Should we consider other factors when analyzing the data and making predictions? Our work should always produce consistently accurate results. Otherwise, it's not scientific at all because the results are not reproducible. We might as well shoot in the dark instead of making ourselves exhausted in sophisticated data analysis. Apart from successful forecasting and optimization, proper data analysis can also help us uncover opportunities. Later, we can realize that what we did is also applicable to other projects and fields. We can also detect outliers and interesting patterns if we dig deep enough. For example, perhaps, customers congregate in clusters that are big enough for us to explore and tap into. Maybe there are unusually higher concentrations of customers that fall into a certain income range or spending level. Those are just typical examples of the applications of proper data analysis. In the next chapter, let's discuss one of the most used examples in illustrating the promising potential of data analysis and machine learning. We'll also discuss its implications and the opportunities it presents.

Chapter 3: Important Python Libraries

You can get a mind map describing software that can be used to analyze data at xmind.net. However, you cannot install all of this software in this chapter. This chapter will explore how to install SciPy, matplotlib, Numpy, and IPython on different platforms. You will also see some sample code that used NumPy.

NumPy is a powerful Python library that features numerical arrays and functions.

SciPy is a powerful Python library, which slightly overlaps NumPy. NumPy and SciPy historically shared their code but were later separated.

Matplotlib provides a plotting library featuring NumPy. You will learn more about matplotlib during data visualization.

IPython contains an architecture for interactive computing. The most critical part of the following project is the IPython shell. You will learn more about the IPython shell in the following chapter.

The installation instructions for the rest of the software will be provided in the rest of the book. When you're done reading this chapter, you will identify pointers on how to determine extra information online if you stumble or are uncertain about the right way to solve problems.

This book uses applications based on Python, so that is why you need to have Python installed on your computer. Some operating systems come with Python already installed, but you will need to check whether the Python version is compatible with the software version you want to install. There are different Python implementations. In the following book, we shall stick to the standard CPython implementation, which is considered compatible with NumPy.

The software application that we will install in this chapter features binary installers for Windows, Mac OS X, and Linux distributions. There are still source distributions in case you prefer that.

Install the Software and Setting Up

You will learn how to install and set up NumPy, matplotlib, and IPython on Linux, Windows, and Mac OS X. Let's explore the process in detail.

For Windows

When you want to install the above Python libraries on a Windows operating system, it's a straightforward process that we dive into detail. You only require to download an installer, and a wizard will direct you through the installation steps. We will provide you with the steps to install Numpy. The steps to install the remaining libraries are the same. The instructions to use are as follows:

1. Get a Windows installer from the SourceForge website or any other legit website. The current versions may change, so just select the one that suits your setup best.
2. Select the right version from the available list.
3. Next, open the EXE installer by double-clicking it.
4. Next, you will see a description of Numpy and its properties. Now, click on the Next button.

Numpy Arrays

Once you are through with the installation of Numpy, now is the time to explore Numpy arrays. Numpy arrays are more efficient compared to Python lists when it comes to numerical operations. Numpy arrays are specialized objects that have broad optimization. Numpy code demands less explicit loops than equivalent Python code.

If we recall some of the mathematics topics you did learn in high school, I am sure you can remember scalars and vectors. For example, the number 2 is a scalar. When you add 3 to 2, you will be doing a scalar addition. We can create a vector out of a group of scalars. In Python programming, you will have a one-dimensional array. This concept can still be extended to a higher dimension. Conducting an operation on two arrays, like addition, can be reduced to a group of scalar computations.

In summary, Numpy is faster than the pure Python code. One thing that is clear, you can acquire the same results whether you use NumPy or not. But the result displayed differs in presentation. Remember that the result from the Numpysum () function doesn't have a lot of commas. This is because you are not working with a Python list but with a NumPy array.

Using IPython as a Shell

Data analysts, engineers, and scientists like to experiment. IPython was built by scientists with the goal of experimentation. The interactive environment provided by IPython is considered by many as a direct solution to MATLAB.

Below is a list of properties of the IPython shell:

- History mechanism
- The pylab switch

- Tab completion, which helps you identify a command
- Access to Python debugger and profiler

Web Scraping Using Python Libraries

In this section, you will scrape the “Rate My Professor” website. A brief description of Rate My Professor website, it’s a website that has ratings of schools, professors, and universities. In this website, you can search for any professor or school and see their ratings before registering their courses. It is a great feature that helps you to learn more about your professor or the university that you wish to join. In this section, you will learn how to scrape and extract crucial professor’s tag. While this is not an illegal process, mass scraping of data from this website may result in your IP address being blacklisted. So, you should only attempt it once or twice, but don’t do it foolishly.

Web Scraping

Also known as scraping, data harvesting, or data extraction is a mechanism used to mine data from websites. In some cases, web scraping can be vital when you get the data that you are searching for straight from the web, but sometimes, it is a bad way to handle it because it’s like stealing the expensive data from the website without their permission. However, you should reduce your scraping process to only once or twice so that you don’t fall in trouble.

The most important libraries for web scraping include:

1. Beautiful soup
2. Requests

Here are the steps that we would be following.

1. First, we import the relevant libraries
2. Finding the URL and keeping it in a variable
3. Sending a request to the website using the requests library
4. Applying the Beautiful Soup library to find the HTML data from the website.
5. Using soup to identify all methods to get the necessary tag that we are searching for.
6. Removing all the HTML tags and converting it to a plain text format.

You may be wondering the type of tags to extract, well in the Rate My Professor website, every professor has his or her respected tags. We will try to extract these tags.

Before we start, ensure that you scrape the data at a low pace, and you can use a VPN service to change your IP address. This prevents your IP address from being blocked. Hope you will follow the instructions.

One thing that you should note is that there is no need to explain each and every line of code because Python code is self-explanatory. But you will not be confused because everything will be made clear in an easy format. So, this guide is written in such a way that everybody can understand regardless of their programming level.

You may find a lot of online tutorials, but this guide is easy to understand because the codes are explained. However, some parts are a mechanical process, wherein you need to follow them.

Let's dive in!

1. **First, import the necessary libraries**

Let us import several libraries such as Beautiful Soup and Requests.

```
import requests
from bs4 import BeautifulSoup
```

2. **Find the URL and store it in a variable**

Let store the professor's URL within a variable called "url". The website URL is Rate My Professor.

3. **Send a request to the website using the requests library**

In the following case, we apply the requests library by passing "url" as the parameter. Be careful that you don't run this numerous times. If you receive like Response 200, then it's a success. If you find something different, then there is something wrong with maybe the code or your browser.

4. **Apply the Beautiful Soup library to get the raw HTML data from the website.**

Here, we apply the BeautifulSoup by passing the page.text as a parameter and applying the HTML parser. You can attempt to output the soup, but displaying the soup doesn't provide you the response. Instead, it contains a huge percentage of HTML data.

Here is a code snippet.

```
soup = BeautifulSoup(page.text, "html.parser")
```

5. **Applying the soup.findAll method to find the respected tag that you are searching for.**

Here is where you will be adding the tags that you're searching for. To find the tag name, all you need to do is to right-click on the relevant tag or click Ctrl-Shift-I on the tag within the webpage. Next, a webpage with the required tag will open for you to your right-hand side.

Next, you can copy the HTML tag and class if available, and then place it within the soup.findAll method. In the following case, the HTML tag is "span" and the class is "tag-box-choosetags".

proftags = soup.findAll("span", {"class": "tag-box-choosetags" })
proftags

6. **Eliminating all the HTML tags and changing it to a plain text format**

In the following step, you need to remove all the HTML tags and change it to a text format. This can be done using the get_text method aligned within the for loop. This changes the HTML into the text format.

for mytag in proftags:
*print(mytag.get_text())****Hilarious (11)***
Caring (9)
Accessible outside class (8)
Amazing lectures (5)
Clear grading criteria (4)
Inspirational (4)
Group projects (3)
Respected (3)
Gives good feedback (2)
EXTRA CREDIT (2)
Participation matters (1)
Lecture heavy (1)
Test heavy (1)
So many papers (1)

Therefore, you get the above information that you are looking for. You get all the tags of the professor. This is how you scrape the data from the internet using Requests and Beautiful Soup libraries.

In summary, Python has three core data science libraries which many others have been built:

- Numpy
- SciPy
- Matplotlib

For simplicity, you can consider Numpy as your one-stop for arrays. Numpy arrays are different from the typical Python lists in many different ways, but a few to recall is that they are faster, consume less space, and have more functionality. It is important to remember, though, that these arrays are of a fixed size and type, which you define at creation. No infinitely appending new values like you want with a list.

SciPy is defined on top of Numpy and provides a lot of the optimization, linear algebra, and statistics functions you will need. Although sometimes, Numpy has similar functionality, SciPy's functionality is better. If you want to compute a correlation coefficient or generate some normally distributed data, then SciPy is the library for you.

Matplotlib

This is probably not grabbing any beauty awards, but it is the major library for plotting in Python. It has a lot of functionality and enables you to have huge control when needed.

2nd generation

The main libraries are great, and you will find yourself using them a lot. There are three 2nd generation libraries, which have been highly developed on top of the main one to give you additional functionality with less code.

Pandas were designed to make data analysis easier in Python. Pandas make it easy to load structured data, compute statistics on it, and slice and dice the data in whichever way you want. It is one of the most indispensable tools during the data exploration and analysis stage. However, it's not good to use it in production because it doesn't scale very well to large datasets. You can gain a huge boost in production by converting your Pandas code to raw Numpy.

Although Matplotlib isn't the best out of the box, Seaborn makes it easy to define beautiful visualizations. It is centered upon Matplotlib, so you can still apply Matplotlib functionality to augment Seaborn charts. It also makes it easier to define complex chart types.

Chapter 4: Data Manipulation

The Python Pandas library is an open-source project that has easy to use tools for data manipulation and analysis. A significant amount of time in any machine learning project will be required to prepare the data and analyze the basic trends and patterns before creating any models.

The Pandas library has risen into a powerhouse of data manipulation tasks in Python because it was built in 2008. With its elaborate syntax and flexible data structure, it's easy to learn and support faster data computation. The development of Numpy and Pandas library has stretched Python's multi-purpose nature to compute machine learning challenges. The acceptance of Python language in machine learning has been critical since then.
This is one of the reasons highlighting the need for you to master these libraries.

In the following chapter, you will learn about how to use Numpy and Pandas libraries for data manipulation from scratch. We shall mix both theory and practical aspects.

First, we will recap the syntax and commonly used functions of the respective libraries. Then later work on a real-life data set.

So, you need to be good at the basics of Python. No further knowledge is required. Also, ensure you have Python installed in your machine.

6 Key Things You Need to Know About Numpy and Pandas

- The data properties of Pandas are designed on top of the Numpy library. In one way, Numpy is a dependency of the Pandas library.
- Pandas is perfect at handling tabular data sets made of unique variables. Besides that, the Pandas library can still be used to perform even the most naïve of tasks such as loading data or performing feature engineering on time series data.
- Numpy is suitable for performing basic numerical computations like median, range, mean, etc. Besides that, it supports the development of multidimensional arrays.
- The Numpy library can also be used to mix C/C++ and Fortran code.
- Keep in mind, Python is a zero-indexing language, unlike R language, where indexing begins at one.
- The best thing about learning Pandas and Numpy is the strong, active community support you gain from the world.

Just to give you a grasp of the Numpy library, we'll quickly revise its syntax structures and some

critical commands such as indexing, slicing, concatenation, etc. All these commands will be useful when using Pandas. Let's get started.

Getting Started with Numpy

First, load the library and confirm its version, just to be sure that we are not using an older version.

```
import numpy as np
np.__version__
'1.12.1'
```

Next, build a list of numbers running from 0 to 9.

```
L = list(range(10))
```

Then convert integers to a string. This method of working with lists is known as a list comprehension.

List comprehension provides a versatile means to deal with list manipulation tasks easily.

Array Indexing

The important thing to remember is that indexing in Python starts at zero.

```
x1 = np.array([4, 3, 4, 4, 8, 4])
x1
array([4, 3, 4, 4, 8, 4])

#assess value to index zero
x1[0]
4

#assess fifth value
x1[4]
8

#get the last value
x1[-1]
4

#get the second last value
x1[-2]
```

8

```
#in a multidimensional array, we need to specify row and column index
x2
array([[3, 7, 5, 5],
 [0, 1, 5, 9],
 [3, 0, 5, 0]])


#1st row and 2nd column value
x2[2,3]
0

#3rd row and last value from the 3rd column
x2[2,-1]
0

#replace value at 0,0 index
x2[0,0] = 12
x2
array([[12, 7, 5, 5],
 [ 0, 1, 5, 9],
 [ 3, 0, 5, 0]])
```

Array Slicing

Now, we'll learn to access multiple or a range of elements from an array.

```
y = np.arange(10)
y
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])


#from start to 4th position
y[:5]
array([0, 1, 2, 3, 4])


#from 4th position to end
y[4:]
```

```
array([4, 5, 6, 7, 8, 9])


#from 4th to 6th position
y[4:7]
array([4, 5, 6])


#return elements at even place
y[ : : 2]
array([0, 2, 4, 6, 8])


#return elements from first position step by two
y[1::2]
array([1, 3, 5, 7, 9])



#reverse the array
y[::-1]
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

Array Concatenation

Many a time, we are required to combine different arrays. So, instead of typing each of their elements manually, you can use array concatenation to handle such tasks easily.

```
#Using its axis parameter, you can define row-wise or column-wise matrix

np.concatenate([grid,grid],axis=1)
array([[1, 2, 3, 1, 2, 3],
 [4, 5, 6, 4, 5, 6]])
```

Until now, we used the concatenation function of arrays of equal dimension. But, what if you are required to combine a 2D array with a 1D array? In such situations, np.concatenate might not be the best option to use. Instead, you can use np.vstack or np.hstack to do the task. Let's see how!

```
x = np.array([3,4,5])
grid = np.array([[1,2,3],[17,18,19]])
np.vstack([x,grid])
```

```
array([[ 3, 4, 5],
 [ 1, 2, 3],
 [17, 18, 19]])

#Similarly, you can add an array using np.hstack
z = np.array([[9],[9]])
np.hstack([grid,z])
array([[ 1, 2, 3, 9],
 [17, 18, 19, 9]])
```

Besides the functions we have learned above, there are other mathematical functions available in the Numpy library, such as divide, multiple, mod, sin, cos, var, min, max, abs, etc. which you can apply to complete basic arithmetic calculations. Be free to go back to Numpy documentation for additional information on such functions.

Let’s proceed to Pandas now. Ensure you follow every line below because it will enable you to perform the computation using Pandas.

Getting Started with Pandas

The load library - pd is just an alias. The pd is used because it's short and literally abbreviates Pandas. Still, you can use any name as an alias.

```
import Pandas as pd
```

Next, create a data frame. Dictionary is used here where keys get converted to column names and values to row values.

```
data = pd.DataFrame({'Country': ['Russia','Colombia','Chile','Equador','Nigeria'],
 'Rank':[121,40,100,130,11]})
data
```

	Country	Rank
0	Russia	121
1	Columbia	40
2	Chile	100
3	Equador	130
4	Nigeria	11

We can perform a quick analysis of any data set using:
data.describe()

	Rank
count	5.000000
mean	80.400000
std	52.300096
min	11.00000
25%	40.000000
50%	100.000000
75%	121.000000
max	130.000000

Remember, describe () method performs summary statistics of integer/double variables. To find complete information about the data set, we can apply the info () function.

Among other things, it shows the data set has 5 rows and 2 columns with their respective names.

#Let's create another data frame.
data = pd.DataFrame({'group':['a', 'a', 'a', 'b','b', 'b', 'c', 'c','c'],'ounces':[4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
data

#Let's sort the data frame by ounces - inplace = True will make changes to the data
data.sort_values(by=['ounces'],ascending=True,inplace=False)

	group	ounces
1	a	3.0
6	c	3.0
0	a	4.0
7	c	5.0
3	b	6.0
8	c	6.0
4	b	7.5
5	b	8.0
2	a	12.0

Still, you can sort the data by not just one column but numerous columns as well.

data.sort_values(by=['group','ounces'],ascending=[True,False],inplace=False)

	group	Ounces
2	a	12.0
0	a	4.0
1	a	3.0
5	b	8.0
4	b	7.5
3	b	6.0
8	c	6.0
7	c	5.0
6	c	3.0

Typically, we get data sets with duplicate rows, which are just noise. As a result, before we train the model, we need to ensure we eliminate such inconsistencies within the data set. Here is how we can remove duplicate rows.

#sort values
data.sort_values(by='k2')

	K1	K2
2	one	3
1	one	2
0	one	1
3	two	3
4	two	3
5	two	4
6	two	4

#remove duplicates - ta da!
data.drop_duplicates()

	K1	K2
2	one	3
1	one	2

0	one	1
3	two	3
4	two	3
5	two	4

Here, we removed duplicates based on matching row values across all columns. Alternatively, we can also remove duplicates based on a particular column. Let's remove duplicate values from the k1 column.

data.drop_duplicates(subset='k1')

	K1	K2
o	one	3
3	two	3

Now, we will learn to categorize rows based on a predefined criterion. It happens a lot while data processing where you need to categorize a variable. For example, say we have got a column with country names, and we want to create a new variable 'continent' based on these country names.

In this case, we will follow these steps:

```
data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon',
 'Pastrami','corned beef', 'Bacon', 'pastrami', 'honey ham','nova lox'],
                     'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
data
```

Now, we want to create a new variable that indicates the type of animal which acts as the source of the food. To do that, first, we'll create a dictionary to map the food to the animals. Then, we'll use map function to map the dictionary's values to the keys. Let's see how is it done.

```
meat_to_animal = {
'bacon': 'pig',
'pulled pork': 'pig',
'pastrami': 'cow',
'corned beef': 'cow',
'honey ham': 'pig',
'nova lox': 'salmon'
}
```

```
def meat_2_animal(series):
 if series['food'] == 'bacon':
 return 'pig'
 elif series['food'] == 'pulled pork':
 return 'pig'
 elif series['food'] == 'pastrami':
 return 'cow'
 elif series['food'] == 'corned beef':
 return 'cow'
 elif series['food'] == 'honey ham':
 return 'pig'
 else:
 return 'salmon'
```

Another method to create a new variable is by using the assign function. With the following tutorial, you can continue to discover the new functions, and you will learn how powerful Pandas are.

```
data.assign(new_variable = data [‘ounces’]*10
```

We constantly find missing values within the data set. A quick approach for inputting missing values is by completing the missing value using any random number. Not only missing values, but you may find a lot of outliers within your data set, which demands replacement.

Let's proceed and learn about grouping data and creating pivots in Pandas. It's an immensely important data analysis method which you'd probably have to use on every data set you work with.

Importing Data

Pandas has tools to help read data from an extensive source. Since the dataset being used is a csv file, you will require to access read_csv function. This function has numerous options for parsing data. In a lot of files, the default choice works well.

```
import pandas as pd
train_values = pd.read_csv('train_values.csv')
train_labels = pd.read_csv('train_labels.csv')
```

For data to be analyzed, it is important to train the values and labels into a single dataframe. Pandas deliver a merge function that will integrate dataframes on either indexes or columns.

Missing Data

Pandas has multiple functions to handle missing data. To begin with, the isna() function can be important to learn the number of missing values present in the data.

The basic function of this appears at each value within a row and column, and it will display True if it is absent and false in case it is not. Therefore, it is possible to create a function that displays the fraction of missing values in every column.

In our dataset, there are no missing values available. But in case it was present, then we could apply a function to replace another value, or a function to remove the rows featuring missing values.

Anytime you apply fillna(), you have several options. You can decide to replace it with a static value which can be a string or a number. Also, you can substitute it with computations. It is true that you will need to apply different techniques for various columns based on the types of data and missing values.

The following code illustrates how you can apply pandas functions to fill the numerical values using mean.

```
train[train.select_dtypes(include=['int64', 'float64']).columns] =
train[train.select_dtypes(include=['int64', 'float64']).columns].apply(lambda x:x.fillna(x.mean()))
```

Visualize the Data

When you plot values in Pandas, it doesn't appear attractive. However, if you want to highlight trends from data, it can always be the most effective means to achieve this.

The basic function for plotting that is normally used is the plt.plot(). Whenever you plot in Pandas, you will need to refer to the matplotlib API. Therefore, it is critical that you use matplotlib first.

This function supports different visualization types, including histograms, boxplots, and scatter plots. Where the plotting function in Pandas becomes critical is when you use it with other data aggregation functions.

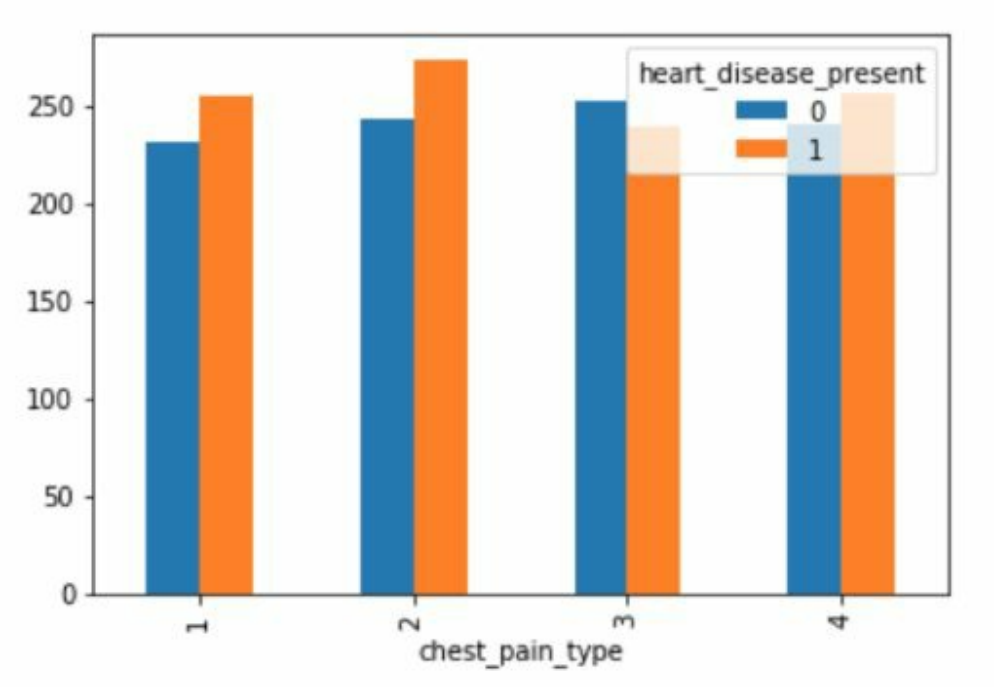
To integrate the value_counts() using bar plot choice provides a rapid visualization for specific features. In the following code, we are exploring the distribution using this method.

```
import matplotlib.pyplot as plt
% matplotlib inlinetrain['thal'].value_counts().plot.bar()
```

When you apply the groupby function, you require to plot mean.

```
train.groupby("slope_of_peak_exercise_st_segment")['resting_blood_pressure'].mean().plot(kind='bar')
```

The Pandas pivot tables can still be used to create visualizations of compiled data. In this example, a comparison is made, and the relationship to heart disease available.



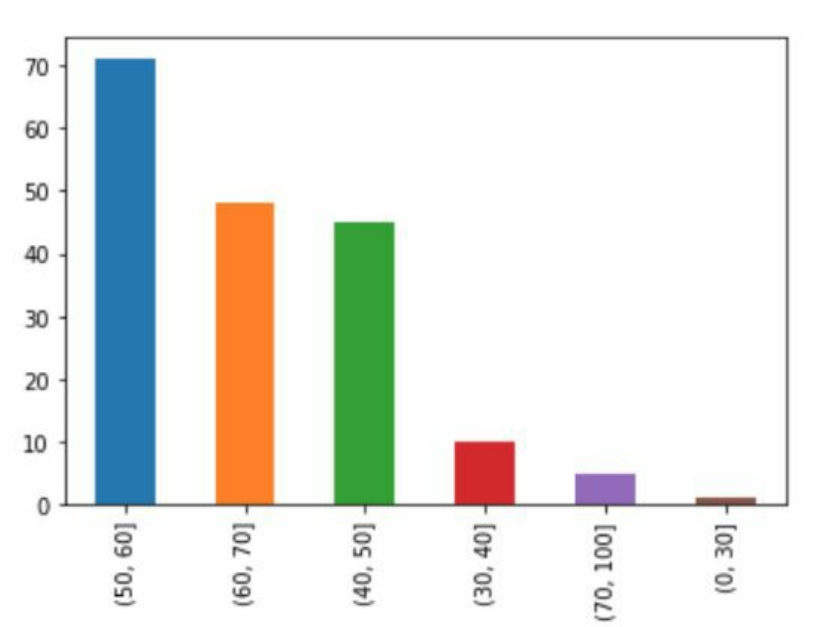
Transformation of Feature

Pandas does have different functions that can be applied in feature transformations.

For instance, the most popular machine learning libraries expect data to be in numerical form. As a result, it is important to change any non-numeric properties. The right way to achieve this is by hot encoding. Pandas have a function for this known as the get_dummies. Once this method is applied to a data column, it changes every unique value into a new binary column.

Another style which a feature may require to be converted for machine learning is called binning. A great example of this data set is the age feature. It can be significant to gather the ages into ranges for the model to learn. Pandas do have a function pd.cut that can be applied.

```
bins = [0, 30, 40, 50, 60, 70, 100]
train['age_group'] = pd.cut(train['age'], bins)
train['age_group'].value_counts().plot(kind='bar')
```



This is only a small introduction to different properties in Pandas for application in the early parts of the machine learning project. There are a lot of features to data analysis, manipulation, and the Pandas library itself. This can be a time-consuming stage, and Pandas deliver different tools and functions that can make the process efficient.

Chapter 5: Data Aggregation

This represents the first part of aggregation and clustering using Pharo DataFrame. This will only handle the basic functionality like clustering a data series using values of a separate series of corresponding size and using aggregation functions to the grouped data structures.

The next iterations will deal with functionality extended based on the targeted scenarios. The implementation is likely to change into something optimized.

Definition of Data Frame

This represent spreadsheet such as data structures that deliver an API for cleaning, slicing, and analyzing data.

In case you want to read more about the DataFrame project, you need to consider the documentation.

Split-Apply-Combine

The split-apply-combine is a technique where you categorize a certain task into manageable parts and then integrate all the parts together.

The data aggregation and grouping facilitates the production of summaries for analysis and display. For example, when you calculate the average values or creating a table of counts. This is a step that adheres the split-apply-combine procedure.

1. Separate the data into sections based on a given procedure.
2. Use the function to every cluster independently.
3. Combine the results using a data structure.

Implementation

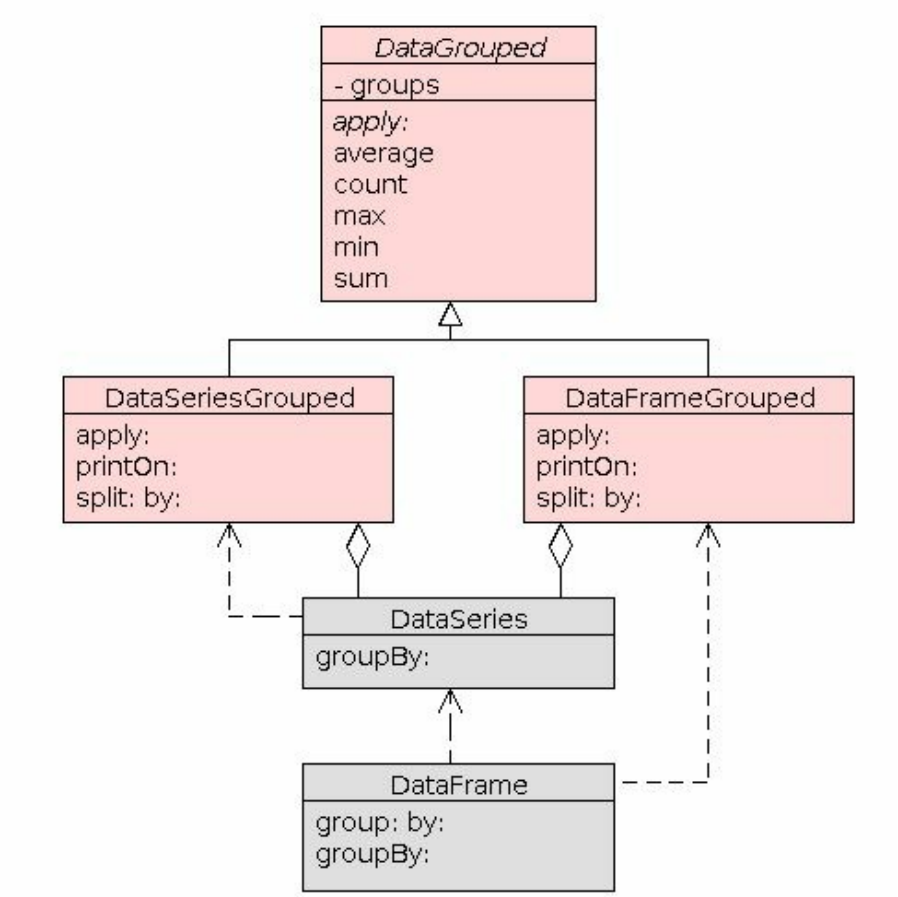
In this part, you will discover how the grouping and aggregation function is being implemented. In case you don’t want these details, you can skip to the next part.

Take, for instance, this message that has been sent to firstSeries object:

```
firstSeries groupBy: secondSeries.
```

Once this message is sent, firstSeries will define an object of DataSeriesGrouped, which divides firstSeries into various subseries depending on the values of secondSeries.

The collection of subseries is later kept as an object of DataSeries whose keys are equivalent to the special values of the secondSeries and values store the subseries of firstSeries. That will match each of those unique values.



When a receiver of a groupBy: message is a DataFrame, it creates an instance of DataFrameGrouped, which splits the data similarly to the way DataSeriesGroup does it, except the values of groups series are sub-data frames, not subseries.

This means groups represent DataSeries that hold keys that match the unique values contained in a series which the data frame is identified. Where the data frame is categorized by a single column, this column is removed from data frame before the grouping. Therefore, this eliminates data duplication because the same values will be kept as keys.

For the case of DataSeriesGrouped, every subseries will be attached to a scalar, and all the following scalars will be joined into a DataSeries. When it comes to the DataFrameGrouped, it will include the block to every column of each sub-data frame and display the eventual matrix of scalars as a new DataFrame.

Aggregation happens with the use of messages. It requires a block as an argument, and uses it on every value of the groups series, and then integrates into a new data structure.

The most common aggregation functions, such as average, min, and max deliver shorter messages. In the following iteration, these messages are useful and work as shortcuts.

```
average
 ^ self apply: [ :each | each average ].
```

However, these messages will carry the optimized implementations of the likened aggregations because it is necessary that these functions are time and memory efficient.

Let’s examine the grouping series.

The easiest example of using this groupBy operator is to classify the values of a series using values of the same size.

```
bill := tips column: #total_bill.
sex := tips column: #sex.bill groupBy: sex.
```

The result of the above query will be an object. This object will separate the bill into two series.

Because a lot of time, you need to classify the group series that resemble columns of a single data frame. There is a useful shortcut.

How to Group Data Frames?

Besides the shortcut for classifying columns. The DataFrame has a method for classifying one of its columns.

The response of the above query will be an object of DataFrameGrouped, keeping two different data frames for smokers and non-smokers.

The smoker column will be removed from the above data frames because its values will be kept as keys within a DataFrameGrouped object. Additionally, the different groups of smokers and non-smokers will enable the complete reconstruction of the smoker column when needed.

The aggregation functions represent the ones that accept different input and display a scalar value that sums up the values of that particular series. These refer to statistical functions such as min, max, stdev, and many more.

Once the data has been combined, next you can use aggregation function to get the integrated data structure that sums up the original data.

```
grouped := tips group: #total_bill by: #day.
grouped apply: [ :each | each average round: 2].
```

Since the grouping is being done to a column of DataFrame by a separate column, the result will be a DataSeries object.

As said before, the DataGrouped presents shortcuts for popularly applied aggregation functions such as count, sum, min, and max. At the moment, these are shortcuts, but in future, they will execute the optimized aggregations that will be used faster.

Once the data frame was grouped into an object of DataFrameGrouped, we can also apply an aggregation function to this object. DataFrameGrouped implements the apply: message in such a way that the function is applied to each column of each sub-data frame, producing the scalar value. These scalars are then combined into a new data frame.

The result of this query will be a data frame containing the number of non-empty cells for each column, corresponding to ‘Male’ and ‘Female’ rows

	total_bill	tip	smoker	day	time	size
Female	87	87	87	87	87	87
Male	157	157	157	157	157	157

Chapter 6: Data Visualization

Data Visualization is an important element for every data scientist. During the early periods of a project, you will need to perform an exploratory data analysis to identify insights into your data. Creating visualizations allows you to simplify things, particularly with a wide dimensional dataset. Towards the end of your project, you need to deliver the final result in a transparent and compelling manner that your audience can understand.

Data Visualization to the End-User

Usually, the data scientist has a role in submitting their insights to the final user. The results can be conveyed in different ways:

- *A single presentation*. In ***the*** following case, the research questions consist of one-shot deals because the business decision extracted from them will direct the organization to a given course for several years to come. For instance, company investment decisions.
- *Do you distribute the goods from two distribution centers or just one*? Where are they supposed to be located for the best efficiency? When the decision is made, the exercise might not be repeated until you retire. In the following case, the results are generated as a report with a presentation as the icing on the cake.
- *A new viewport on data*. The most common example of this is customer segmentation.

For sure, the segments themselves will be send using reports and presentations, but in essence, they comprise of tools, but not the final result itself.
Once a clear and important customer segmentation is identified, it can be supplied back to the database as a new channel on the data from which it was extracted.

From this point, people can create their own reports. For instance, how many products were sold to every customer segment?

- *For a real-time, Dashboard-Your functions as a data scientist doesn't complete once you have the new information. You can send your information back to the database and get done with it. However, when other people start to create reports on the discovered gold nugget, they can interpret it incorrectly and generate reports that don't make sense.*

Since you are the data scientist that found this new information, you need to set the example. In the following case, you need to create the first refreshable report so that the rest can learn from it and use your footsteps. Creating the first dashboard is still a means to reduce the delivery time of your insights to the final user who wants to make use of it daily. By doing this, they already

have something to build upon until the reporting department discovers the time to establish a permanent report regarding the company's reporting software.

You may have discovered that some important elements are at play:

- First, what type of decision are you supporting? Is it strategic or operational? Strategic decisions need you to conduct an analysis and generate a report. But still, operational decisions require the report to be updated often.

- *What is the size of your organization?* For smaller organizations, you will deal with the general cycle. This one ranges from collection to reporting. For bigger teams, reporters could be available to create the dashboards for you. Still, in the last part, creating a prototype dashboard can be relevant because it provides an example and reduces the delivery time.

Matplotlib is a great Python library that can be used to build your Data Visualizations. However, designing the data, parameters, and plots can get messy and tiresome to do regularly.

This section will guide you through data visualizations and create some rapid and easy functions with the help of Python's matplotlib. You will learn how to create basic plots using Seaborn, Matplotlib, and Pandas visualization.

Python offers many graphing libraries that are packed with a lot of features. No matter whether you want to describe an interactive, or complex Python plots, it delivers a powerful library.

To provide some overview, the popular plotting libraries consist of:

- Seaborn. This has an advanced interface and important default styles.
- Plotly. This is important in the development of significant plots.
- Visualization using Pandas. It is an easy to apply interface and has been built on Matplotlib.
- Ggplot. This one relies on R's ggplot2 and applies Grammar of Graphics.

Matplotlib

This is the most common Python library. It is a low library type that has Matlab interface which has more freedom.

Matplotlib is important for creating basic graphs such as bar charts, line charts, and histograms. You can import the following library by using the following line of code:

```
import matplotlib.pyplot as plt
```

Line Chart

The Matplotlib library allows the development of a line chart by applying the plot method. Still, it is possible to create multiple columns using a single graph by plotting and looping every column on the same axis.

Histogram

By using Matplotlib, you can design a histogram using the hist method. In case you relay categorical data like column points, it will help determine the likelihood of each class happening.

Bar Chart

In case you want to show data in a bar chart, then the bar function is useful. The bar chart is not automatically created using the frequency of a category, so you will require to use Pandas to achieve this. The bar chart is useful for grouping data that doesn't spread categories because it can become messy.

Visualization Using Pandas

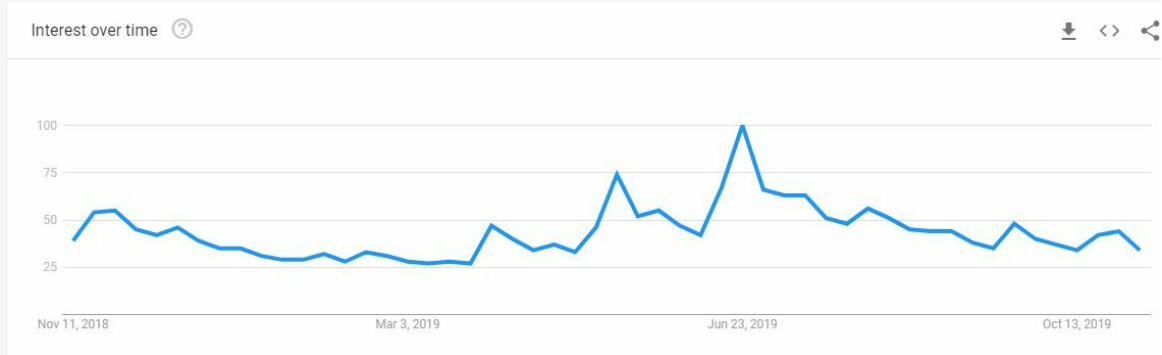
Pandas represent an advanced level of open-source library. It is a simple library that represents data structures and data analysis tools.

Visualizations, with the help of pandas, makes it easy to create data frame and series. Still, it has an advanced level of API than the Matplotlib. As result, minimum code is required for similar results.

If you want to install Pandas, then you require to run the pip command.

The Objective of Visualization

Data communication and exploration are the major focus of data visualization. Once data is visualized, the patterns become visible. You will immediately tell whether there is an increasing trend or the relative magnitude of something in connection to other factors. Rather than tell people the long list of numbers, why not display to them for better clarity?
For instance, let's consider the worldwide trend search on the word 'bitcoin'.



You should see that there is a temporary rise in the bitcoin interest, but it starts to decrease after the peak. Overall, during the peak interval, there's a huge hype connected to the technological and social effects of bitcoin. Again, the following hype decreases because people understand it, or it's a common thing related to hypes.

No matter the situation, data visualization helps us to determine the patterns in a very clear style. Keep in mind the importance of data visualization is to explore data. In the following case, you can quickly choose the patterns as well as the data send to us.

This is critical when you submit it to the public audience. Others may decide to go for a quick brief of the data without rushing into detail. You don't really need to disturb them with texts and numbers. What presents a wide effect is the way you build them using numbers and texts. What sets a big difference is how you define the data so that individuals can quickly recall its importance.

This is where data visualization becomes helpful to allow people to mine data and communicate whatever you are trying to speak.
There are numerous methods of visualizing data. Some of these methods have already been discussed already.

The Simplest Method to Complex Visualization of Data

To create excellent data, visualizations are a powerful skill that every data scientist needs to be aware.

It is more than just creating beautiful charts, representing dataset's information in a way that it is easy for individuals to learn. When you have the right visualization, an individual can quickly learn the patterns and information that is found beneath the data.

In the early stages of a project, you will conduct an exploratory data analysis to generate insights into your data. Creating visualizations will increase your analysis.

At the end of your project, it is vital to submit your final results in a brief and compelling manner such that any audience can be able to read.

There’s no doubt your visualizations to the next stage will let you defeat your next presentation.

This section will explore ways in which you can define an attractive, complex data visualization. You will apply the plotly python library that is excellent in creating interactive visualizations.

Overview of Plotly

Plotly represent an interactive, browser-depended graphic python library. It is a library that allows you to improve the visualization capabilities compared to the standard Matplotlib.

There are two benefits of applying Plotly instead of other Python libraries such as Matplotlib, Pandas, and Seaborn. That is:

1. The ease of application. This will define an interactive plot and other complex graphics. Performing the same operation using other libraries takes a lot of work.
2. It provides additional functionalities.

Since Plotly is designed from D3.js, the plotting capability is powerful than other plotting libraries.

The Sunburst charts and many more are possible using plotly.

Building Attractive Plots Using Plotly

Plotly is useful in building fancy plots.

To start, first, let’s import plotly and its internal graph objects component. You will also import Pandas to load the dataset.

```
import plotly
import plotly.graph_objs as goimport Pandas as pd
```

To read the dataset, you basically write a one-liner in Pandas.

Scatter Plots

For this particular section, we are going to plot a scatter plot for sales price against year built. To achieve that, you will need to define a scatter graph object and store it in a trace.

```
trace = go.Scatter(
 x = data['YearBuilt'],
 y = data['SalePrice'],
 mode = 'markers',
 showlegend = True
)
plot_data = [trace]
```

Then, to plot, you only write a single line.

```
plotly.offline.plot(plot_data, filename='basic-scatter')
```

The following command will create a new tab within your browser with the plot.

Graph interactivity comes automatically built-in with Plotly.

Box Plots

This time, we will look at the box plots.
The process is quite similar. For that reason, we are going to define a graph object, store it into a trace, and then represent it in a browser.

```
import plotly
import plotly.graph_objs as go

import Pandas as pd

data = pd.read_csv('train.csv')

trace = go.Box(
 x = data['YearBuilt'],
 y = data['SalePrice'],
 marker = {'color': 'green'},
 showlegend = True,
)
plot_data = [trace]

plotly.offline.plot(plot_data, filename='basic-box')
```

The box plot will feature attractive properties with box plots. By default, we attain the same zooming, panning, and point of selection. Now that the box plot exists, if you hover around each box plot, it will reveal the following:

- Median
- 1st and 3rd quartiles
- Min and Max values of the data range
- The upper and/or lower fences if there are outliers

Heat Maps

Heat maps are a critical tool for a data scientist. They are effective for displaying the association between multiple feature variables in a single graph plus the relative significance of each relationship.

To demonstrate the way your Heat Maps can be improved with Plotly. We are going to create a correlation matrix of the House Prices dataset as a heat map.

```
import plotly
import plotly.graph_objs as go

import Pandas as pd

data = pd.read_csv('train.csv')

corrmat = data.corr()

trace = go.Heatmap(z=corrmat, x=corrmat.columns.tolist(), y=corrmat.columns.tolist())

plot_data = [trace]

plotly.offline.plot(plot_data, filename='basic-heatmap')
```

Heatmaps in Matplotlib can be somehow difficult because you cannot identify the correct value of each cell_ you can only tell from the color. You can write the code to make it interactive, but that's probably the hassle in Matplotlib.

Plotly provides interactivity beyond the box, so when you plot a heat map, you get an attractive overview and an option to confirm exact values when needed.

Both the pan-and-zoom functionality of Plotly is super clean, providing an easy means to perform a comprehensive exploration from a visual point of view.

These are just to indicate the significance and possibilities of applying plotly. Keep in mind that you can create a publication using quality data visualizations. Additionally, you can change the example codes to your objective. There's no need to invent something unique. You can copy the right sections and apply them to your data.

Probably, in the future, there will be easier and effective methods to build data visualizations, especially when dealing with huge datasets. You can still build animated presentations that can change with time. Whichever way, the main goal of data visualization is to communicate data. You can select other methods, but the goal normally remains the same.
In the following chapter, you have learned general aspects of data visualization. You have learned that data visualization is the practice of understanding data by representing it in a graphical style so that trends may not be seen exposed.
Python provides many different graphic libraries that are packed with lots of different attributes.

In this section, we have looked at Matplotlib, Pandas visualization, and Seaborn. In the coming chapters, we will take a look at the complex topics that are specific to machine learning and complex data analysis. The original goal is to make sure that you know the common features and terms applied in data science.

Chapter 7: Machine Learning

Machine learning is a division of AI that delivers technologies with the ability to learn from past scenarios without any programming. Machine learning is abbreviated as ML. This field is concerned with the creation of computer applications that process data and learn from it.

The learning process starts with observations, direct experience, or examples to build patterns from the data and use these patterns to predict the future. The main goal of machine learning is to enable computers to automatically learn without intervention by humans
With ML, it is possible to analyze massive quantities of data. ML generates useful results, but still we may require different resources to reach this state. Extra time may be required to train the machine learning models.

Machine Learning Algorithms Classifications

Let’s start with the supervised learning

Supervised Learning

For the case of supervised learning, the human is required to provide both the inputs and the outputs which are required and furnish the feedback based on the accuracy of the predictions during the time of training. After completion of the training, the algorithm will have to use what was applied to the next data.

The idea of supervised learning can be said to resemble learning under a teacher’s supervision. The teacher provides examples to the student, and the student comes with new rules and knowledge based on these examples so that to apply it somewhere else.

It is also perfect for you to know the distinction between the regression problems and classification problems. When it comes to regression issues, the target is a numeric value, while in classification, the target is a class. A regression task can help compute the average cost of all houses in London. On the other hand, a classification task will allow you to determine the types of flowers based on the length of the petals and sepals.

Unsupervised Learning

When it comes to unsupervised learning, the algorithms don’t expect to be supplied with the output data. A technique called deep learning, which is an iterative approach, is employed to revise the data and reach a conclusion. This makes them perfect for processing tasks that are complex compared to the supervised learning algorithms. In other words, unsupervised learning

algorithms learn from examples without responses to these. The algorithm finds patterns from the examples on its own.

The supervised learning algorithms work the same way humans determine any similarities between two or more objects. Most of the recommender systems you come across when buying items online work based on unsupervised learning algorithms. In the following case, the algorithms find what to recommend to you what you have bought before. The algorithm has to approximate the type of customers whom you resemble, and a suggestion is derived from that.

This seems to be the essence of true Artificial Intelligence wherein the computer can learn without human intervention. It's about learning from the data itself and trying to find the relationship between different inputs (notice there's no expected output here in contrast to Regression and Classification discussed earlier). The focus is on inputs and trying to find the patterns and relationships among them. Perhaps there are natural clusters, or there are clear associations among the inputs. It's also possible that there's no useful relationship at all.

Reinforcement Learning

This type of learning happens when the algorithm is presented with examples that don't have labels, as it is the case with unsupervised learning. But the example can be achieved with the help of a positive or negative feedback based on the solution suggested by the algorithm. It is hooked with applications in which the algorithm has to make decisions, and these decisions are linked with a consequence. It is same as trial and error in human learning.

Errors become necessary in learning when they are connected with a penalty such as cost, loss of time, and pain. When it comes to reinforced learning, some actions are likely to succeed than others.

ML processes resemble those of data mining and predictive modeling. In both cases, searching through the data is a must so as to draw patterns then adjust the actions of the program accordingly. A great example of ML is the recommender systems. If you buy an item online, you will see an ad that is associated to that item, and that is a great example of ML.

How to Approach a Problem

Many data scientists approach a problem in a binary way. Does the task fall
under Supervised or Unsupervised Learning?

The quickest way to figure it out is by determining the expected output. Are we trying to predict y values based on new x values (Supervised Learning, regression)? Is a new input under category A or category B based on previously labeled data (Supervised Learning, Classification)? Are we trying to discover and reveal how data points aggregate and if there are natural clusters

(Unsupervised Learning, Clustering)? Do inputs have an interesting relationship with one another (do they have a high probability of co-occurrence)?

Many advanced data analysis problems fall under those general questions. After all, the objective is always to predict something (based on previous examples) or explore the data (find out if there are patterns).

What is Deep Learning

Deep learning refers to a sub-branch of ML that involves algorithms that are motivated by the function and structure of the brain known as the artificial neural networks. This makes machines to do what is natural to humans, and that is, learn from the past. This is the system behind the idea of driverless cars.
It is with the help of deep learning that a computer can perform classification tasks. Deep learning models can attain accuracy, which sometimes exceeds human-level performance.

Neural Networks with Scikit-Learn

Neural networks refer to ML technology that attempts to copy the natural biological neural networks operate. Humans have the capacity to identify patterns with a very high degree of accuracy. Anytime you see a cow, you can immediately recognize that it is a cow. This also applies to when you see a goat. The reason is that you have learned over a period of time how a cow or a goat looks like and what differentiates between the two.

Artificial neural networks refer to systems that try to imitate the capabilities of human learning via a complex architecture that resembles the nervous system of a human being.
The idea behind artificial neural networks is actually old. But recently it has undergone massive reemergence that many people (whether they understand it or not) talk about it.

Why did it become popular again? It's because of data availability and technological developments (especially massive increase in computational power). Back then, creating and implementing an ANN might be impractical in terms of time and other resources. But it all changed because of more data and increased computational power. It's very likely that you can implement an artificial neural network right in your desktop or laptop computer. And also, behind the scenes, ANNs are already working to give you the most relevant search results, most likely products you'll purchase, or the most probable ads you'll click. ANNs are also being used to recognize the content of audio, image, and video.

Many experts say that we're only scratching the surface, and artificial neural networks still have a lot of potential. It's like when an experiment about electricity (done by Michael Faraday) was performed, and no one had no idea what use would come from it. As the story goes, Faraday told that the UK Prime Minister would soon be able to tax it. Today, almost every aspect of our lives

directly or indirectly depends on electricity.
This might also be the case with artificial neural networks and the exciting field of Deep Learning (a subfield of machine learning that is more focused on ANNs).

The Structure of Neuron

A neutron is made up of the cell body, having a number of extensions from it.
Majority of these are in the form of branches commonly known as “dendrites”.

A long process or a branching exists, and this is referred to as the “axon”.
The transmission of signals begins at a region in this axon, and this region is known as the “hillock”.

The neuron has a boundary, which is known as the “cell membrane”. A potential difference exists between the inside and the outside of the cell membrane. This is known as the “membrane potential”.
If the input becomes large enough, some action potential will be generated.
This action potential then travels will then travel down the axon and away from the cell body.
A neuron is connected to another neuron by synapses. The information leaves the neuron via an axon and is then passed to the synapses and to the neuron, which is expected to receive it. Note that a neuron will only fire once the threshold exceeds a certain amount. The signals are very important as they are received by the other neurons. The neurons use the signals or the spikes for communication. The spikes are also responsible for encoding the information which is being sent.

Synapses can either be inhibitory or excitatory. When spikes arrive at the excitatory synapse, the receiving neuron will be caused to fire. If the signals are received at an inhibitory synapse, then the receiving neuron is inhibited from firing.
The synapses and the cell body usually calculate the difference in the incoming inhibitory and excitatory inputs. If this difference is found to be too large, the neuron will be made to fire.

Back Propagation

To ensure that a neural network completes a problem, the units must be changed so that we can limit the error between the original output and the required output. In other words, the difference of the weights must be calculated by the network. To simplify everything, the network has to follow up with the changes in error. The backpropagation algorithm is popularly used in the error computation.

In case you have the network units as linear, then this particular algorithm will be easy to master. For the algorithm to compute the error difference of the weights, it requires to first understand the rate at which the error changes because the activity level is being changed.

For the case of the output units, the error difference is found by computing the difference between the real and the target output.
To compute the error change rate for the unknown measurement, make sure all the weights between the hidden unit and output units are calculated.

We can proceed and multiply the weights using the error derivatives in the weights, and then the product is compiled. The sum you find will be equivalent to the error change rate for the hidden unit. Once you get the error change rate in the weights of the hidden layer which is just before the output layer, we will manage to calculate these error changes for the other layers.

The computation for this will happen from one layer to the next, and in the direction which is different from the one transmitted through network.
This indicates where the name “back propagation” originates from. After the error change rate has been determined for some unit, the error difference for the weights for all the connections of the weight can be computed easily. The error difference for the weights can used by multiplying the rate of error change with activity through incoming connection.

Scikit-Learn

Scikit-learn offers users with various algorithms with the help of the Python interface. The library itself was designed in Python, and some of its algorithms were built in Cython to provide better performance. Scikit-learn is a better library for creating machine learning models. This library is open source and is supported by BSD license.

The Neural Networks Using TensorFlow

TensorFlow refers to a Google framework for the development of a deep learning model. Tensor flow depends on data-flow graphs for numerical calculation. TensorFlow has allowed machine learning to become easy. It simplifies the process of getting data, creating predictions, and changing future results.

TensorFlow

The artificial neural network features a simple structure that requires the multiplication of matrices. The inputs are first multiplied with random weights and then relayed through an activation function. Next, the output values are extracted for making predictions.

This demonstrates that this network is far from the truth. The loss metric is used. A higher loss function implies a dumper model. To enhance network knowledge, some optimization has to be carried out by regulating the weights. Stochastic gradient is applied to change the values of the weights in the correct direction. Once the weights have been modified, the network can apply a

new batch of data to test new knowledge.

The error may be lower than what you had initially, but not small enough. The optimization procedure should be implemented iteratively until the error is reduced, such that no additional knowledge can be extracted.

Despite that, the challenge with this particular model is that it has no memory. This implies that the input and output are independent.

What this shows is that the model doesn't care about what took place. This sparks several questions as when you need to predict time series because the network will require to have information about past words or historical data. To solve this issue, a new architecture known as a recurrent neural network was created.

Preparing the Environment

Before you get into the practical side of machine learning and deep learning, you need to have two libraries running in your computer. The libraries include:

- Scikit-Learn
- TensorFlow

Installing Scikit-Learn

Scikit-learn works in Python 2.7 and above. Before you can install Scikitlearn, make sure that you have Numpy and SciPy libraries already installed.

Additionally, make sure that you have the latest versions of the above libraries. Once you install the above libraries, you can proceed to install Scikit-learn on your computer. The process of installation can be achieved with the help of pip. This is a tool that has Python. If you want to install scikit-learn, first run the following command on your terminal.

```
pip install scikit-learn
```

This installation should proceed until it reaches completion. Again, you can use conda to install the scikit-learn. You can do this by running the following command:

```
conda install scikit-learn
```

Once you are through with the installation of scikit-learn, you require to import it into your Python program to apply algorithms. This can be achieved with the help of import statements, as shown below:

Import Scikitlearn

In case the command runs without any error, understand that the installation of scikitlearn went successful. However, if the command sends an error, then the installation was not successful. Now, you can use the scikit-learn to build your own machine learning models.

Installing TensorFlow

TensorFlow has APIs for programming languages such as C++. Java, and it comes packed with a third-party package for R language. You will learn how to install TensorFlow on Windows. For Windows, TensorFlow can be installed using pip.

The native pip will install the TensorFlow on your system without the need to go through a virtual environment. Keep in mind that the installation of TensorFlow using pip interface with other Python installations on your system. But, the best thing is that you only need to run a single command and TensorFlow will be installed on your computer. Additionally, once TensorFlow is installed through pip, users will be enabled to run the TensorFlow programs from the directory they want.

If you want to install TensorFlow using Anaconda, you may need to establish a virtual environment. But within the Anaconda itself, it is advised that you install TensorFlow through pip install command instead of the conda install command.

Make sure that you have Python3.5 + installed. Python3 comes with a pip3 program which can be used in the installation of TensorFlow. Therefore, we need to apply the pip3 install command to facilitate the installation.
This same command will install TensorFlow on your Windows system. Still, it is possible to install TensorFlow using the Anaconda package. The pip comes installed with Python, but Anaconda does not. In other words, for you to install TensorFlow using Anaconda, you will need to have the Anaconda installed. So, navigate to the Anaconda website, and you will find all the instructions to guide you on the installation.

Once you have the Anaconda installed, then you can search for a package called conda, which is better for the management of virtual environments and installation of packages. To use the above package, you need to start the Anaconda.
Go to Windows, select Start, and click “All Programs”, expand the “Anaconda…” folder. This should start the anaconda on your system. If you want to see the details of the conda package.

Run the following command on the terminal.

```
conda info
```

This should return more details regarding the package manager.

There is something special about the Anaconda. It allows us to develop a virtual Python environment with the help of the conda package. This particular virtual environment is a single copy of Python with the ability to maintain its own files, directories, and paths so that a person can work with particular versions of Python or other libraries affecting Python projects.

Virtual environments offer a means of separating projects and eliminate problems that may come up because of version requirements and different dependencies across multiple components. Keep in mind that the above virtual environment will remain different from your normal Python environment, implying that the packages running in the virtual environment will not impact the ones you have within your python usual environment.
You need to build a virtual environment for the TensorFlow package. This can be conducted with the help of conda create command. The command uses the following syntax:

```
conda create -n [environment-name]
```

In the above example, you need to assign the above environment the name tensoreviron. You can create it by running this command:

```
conda create -n tensorenviron
```

You will be requested to let the process of building the environment to proceed or not. Enter “y” to accept and hit the enter key on the keyboard. The installation will proceed successfully.
Once you create an environment, you need to activate it so that it is possible to use it. The activation can happen with the help of the activate command, followed by the environment name.Now that you have the TensorFlow environment activated, you can proceed to install the TensorFlow package in it. You can achieve that by running the command below.

```
conda install tensorflow
```

You will get a complete list of packages that will be installed with the TensorFlow package. You will be asked to allow the installation of this package. Just type “y” and press the enter key on your computer keyboard. The installation of the above packages will start immediately. Remember that the installation procedure may take longer, so you need to be patient. Despite that, the speed of your internet connection will determine how long this process is going to take.

The steps of the installation process will also be displayed in the prompt window. After a few minutes, the installation process will finish, and it will be time for you to confirm whether the installation was successful. You can do this by navigating to the Python’s import statement. The statement should be executed from the Python terminal. While still on the anaconda prompt, enter the phrase Python and press the enter key. This should direct you to the Python terminal. Now execute the following command.

```
import tensorflow as tf
```

If the package was not installed successfully, you would get an error, otherwise, the installation of the package was successful.

Chapter 8: Artificial Neural Networks

For us, humans, it’s easy for us to identify objects and digits. It’s also easy for us to understand

the meaning of a sentence or piece of text.
However, it is a completely different case with computers. What's trivial and automatic for us might be an enormous task for algorithms and computers.

In contrast, computers can achieve long and complex mathematical computations while we humans are terrible at it. It's interesting that the abilities of humans and computers are complementary. However, the natural step is to replace humans at what they are expert at.

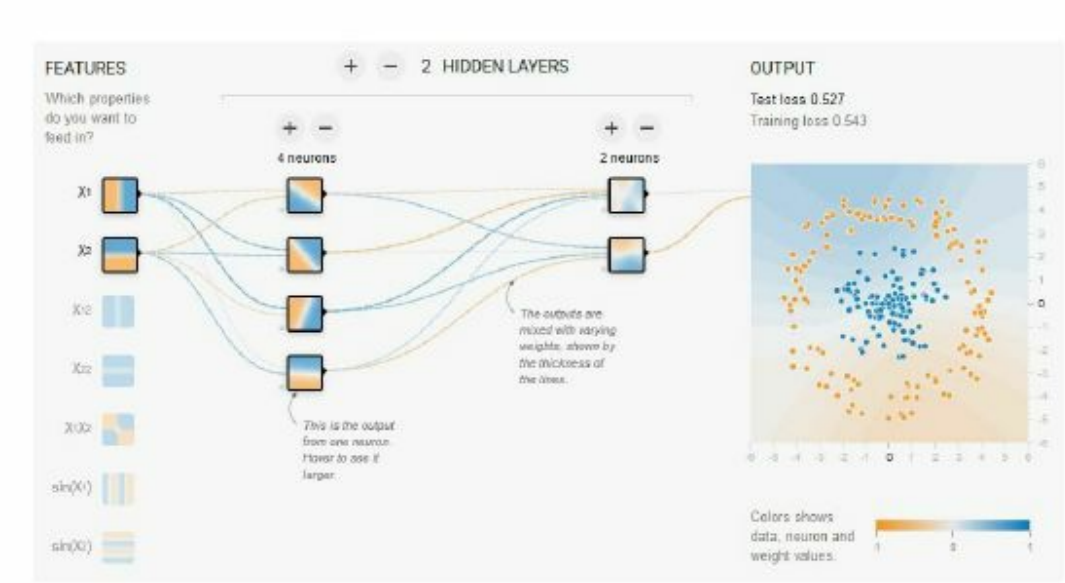
In the future, we might not be able to tell the difference whether whom we are talking to is human or not.

How the Brain Works

The most popular and promising style is the application of artificial neural networks; these are loosely motivated by how our brains and neurons work. The prevailing model concerning how the brain functions is by neurons processing and sending signals.

While it's not a 100% accurate method concerning the brain and neurons, this model is important for most applications.

This is the issue with artificial neural networks where there are neurons in one of few layers receiving and sending signals. Below is a basic demonstration from TensorFlow playground:



Remember that it began with the inputs, and then they are linked with 2 "hidden layers" of neurons.

Finally, there's a result wherein the data was already computed iteratively to define important models or generalization. In most cases, the way artificial neural networks function is similar to how supervised learning functions. When it comes to ANNs, we like to accept a large number of training examples and then create a system that supports learning from the above examples.

At the time of learning, the ANN automatically infers rules for recognizing an image, audio, text, or any other kind of data.

.

As you may have already noticed, the accuracy of recognition heavily relies on the quantity and quality of our data. At the end of the day, it's Garbage In Garbage Out. Artificial neural networks tend to learn from the data fed into it.

We may still boost the accuracy and performance via means other than enhancing the quality and quantity of data.

Constraints and Opportunities

The notion behind ANN dates back long ago. However, in recent times it has undergone huge reemergence that most people discuss about it.

Why did it increase in popularity? The reason is the availability of data and technological development. Long ago, building and executing an ANN was impractical based on resources and time.

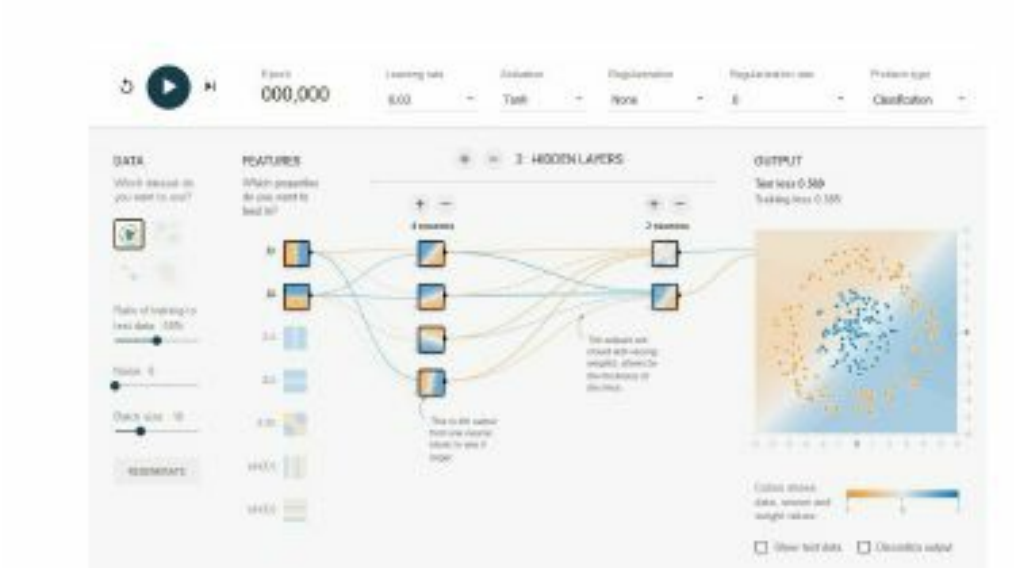
However, it all changed because of the increased speed of computation and data. It is likely that you can execute an artificial neural network right in your laptop computer.

Additionally, ANNs is already working behind the scenes to present the most important search results. ANNs are also being used to identify the content of image, video, and audio.

Most professionals believe that we are only lightly scratching ANN. It's like when an experiment about electricity was executed, and no one had an idea what function would emerge from it.

Let's See an Example

With the presence of TensorFlow playground, it's possible to identify a quick idea of how it operates. You can visit their website and record separate words such as activation, learning rate , and regularization.



Select the “Play” button and wait to see a cool animation. Be careful to the output at the far right. After some time, it will appear this:

The connections increase in clarity as well as other features. Keep in mind that the output has a clear Blue region.

This might be a classification problem wherein the blue dots are part of the Class A while the orange ones are part of the Class B.

As the ANN runs, remember that the division between Class A and Class B becomes clearer. And that is because the technology keeps learning from the training examples. As learning becomes strong, the classification becomes accurate.

By learning more about TensorFlow playground, it provides the easiest way to understand how neural networks work. It’s a rapid visualization even it is not 100% accurate representation.

Therefore, we can understand the hidden layers, features, and output.

We can still introduce some changes in the learning rate, the ratio of training to test data, and the number of hidden layers. For example, we can set the number of hidden layers to 3 and adjust the learning rate to 1. You will see something like this:

If you press the Play button and allow it to run for some time, the image will remain like this:

Keep an eye on the output. You should see that the Classification appears worse. However, surrounding most of the yellow points under the Yellow region, there seems to be a lot of misses. This happens because of the parameter change.

For example, the Learning Rate has a big effect on the level of accuracy and striking the correct convergence. If you make the learning rate to be low, convergence may take a lot of time. However, if the learning rate is very high, we may not attain the convergence because we have overshot it and missed it.

There are different methods to attain convergence within an appropriate time. Learning Rate is simply right, a lot of hidden layers, perhaps fewer or more features to apply. However, over-optimizing everything may not make economic sense. It's right to set a clear goal at the beginning and stick to it.

In case there are great opportunities that arise, you may want to enhance the parameters and boost the performance of the model.
However, if you want to get some idea on the appearance of ANN in python, you can take a look at the following code:

```
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in xrange(60000):
l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
l2_delta = (y - l2)*(l2*(1-l2))
l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
syn1 += l1.T.dot(l2_delta)
syn0 += X.T.dot(l1_delta)
```

This is a simple example. When it comes to the real world, ANN would appear long and complex when designed from scratch. Luckily, the way you work with them is becoming more "democratized," which implies people with limited technical backgrounds would manage to take advantage of them.

Chapter 9: How to use Scikit-Learn

Now that you are done with the installations, you can begin to use the libraries. We will begin with the Scikit-Learn library.
To be able to use scikit-learn in your code, you should first import it by running this statement:

import sklearn

Loading Datasets

Machine learning is all about analyzing sets of data. Before this, we should first load the dataset into our workspace. The library comes loaded with a number of datasets that we can load and work with. We will demonstrate this by using a dataset known as *Iris*. This is a dataset of flowers. The following code shows how we can use scikit-learn to load the dataset:

```
# Import scikit-learn library
from sklearn import datasets
# Load iris dataset
iris= datasets.load_iris()
# Confirm by printing the shape of the data
print(iris.data.shape)
```

Simple Linear Regression

We need to use our previous example, which is, predicting the number of marks a student will score in a test depending on the number of hours they have studied for the test. It is a simple linear regression task since we only have two variables.

Import Libraries

Run the following Python statements tom import all the necessary libraries:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

This file is saved in an MS Excel file named *student_marks.csv*. It's in the directory where my Python scripts are found, so no need to give the path leading to the file. The .csv extension in the filename shows that it is a comma-separated values file.

The following statement will help us to import the dataset into the workspace.
We are using the Pandas library (we imported it as *pd*) for this:

dataset = pd.read_csv('student_marks.csv')

We can now explore the dataset to know more about it and see what it has.
Go directly to the Python terminal and type this:

dataset.shape

Which means that the dataset has 25 rows and 2 columns.
To see the first five rows of the data, call *head()* function.

However, you may get an error when you attempt to print the data. The cause of the error could be that Pandas is looking for the amount of information to display, so you should provide *sys* output information.
The error can be solved by modifying your code to the following:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sys
sys.__stdout__ = sys.stdout
dataset = pd.read_csv('student_marks.csv')
print(dataset.head())
```

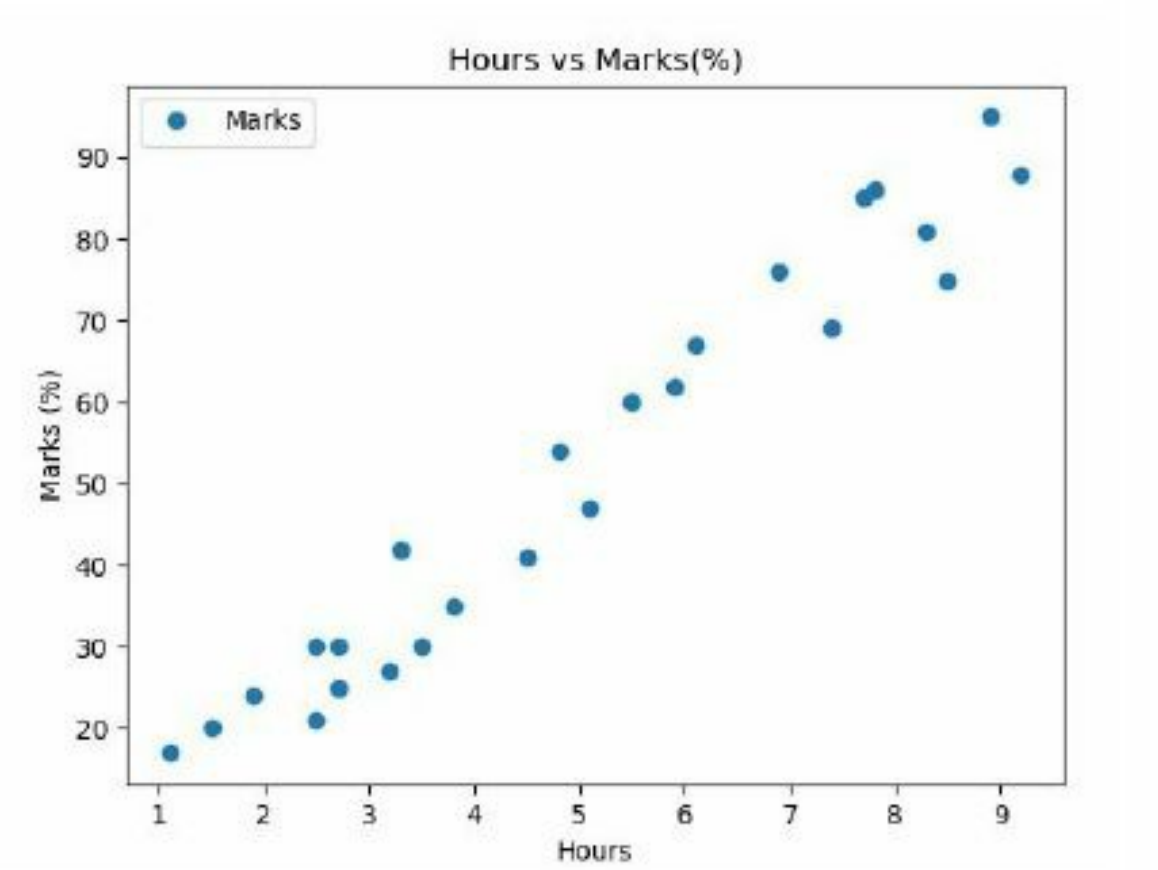
We have simply provided the information to the *sys* library.
To see the statistical details of the dataset, we call the *describe()* function as follows:
dataset.describe()

We can now plot the data points on a 2-D graph and see how they are distributed. You no appreciate why we imported the *Matplotlib* library. The following code will help you to plot the

data points:

```
dataset.plot(x='Hours', y='Marks', style='o')
plt.title('Hours vs Marks(%)')
plt.xlabel('Hours')
plt.ylabel('Marks (%)')
plt.show()
```

The code returns the following plot:



We have called the *plot()* function provided by the Pandas library. We passed the column names to this function, and it was able to create and display the plot. The *show()* function helped us to display the plot.

Data Preparation

The preparation of the data should involve subdividing it into labels and *attributes*. Attributes should create independent variables, and labels create dependent variables.

Our dataset has only two columns. We are predicting Marks based on Hours. This means Hours will form the attribute while Marks will form the label. The attributes and labels can be extracted by running the following code:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

The X variable will store the attributes. Note that we have used -1 because we need all columns to be assigned to attributes except the last one, that is, Marks. The y variable will store the labels. Here, we have used 1 since the column for Marks is at index 1. Remember that column indexes begin at index 0. At this point, we have the attributes and labels for the dataset. We need to divide our data into two sets, namely the *training* and *test* sets. The Scikit- Learn library provides us with a method named "train_test_split()" which can be used for this.

Training the Algorithm

We will be training the algorithm using the *LinearRegression* class, which must be imported from Scikit-Learn. The import can be done as follows:

from sklearn.linear_model import LinearRegression

Now that we have imported the class, we need to instantiate it and give the instance the name *linear_regressor*. This is demonstrated below:

linear_regressor = LinearRegression()

Let us now call the *fit()* method and pass the training data to it:

linear_regressor.fit(X_train, y_train)

As we had stated earlier, Linear Regression works by finding the best values for the slope and the intercept. This is what we have done above. These two have been calculated, so we only have to view their values.
To see the intercept, run the following command:

print(linear_regressor.intercept_)

Predicting

In the training done above, we have created a linear regression model, which is the equation. The values for the slope and the intercept are known. We can make predictions based on the data we preserved as the training set. The following statement helps us make predictions from the test data:

pred_y = linear_regressor.predict(X_test)

We have simply created a numpy array named *predict_y*. This will have all the predicted values for y from the input values contained in the *X_test* series. We now have the actual values for the X_test as well as the predicted values. We need to compare these two and see the amount of similarity or difference between the two. Just run the following code:

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': pred_y})
print(df)
```

The model is not accurate, but the values are close to each other.

Evaluating the Accuracy

We now need to evaluate the accuracy of our algorithm. We need to determine how well the algorithm performed on the dataset. When it comes to regression algorithms, three evaluation metrics are used. These include the following:

1. Mean Absolute Error.
2. Mean Square Error.
3. Root Mean Squared Error.

Multiple Linear Regression

You now know how to do a Linear Regression when you have two variables.
However, this is not a true reflection of what we have in the real world. Most of the problems the world is facing involve more than two variables. This explains why you need to learn Multiple Linear Regression. The steps between the two are almost the same. However, the difference comes when it comes to evaluation. When evaluating the multiple linear regression model, we need to know the factor with the highest impact on the output variable. We also need to determine the relationship between the various variables.

We need to demonstrate this by prediction the consumption of fuel in US states. We will consider factors like per capita income, gas taxes, paved highways, and the proportion of persons who have a driver’s license.
Let us first import the libraries that we need to use:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Again, you may get an error for trying to print the contents of the dataset. Use the method we used previously to solve the problem. You should have the following code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sys
sys.__stdout__ = sys.stdout
dataset = pd.read_csv('fuel_consumption.csv')
print(dataset.head())
```

Data Preparation

In the example of simple linear regression, we subdivided the data into attributes and labels. In this case, we will be directly using the column names for this purpose. Run the code given below:

```
X = dataset[['Tax', 'Income', 'Highways',
'Licence']]
y = dataset['Consumption']
```

We have four columns for the attributes (the independent variables) and one column for the label (the dependent variable).

Let us now subdivide the data into training and test sets. 80% of the data will be used as the training set while the remaining 20% will be used as the test set:
First, let us import the *train_test_split()* method from Scikit-Learn:

from sklearn.model_selection import train_test_split

Run the following command to do the division of the data:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

Training the Algorithm

We now need to train the algorithm. This can be done by calling the *fit()* method, as we did previously. Let us first import the *LinearRegression* class from the Scikit-Learn library:

from sklearn.linear_model import LinearRegression

Next, create an instance of the above class then use it to call the *fit()* method:
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

Note that this is a multiple linear regression and we have many
variables. The linear regression model has to find the optimal coefficients for each attribute. You can see the chosen coefficients by running the following command:

coeff = pd.DataFrame(linear_regressor.coef_, X.columns, columns=['Coefficient'])
print(coeff)

	Coefficient
Tax	-40.016660
Income	-0.065413
Highways	-0.004741
Licence	1341.862121

This means that any unit increase in fuel tax will lead to a decrease of 40.02 million gallons in gas Consumption. Also, a unit increase in the proportion of the population with Driver's license will lead to an increase of 1.342 billion gallons in gas Consumption. The results also show that average Income and Highways have a very small impact on gas Consumption.

Predicting

We will make the predictions using the test data. The following script can help us do this:

pred_y = linear_regressor.predict(X_test)

At this point, you have the actual X_text values as well as the predicted values. We need to

perform a comparison between these two to determine their similarities and differences. This can be done by running the following script:

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': pred_y})
print(df)
```

Evaluating the Accuracy

We now need to evaluate the algorithm in terms of performance. This can be done by determining the values for the various types of errors. These include the MAE, RMSE, and MSE. This requires us to firm import the *metrics* class from Scikit-Learn:

```
from sklearn import metrics
```

The calculation of the error values can then be done using the following script:

```
print('MAE:', metrics.mean_absolute_error(y_test, pred_y))
print('MSE:', metrics.mean_squared_error(y_test, pred_y))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_y)))
```

The value for the root means the square error is 68.31, as shown above. This is slightly greater than 10% of the mean value for gas consumption in all the states. It is true that our algorithm was not very much accurate, but we can still use it to make predictions. The big error could be brought by a number of factors. Maybe there was a need for more data. We have used data for only one year. Having data collected over multiple years could have helped us improve the accuracy of the model. Also, we had assumed that our data has a linear relationship. This could not be the case, hence the big error. I recommend that you visualize the data to see whether this is true. Also, the features we have used could not be correlated.

Chapter 10: K-Nearest Neighbors Algorithm

The KNN algorithm is highly used for building more complex classifiers. It is a simple algorithm, but it has outperformed many powerful classifiers. That is why it is used in numerous applications data compression, economic forecasting, and genetics. KNN is a supervised learning algorithm, which means that we are given a labeled dataset made up of training observations (x, y) and our goal is to determine the relationship between x and y. This means that we should find a function that x to y such that when we are given an input value for x, we are able to predict the corresponding value for y. The concept behind the KNN algorithm is very simple. We will use a dataset named Iris. We had explored it previously. We will be using this to demonstrate how to implement the KNN algorithm.

First, import all the libraries that are needed:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Splitting the Dataset

We need to be able to tell how well our algorithm performed. This will be done during the testing phase. This means that we should have training and testing data. The data set should be divided into two parts. We need to split the data into two parts. 80% of the data will be used as the training set, while 20% will be used as the test set. Let us first import the *train_test_split* method from Scikit-Learn.

Feature Scaling

Before we can make real predictions, it is a good idea for us to scale the features. After that, all the features will be evaluated uniformly. Scikit-Learn comes with a class named *StandardScaler*, which can help us perform the feature scaling. Let us first import this class.

We then instantiate the class then use it to fit a model based on it:

```
feature_scaler = StandardScaler()
feature_scaler.fit(X_train)
X_train = feature_scaler.transform(X_train)
X_test = feature_scaler.transform(X_test)
```

The instance was given the name *feature_scaler*.

Training the Algorithm

With the Scikit-Learn library, it is easy for us to train the KNN algorithm. Let us first import the *KNeighborsClassifier* from the Scikit-Learn library:
from sklearn.neighbors import KNeighborsClassifier

The following code will help us train the algorithm:

```
knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)
```

Note that we have created an instance of the class we have created and named the instance *knn_classifier*. We have used one parameter in the instantiation, that is, *n_neighbors*. We have used 5 as the value of this parameter, and this basically, denotes the value of K. Note that there is no specific value for K, and it is chosen after testing and evaluation. However, for a start, 5 is used as the most popular value in most KNN applications. We can then use the test data to make predictions. This can be done by running the script given below:

pred_y = knn_classifier.predict(X_test)

Evaluating the Accuracy

Evaluation of the KNN algorithm is not done in the same way as evaluating the accuracy of the linear regression algorithm. We were using metrics like RMSE, MAE, etc. In this case, we will use metrics like confusion matrix, precision, recall, and f1 score. We can use the *classification_report* and *confusion_matrix* methods to calculate these metrics. Let us first import these from the Scikit-Learn library: from sklearn.metrics import confusion_matrix, classification_report
Run the following script:

```
print(confusion_matrix(y_test, pred_y))
print(classification_report(y_test, pred_y))
```

The results given above show that the KNN algorithm did a good job of classifying the 30 records that we have in the test dataset. The results show that the average accuracy of the algorithm on the dataset was about 90%. This is not a bad percentage.

K Means Clustering

Let us manually demonstrate how this algorithm works before implementing it on Scikit-Learn:

Suppose we have two-dimensional data instances given below and by the name D:

```
D = { (2,3), (10,12), (12,15), (5,10), (30,45), (85,70), (71,80), (90,78), (22,25), (80,82) }
```

Our objective is to classify the data based on the similarity between the data points.

We should first initialize the values for the centroids of both clusters, and this should be done randomly. The centroids will be named c1 and c2 for clusters C1 and C2 respectively, and we will initialize them with the values for the first two data points, that is, (5,3) and (10,15) . It is after this that you should begin the iterations. Anytime that you calculate the Euclidean distance, the data point should be assigned to the cluster with the shortest Euclidean distance. Let us take the example of the data point (5,3):

```
Euclidean Distance from the Cluster Centroid c1 = (5,3) = 0
Euclidean Distance from the Cluster Centroid c2 = (10,15) = 13
```

The Euclidean distance for the data point from point centroid c1 is shorter compared to the distance of the same data point from centroid c2. This means that this data point will be assigned to the cluster C1 The distance from the data point to the centroid c2 is shorter; hence, it will be assigned to the cluster C2. Now that the data points have been assigned to the right clusters, the next step should involve the calculation of the new centroid values. The values should be calculated by determining the means of the coordinates for the data points belonging to a certain cluster. If for example for C1 we had allocated the following two data points to the cluster:

(5, 3) and (24, 10). The new value for x coordinate will be the mean of the two:
x = (5 + 24) / 2

x = 14.5
The new value for y will be:
y = (3 + 10) / 2
y = 13/2
y = 6.5

The new centroid value for the c1 will be (14.5, 6.5).

This should be done for c2, and the entire process is repeated. The iterations should be repeated until when the centroid values do not update anymore. This means if, for example, you do three iterations, you may find that the updated values for centroids c1 and c2 in the fourth iterations are equal to what we had in iteration 3. This means that your data cannot be clustered any further. You are now familiar with how the K-Means algorithm works. Let us discuss how you can implement it in the Scikit-Learn library. Let us first import all the libraries that we need to use:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
```

Data Preparation

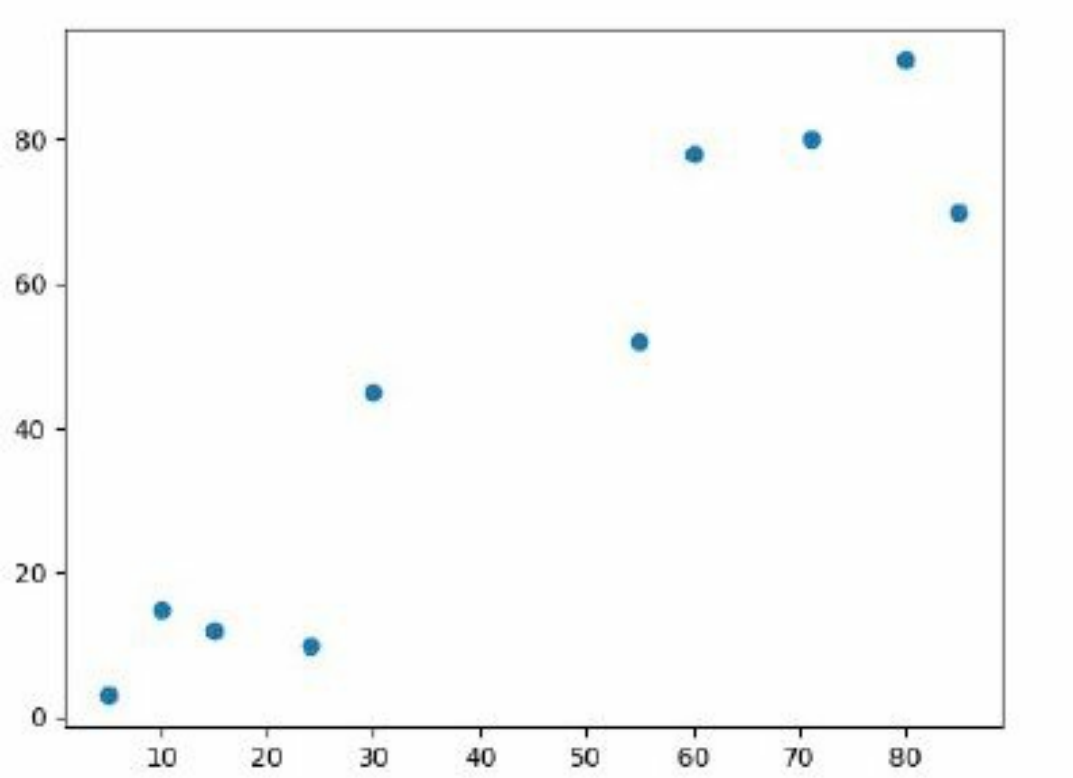
We should now prepare the data that is to be used. We will be creating a numpy array with a total of 10 rows and 2 columns. So, why have we chosen to work with a numpy array? It is because the Scikit-Learn library can work with the numpy array data inputs without the need for preprocessing.

Visualizing the Data

Now that we have the data, we can create a plot and see how the data points are distributed. We will then be able to tell whether there are any clusters at the moment:

```
plt.scatter(X[:,0],X[:,1], label='True Position')
plt.show()
```

The code gives the following plot:



If we use our eyes, we will probably make two clusters from the above data, one at the bottom with five points and another one at the top with five points. We now need to investigate whether this is what the K-Means clustering algorithm will do.

Creating Clusters

We have seen that we can form two clusters from the data points, hence the value of K is now 2. These two clusters can be created by running the following code:

```
kmeans_clusters = KMeans(n_clusters=2)
kmeans_clusters.fit(X)
```

We have created an object named *kmeans_clusters,* and 2 have been used as the value for the parameter *n_clusters*. We have then called the *fit()* method on this object and passed the data we have in our numpy array as the parameter to the method. We can now have a look at the centroid values that the algorithm has created for the final clusters: print (kmeans_clusters.cluster_centers_) This returns the following: The first row above gives us the coordinates for the first centroid, which is, (16.8, 17). The second row gives us the coordinates of the second centroid, which is, (70.2, 74.2). If you followed the manual process of calculating the values of these, they should be the same. This will be an indication that the K-Means algorithm worked well.
The following script will help us see the data point labels:

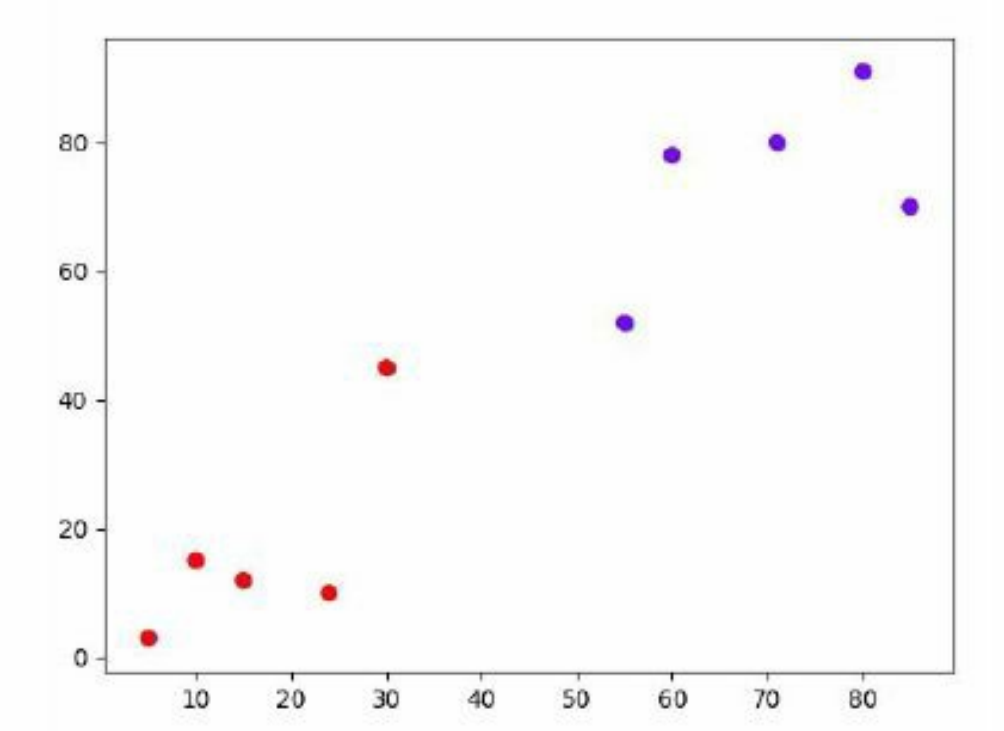
print(kmeans_clusters.labels_)

This returns the following:
The above output shows a one-dimensional array of 10 elements that correspond to the clusters that are assigned to the 10 data points. You clearly see that we first have a sequence of zeroes, which shows that the first 5 points have been clustered together while the last five points have been clustered together. Note that the 0 and 1 have no mathematical significance, but they have simply been used to represent the cluster IDs. If we had three clusters, then the last one would have been represented using 2's.

We can now plot the data points and see how they have been clustered. We need to plot the data points alongside their assigned labels to be able to distinguish the clusters. Just execute the script given below:

```
plt.scatter(X[:,0],X[:,1], c=kmeans_clusters.labels_, cmap='rainbow')
plt.show()
```

The script returns the following plot:

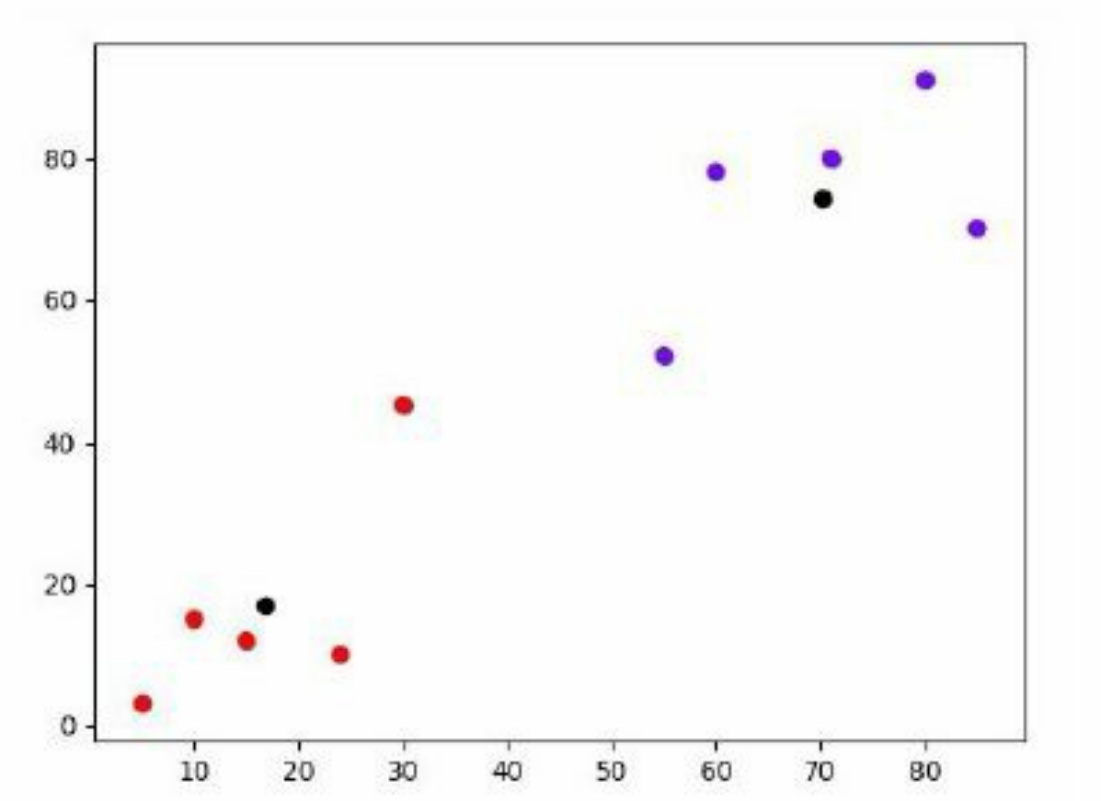


We have simply plotted the first column of the array named X against the second column. At the same time, we have passed *kmeans_labels_* as the value for parameter c, which corresponds to the labels. Note the use of the parameter *cmap='rainbow'*. This parameter helps us to choose the color type for the different data points.

As you expected, the first five points have been clustered together at the bottom left and assigned a similar color. The remaining five points have been clustered together at the top right and assigned one unique color. We can choose to plot the points together with the centroid coordinates for every cluster to see how the positioning of the centroid affects clustering. Let us use three clusters to see how they affect the centroids. The following script will help you to create the plot:

```
plt.scatter(X[:,0], X[:,1], c=kmeans_clusters.labels_, cmap='rainbow')
plt.scatter(kmeans_clusters.cluster_centers_[:,0] ,kmeans_clusters.cluster_centers_[:,1], color='black')
plt.show()
```

The script returns the following plot:



Chapter 11: Classification

Is it spam or not spam? This is one of the popular applications and examples of Classification. Similar to regression, Classification is also categorized under Supervised Learning. The model learns from labeled data. Then the system uses that learning to a new dataset.

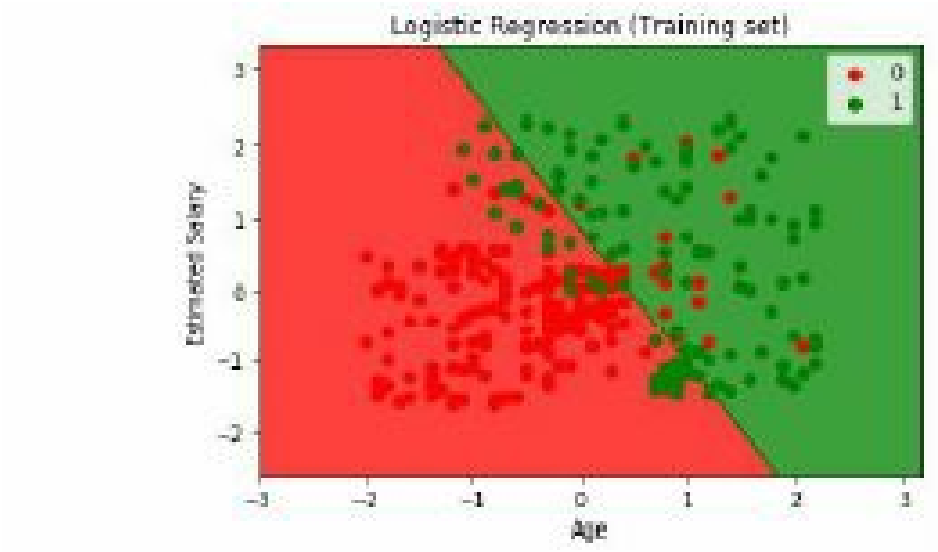
For instance, there's a dataset with various email messages, and each dataset is classified as spam or not spam. The model may then identify features among email messages that are highlighted as spam.
When conducting a prediction, the model may attempt to identify those features in new email messages.

There are different ways of accomplishing a successful classification. Let's look a few of them:

Logistics Regression

In most classification problems, the objective is to confirm whether it's 0 or 1 by using two independent variables. For instance, provided that the Age and Estimated Salary determine a result such as when the person bought, how can we successfully build a model that represents their relationships, and use that for prediction.

This might sound confusing, and that is why you need to get an example.



In the above examples, there are two variables Age and Estimated Salary. Every data point is

then categorized either as 0 or 1. There's a line that separates the two. This method is based on probability.

As with Regression, wherein there's the black box, and the behind the scenes of Logistics regression can be difficult. The good thing is that its execution is straightforward when you apply scikit-learn and Python.

It is a usual step to first learn from the Training Set and then use that learning to test set. At the end of the day, this is the purpose of Supervised Learning. First, you will find there's training and supervision. Next, the lesson will be used in new situations.
As you discover in the visualization for the test set, the majority of the green dots fall under the green region. Therefore, the model can be best for predicting whether a person with specific age and estimated salary can buy or not.

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

First, we convert the data into the same range or scale to avoid skewing or over-dependency on a given variable. In the dataset, the approximated salary is represented in thousands, while age is represented on a smaller scale. We need to set them in the same range so that we can expect a reasonable model.

Besides Logistic Regression, there are other methods of grouping tasks. Let's discuss them:

K-Nearest Neighbors

This algorithm is popular for creating advanced classifiers. It is an easy algorithm, but it has done better than other classifiers. That is the reason it's popular in genetics, forecasting, and data compression.

The KNN belongs to a supervised learning algorithm. In other words, we are presented with a dataset made up of training observations (x, y), and the main goal is to identify the association between x and y. So, we need to define a function x-y such that when we are provided with an input value for x, we can predict the corresponding value for y.

The idea behind the KNN algorithm is quite simple. It computes the distance of the new data point to all the other training data points. The distance can be of different types, including Euclidean, Manhattan, etc. The K-nearest data points are selected, where K can be an integer. Lastly, the data point is assigned to the class, which most of the K data points belong to.

Remember that Logistic Regression appears to feature a linear boundary between 0s and 1s. Therefore, it lacks a few of the data points that require to be on the other end.
Luckily, there are non-linear models that feature a lot of data points in a more accurate style. One of them is the application of K-Nearest Neighbors. It operates by letting a new data point and then counting the number of neighbors that belong to either category. In case a lot of neighbors belong to category A than category B, then the new point should be part of category A.

Therefore, the classification of a given point depends on the majority of its nearest neighbors. This can be achieved the following line of code:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Also, rather than starting from square one, we will import the prebuilt code that simplifies our task. The behind the scenes can be learned and studied. But for many reasons, the prebuilt ones are better to make reasonably helpful models.

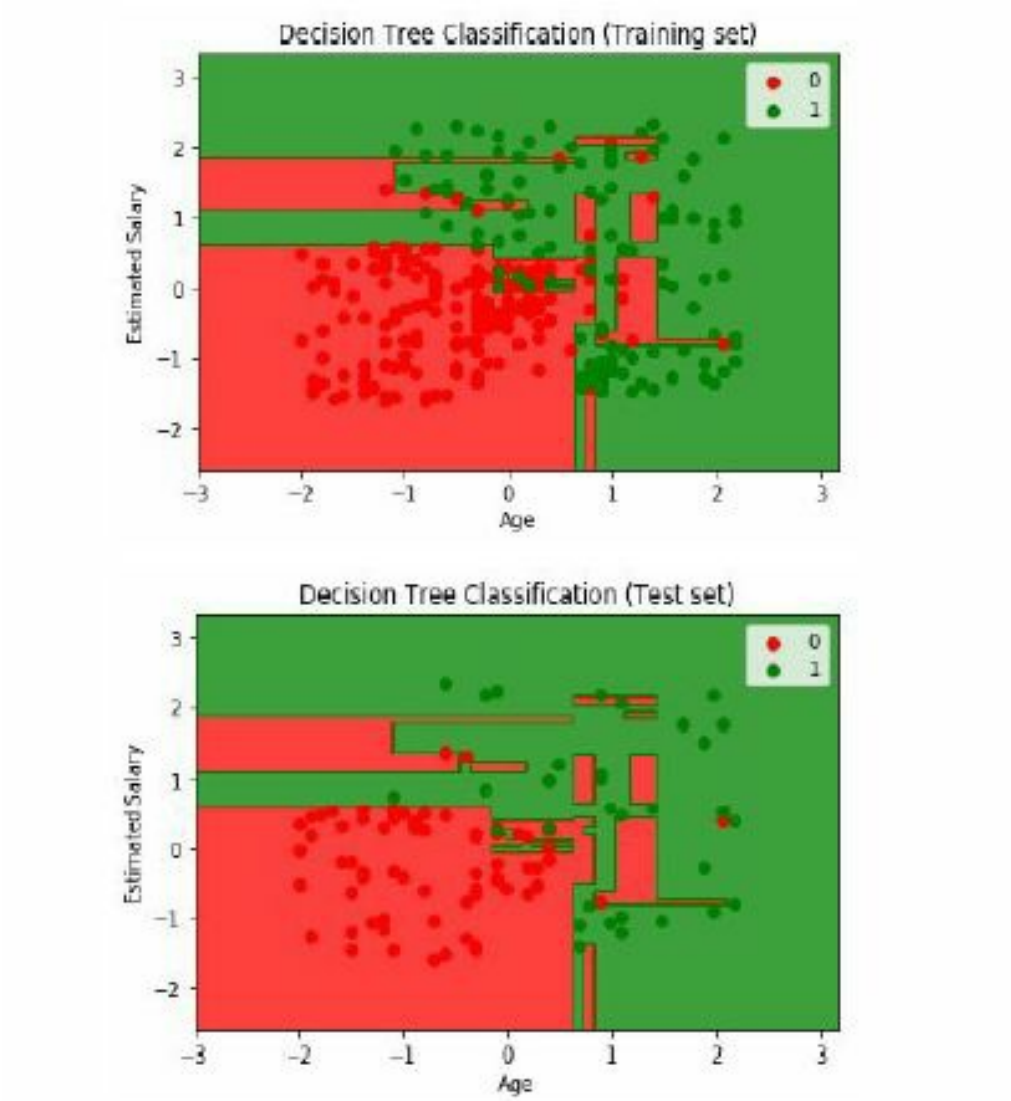
Don’t forget that the boundary is non-linear because of the various approach by the K-NN algorithm. Also, remember that the misses still exist. For you to capture all the misses, you need to use a larger dataset or a different method.

The Decision Tree Classification

Many data scientists also apply decision trees in classification. Building a decision tree involves dividing a dataset into smaller subsets while separating them out. Take a look at the example below:

Are you able to see that the branches and leaves originate from the dataset division? The same can apply in classification.

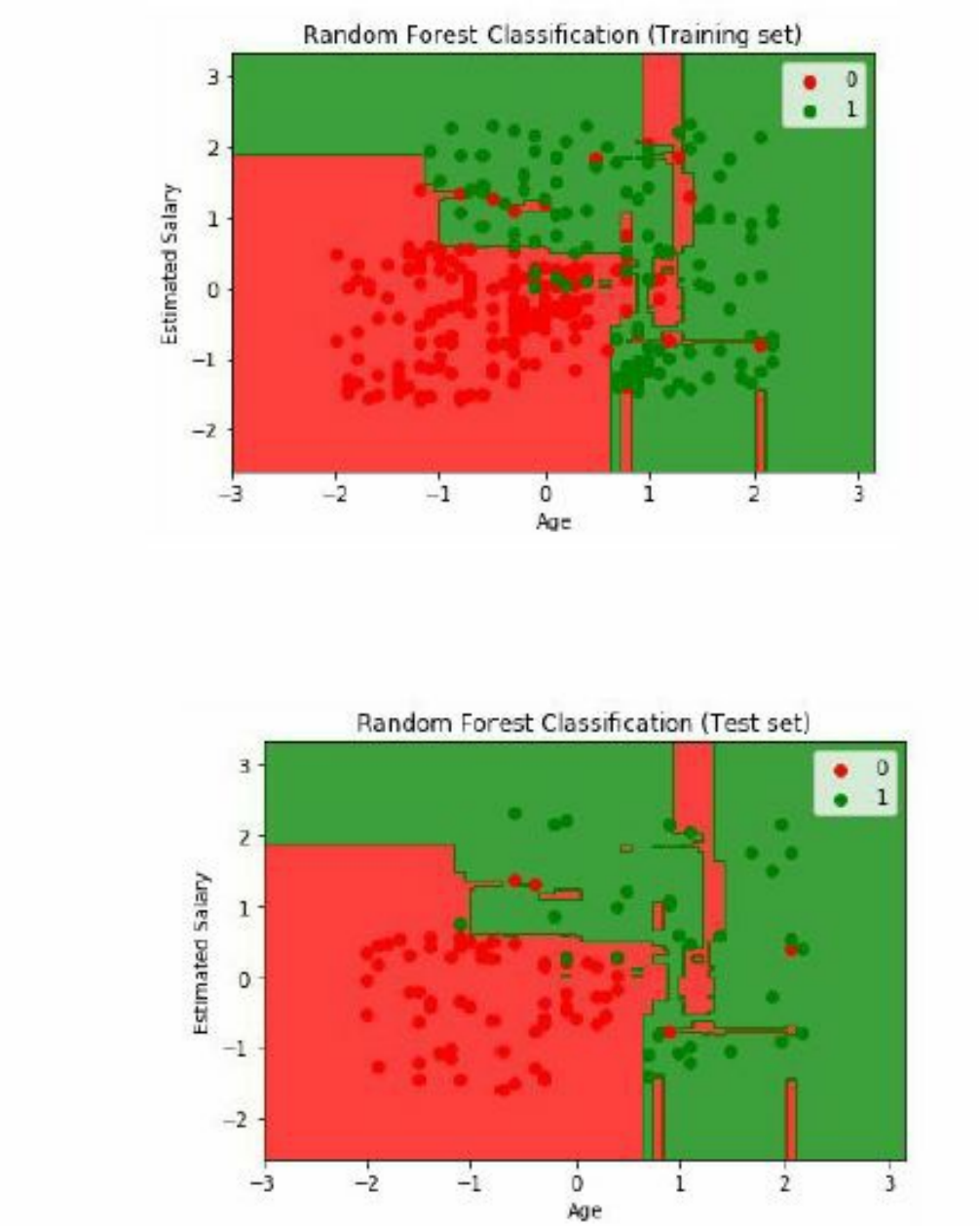
When you run the entire code, you will see the following:



See the big difference when it is compared to Logistic regression and K-Nearest Neighbors (K-NN). In these two examples, there are only two boundaries. In the above decision tree classification, you can identify two points outside the main red region that are within the “mini red regions.” In the following case, the model was able to retain the data points that would be impossible otherwise.

Random Forest Classification

Remember from the previous section on regression, random forest was described as a collection of decision trees. This applies to classification wherein a lot of decision trees are applied, and the results are averaged.



Take a look at the similarities between random forest and decision tree. At the end of the day, they pick a similar approach of dividing down a dataset into smaller subsets. The difference is that the Random Forest applies randomness and averages the decision trees to develop an accurate model.

Clustering

The preceding chapters explored supervised learning. We have mastered learning from “labeled” data. Already, there were correct responses and our role was to learn how to get those answers and use them in learning new data.

In the following chapter, it will be different. The reason is that it will be beginning with unsupervised learning, wherein there were no exact labels provided.

This means there's only input data, but there's no output data. Additionally, no supervision happens when learning from data.

In fact, unsupervised learning is considered to engulf the essence of artificial intelligence. That's because there is no much human supervision or intervention. As a result, the algorithms are left to discover things from data. This is the case in Clustering wherein the eventual goal is to reveal organic aggregates or "clusters" in data.

Objectives and Function of Clustering

This is a form of Unsupervised Learning where there are no labels, or in many cases, there are no truly correct answers. That's because there were no correct answers in the first place. We just have a dataset, and our goal is to see the groupings that have organically formed. We're not trying to predict an outcome here. The goal is to look for structures in the data. In other words, we're "dividing" the dataset into groups wherein members have some similarities or proximities. For example, each e-commerce customer might belong to a particular group (e.g., given their income and spending level). If we have gathered enough data points, it's likely there are aggregates.

At first, the data points will seem scattered (no pattern at all). But once we apply a clustering algorithm. The data will somehow make sense because we'll be able to easily visualize the groups or clusters. Aside from discovering the natural groupings, Clustering algorithms may also reveal outliers for Anomaly Detection (we'll also discuss this later). Clustering is being applied regularly in the fields of marketing, biology, earthquake studies, manufacturing, sensor outputs, product categorization, and other scientific and business areas. However, there are no rules set in stone when it comes to determining the number of clusters and which data point should belong to a certain cluster. It's up to our objective (or if the results are useful enough). This is also where our expertise in a particular domain comes in. As with other data analysis and machine learning algorithms and tools, it's still about our domain knowledge. This way, we can look at and analyze the data in the proper context. Even with the most advanced tools and techniques, the context and objective are still crucial in making sense of data.

K-Means Clustering

One way to make sense of data through Clustering is by K-Means. It's one of the most popular Clustering algorithms because of its simplicity. It works by partitioning objects into k clusters (number of clusters we specified) based on feature similarity. Notice that the number of clusters is arbitrary. We can set it into any number we like. However, it's good to make the number of clusters just enough to make our work meaningful and useful. Let's discuss an example to

illustrate this. Here we have data about Mall Customers ('Mall_Customers.csv') where info about their Gender, Age, Annual Income, and Spending Score are indicated. The higher the Spending Score (out of 100), the more they spend at the Mall.
To start, we import the necessary libraries:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

Then we import the data and take a peek:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

In this example, we're more interested in grouping the Customers according to their Annual Income and Spending Score.

X = dataset.iloc[:, [3, 4]].values

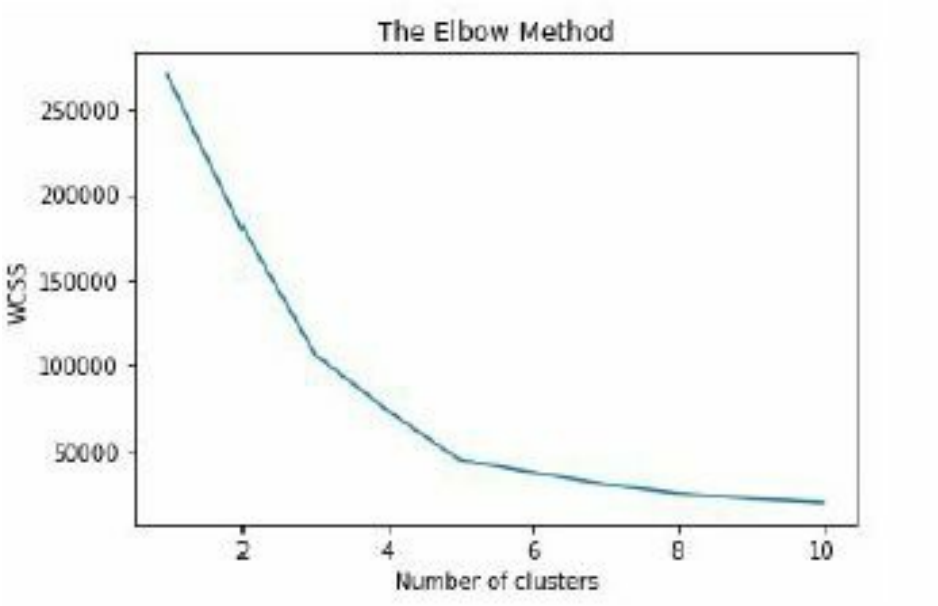
Our goal here is to reveal the clusters and help the marketing department formulate their strategies. For instance, we might subdivide the Customers in 5 distinct groups:

1. Medium Annual Income, Medium Spending Score
2. High Annual Income, Low Spending Score
3. Low Annual Income, Low Spending Score

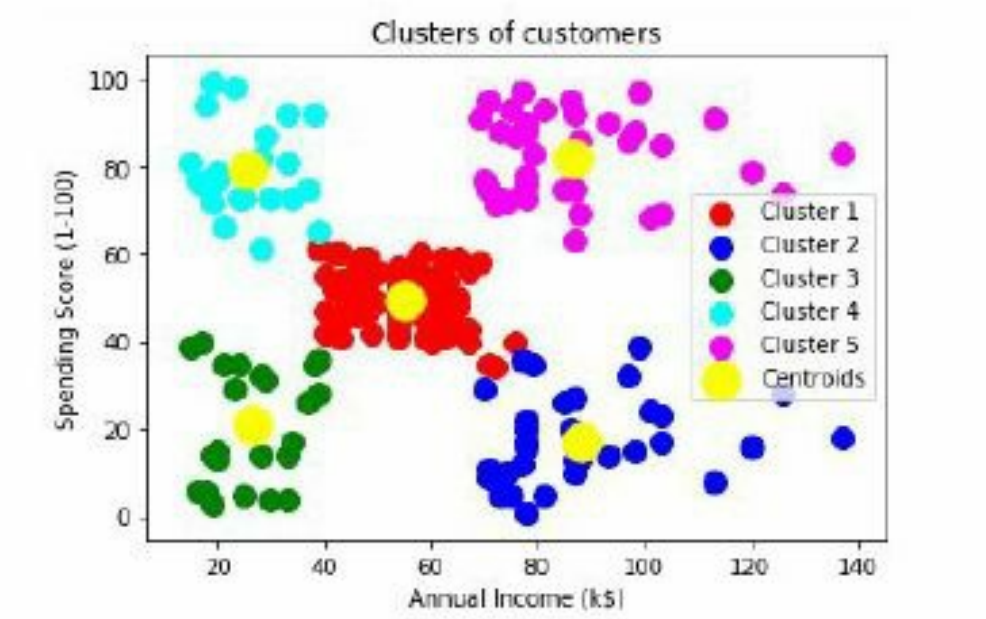
4. Low Annual Income, High Spending Score
5. High Annual Income, High Spending Score

It's worthwhile to pay attention to the #2 Group (High Annual Income, Low Spending Score). If there's a sizable number of customers that fall under this group, it could mean a huge opportunity for the mall. These customers have high Annual Income, and yet they're spending or using most of their money elsewhere (not in the Mall). If we could know that they're in sufficient numbers, the marketing department could formulate specific strategies to entice Cluster #2 to buy more from the Mall.

Although the number of clusters is often arbitrary, there are ways to find that optimal number. One such way is through the Elbow Method and WCSS (within-cluster sums of squares).



Notice that the "elbow" points at 5 (number of clusters). Coincidentally, this number was also the "desired" number of groups that will subdivide the dataset according to their Annual Income and Spending Score. After determining the optimal number of clusters, we can then proceed with applying K-Means to the dataset and then performing data visualization:



There we have it. We have 5 clusters and Cluster #2 (blue points, High Annual Income and Low Spending Score) is significant enough. It might be worthwhile for the marketing department to focus on that group. Also, notice the Centroids (the yellow points). This is a part of how K-Means clustering works. It's an iterative approach where random points are placed initially until they converge to a minimum (e.g., sum of distances is minimized). As mentioned earlier, it can all be arbitrary, and it may depend heavily on our judgment and possible application. We can set n_clusters into anything other
than 5. We only used the Elbow Method so we can have a sounder and consistent basis for the number of clusters. But it's still up to our judgment what should we use and if the results are good enough for our application.

Anomaly Detection

Aside from revealing the natural clusters, it's also a common case to see if there are obvious points that don't belong to those clusters. This is the heart of detecting anomalies or outliers in data. This is a crucial task because any large deviation from the normal can cause a catastrophe. Is a credit card transaction fraudulent? Is a login activity suspicious (you might be logging in from a totally different location or device)? Are the temperature and pressure levels in a tank being maintained consistently (any outlier might cause explosions and operational halt)? Is a certain data point caused by wrong entry or measurement (e.g., perhaps inches were used instead of centimeters)? With straightforward data visualization, we can immediately see the outliers. We can then evaluate if these outliers present a major threat. We can also see and assess those outliers by referring to the mean and standard deviation. If a data point deviates by a standard deviation from the mean, it could be an anomaly. This is also where our domain expertise comes in. If there's an anomaly, how serious are the consequences? For instance, there might be thousands of purchase transactions happening in an online store every day. If we're too tight with

our anomaly detection, many of those transactions will be rejected (which results in loss of sales and profits). On the other hand, if we're allowing much freedom in our anomaly detection, our system would approve more transactions. However, this might lead to complaints later and possibly loss of customers in the long term. Notice here that it's not all about algorithms, especially when we're dealing with business cases. Each field might require a different sensitivity level. There's always a tradeoff, and either of the options could be costly. It's a matter of testing and knowing if our system of detecting anomalies is sufficient for our application.

Chapter 12: Association Rule Learning

This is a continuation of Unsupervised Learning. In the previous chapters, we've discovered natural patterns and aggregates. There was not much supervision and guidance on how the "correct answers" should look like. We've allowed the algorithms to discover and study the data. As a result, we're able to gain insights from the data that we can use. In this chapter, we'll focus on Association Rule Learning. The goal here is to discover how items are "related" or associated with one another. This can be very useful in determining which products should be placed together in grocery stores. For instance, many customers might always be buying bread and milk together. We can then rearrange some shelves and products so the bread and milk will be near to each other. This can also be a good way to recommend related products to customers. For example, many customers might be buying diapers online and then purchasing books about parenting later. These two products have strong associations because they mark the customer's life transition (having a baby). Also, if we notice a demand surge in diapers, we might also get ready with parenting books. This is a good way to somehow forecast and prepare for future demands by buying supplies in advance. In grocery shopping or any business involved in retail and wholesale transactions, Association Rule Learning, can be very useful in optimization (encouraging customers to buy more products) and matching supply with demand (e.g., sales improvement in one product also signals the same thing to another related product).

Explanation

So how do we determine the "level of relatedness" of items to one another and create useful groups out of it.? One straightforward approach is by counting the transactions that involve a particular set.

Transactions	Purchases
1	Egg, ham, hotdog
2	Egg, ham, milk
3	Egg, apple, onion
4	Beer, milk, juice

Our target set is {Egg, ham}. Notice that this combination of purchases occurred in 2 transactions (Transactions 1 and 2). In other words, this combination happened 50% of the time. It's a simple example, but if we're studying 10,000 transactions and 50% is still the case, of course, there's a strong association between egg and ham. We might then realize that it's worthwhile to put eggs and hams together (or offer them in a bundle) to make our customers' lives easier (while we also make more sales). The higher the percentage of our target set in the

total transactions, the better. Or, if the percentage still falls under our arbitrary threshold (e.g., 30%, 20%), we could still pay attention to a particular set and make adjustments to our products and offers. Aside from calculating the actual percentage, another way to know how "popular" an itemset is by working on probabilities. For example, how likely is product X to appear with product Y? If there's a high probability, we can somehow say that the two products are closely related. Those are ways of estimating the "relatedness" or level of association between two products. One or a combination of approaches might be already enough for certain applications. Perhaps working on probabilities yields better results. Or, prioritizing a very popular itemset (high percentage of occurrence) results to more transactions. In the end, it might be about testing different approaches (and combinations of products) and then seeing which one yields the optimal results. It might be even the case that a combination of two products with very low relatedness allows for more purchases to happen.

Apriori

Whichever is the case, let's explore how it all applies to the real world. Let's call the problem "Market Basket Optimization." Our goal here is to generate a list of sets (product sets) and their corresponding level of relatedness or support to one another. Here's a peek of the dataset to give you a better idea:

shrimp,almonds,avocado,vegetables mix,green grapes,whole wheat flour,yams,cottage cheese,energy drink,tomato juice,low fat yogurt,green tea,honey,salad,mineral water,salmon,antioxidant juice,frozen smoothie,spinach,olive oil
burgers,meatballs,eggs
chutney
turkey,avocado
mineral water,milk,energy bar,whole wheat rice,green tea
low-fat yogurt
whole-wheat pasta,french fries
soup,light cream,shallot
frozen vegetables,spaghetti,green tea
french fries

Those are listed according to the transactions where they appear. For example, in the first transaction, the customer bought different things (from shrimp to olive oil). In the second transaction, the customer bought burgers, meatballs, and eggs.

As before, let's import the necessary library/libraries so that we can work on the data:

import Pandas as pd

dataset = pd.read_csv('Market_Basket_Optimisation.csv', header = None)

Next is we add the items in a list so that we can work on them much easier. We can accomplish this by initializing an empty list and then running a for loop (still remember how to do all these?):

```
transactions = []
for i in range(0, 7501):
transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])
```

After we've done that, we should then generate a list of "related products" with their corresponding level of support or relatedness. One way to accomplish this is by the implementation of the Apriori algorithm (for association rule learning). Thankfully, we don't have to write anything from scratch.

We can use Apyori, which is a simple implementation of the Apriori algorithm.

It's prebuilt for us and almost ready for our own usage. It's similar to how we use scikit-learn, Pandas, and numpy. Instead of starting from scratch, we already have blocks of code we can simply implement. Take note that coding everything from scratch is time-consuming and technically challenging. To implement Apyori, we can import it similarly as how we import other libraries:

from apyori import apriori

Next is we set up the rules (the levels of minimum relatedness) so we can somehow generate a useful list of related items. That's because almost any two items might have some level of relatedness. The objective here is to include only the list that could be useful for us.

```
rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2,
min_lift = 3, min_length = 2)
```

Well that's the implementation of Apriori using Apyori. The next step is to generate and view the results. We can accomplish this using the following block of code:

```
results = list(rules)
results_list = []
for i in range(0, len(results)):
results_list.append('RULE:\t' + str(results[i][0]) + '\nSUPPORT:\t' + str(results[i][1]))
print (results_list)
```

When you run all the code in Jupyter Notebook. It will be messy and almost incomprehensible. But if you run it in Spyder (another useful data science package included in Anaconda installation), the result will look a bit neater: Notice that there are different itemsets with their corresponding “Support.” The higher the Support, we can somehow say that the higher the relatedness. For instance, light cream and chicken often go together because people might be using the two to cook something. Another example is in the itemset with an index of 5 (tomato sauce and ground beef). These two items might always go together in the grocery bag because they’re also used to prepare a meal or a recipe.

This is only an introduction to Association Rule Learning. The goal here was to explore the potential applications of it to real-world scenarios such as market basket optimization. There are other, more sophisticated ways to do this. But in general, it’s about determining the level of relatedness among the items and then evaluating that if it’s useful or good enough.

Chapter 13: Reinforcement Learning

Notice that in the previous chapters, the focus is on working on past information and then deriving insights from it. In other words, we're much focused on the past than on the present and future. But for data science and machine learning to become truly useful, the algorithms and systems should work on real-time situations. For instance, we require systems that learn real-time and adjust accordingly to maximize the rewards.

What is Reinforcement Learning?

This is where Reinforcement Learning (RL) comes in. In a nutshell, RL is about reinforcing the correct or desired behaviors as time passes. A reward for every correct behavior and a punishment otherwise. Recently RL was implemented to beat world champions at the game of Go and successfully play various Atari video games (although Reinforcement Learning there was more sophisticated and incorporated deep learning). As the system learns from reinforcement, it was able to achieve a goal or maximize the reward.

One simple example is in the optimization of click-through rates (CTR) of online ads. Perhaps you have 10 ads that essentially say the same thing (maybe the words and designs are slightly different from one another). At first, you want to know which ad performs best and yields the highest CTR. After all, more clicks could mean more prospects and customers for your business. But if you want to maximize the CTR, why not perform the adjustments as the ads are being run? In other words, don't wait for your entire ad budget to run out before knowing which one performed best. Instead, find out which ads are performing best while they're being run. Make adjustments early on, so later, only the highest-performing ads will be shown to the prospects.

It's very similar to a famous problem in probability theory about the multiarmed bandit problem. Let's say you have a limited resource (e.g., advertising budget) and some choices (10 ad variants). How will you allocate your resource among those choices so you can maximize your gain (e.g., optimal CTR)?

First, you have to "explore" and try the ads one by one. Of course, if you're seeing that Ad 1 performs unusually well, you'll "exploit" it and run it for the rest of the campaign. You don't need to waste your money on underperforming ads. Stick to the winner and continuously exploit its performance. There's one catch, though. Early on, Ad 1 might be performing well, so we're tempted to use it again and again. But what if Ad 2 catches up and if we let things unfold Ad 2 will produce higher gains?

We'll never know because the performance of Ad 1 was already exploited. There will always be tradeoffs in many data analysis and machine learning projects. That's why it's always recommended to set performance targets beforehand instead of wondering about the what-ifs later. Even in the most sophisticated techniques and algorithms, tradeoffs and constraints are always
there.

Comparison with Supervised & Unsupervised Learning

Notice that the definition of Reinforcement Learning doesn't exactly fit under either Supervised or Unsupervised Learning. Remember that Supervised Learning is about learning through supervision and training. On the other hand, Unsupervised Learning is actually revealing or discovering insights from unstructured data (no supervision, no labels).

One key difference compared to RL is in maximizing the set reward, learning from user interaction, and the ability to update itself in real-time. Remember that RL is first about exploring and exploiting. In contrast, both Supervised and Unsupervised Learning can be more about passively learning from historical data (not real-time). There's a fine boundary among the 3 because all of them are still concerned about optimization in one way or another. Whichever is the case, all 3 have useful applications in both scientific and business settings.

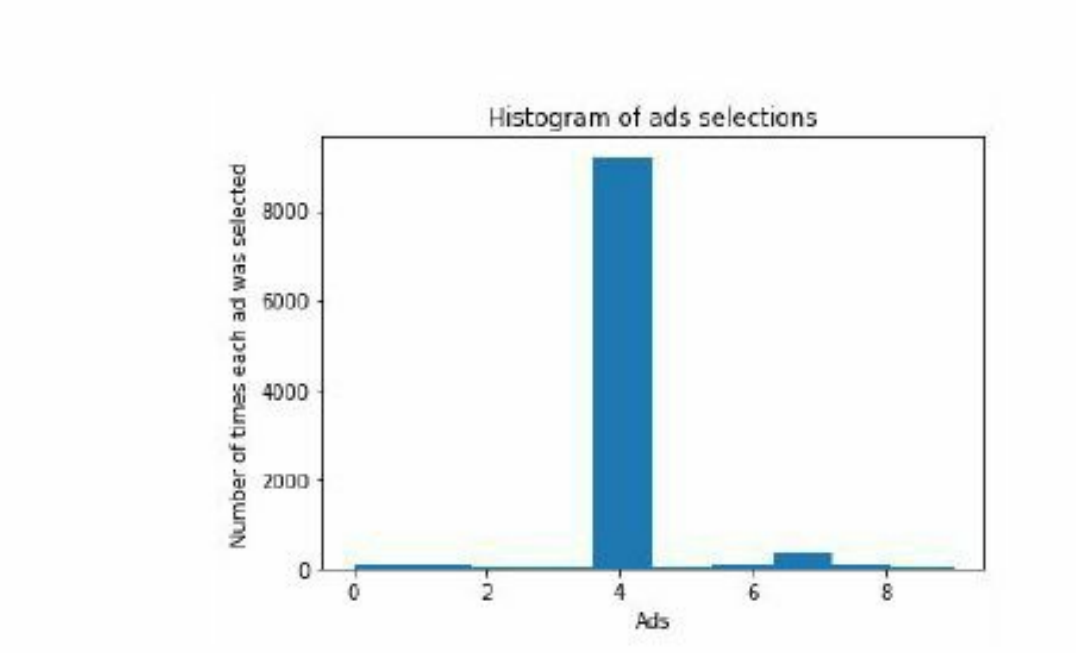
Applying Reinforcement Learning

RL is particularly useful in many business scenarios, such as optimizing click through rates. How can we maximize the number of clicks for a headline? Take note that news stories often have limited lifespans in terms of their relevance and popularity. Given that limited resource (time), how can we immediately show the best performing headline?

This is also the case in maximizing the CTR of online ads. We have a limited ad budget, and we want to get the most out of it. Let's explore an example (using the data from Ads_CTR_Optimisation.csv) to better illustrate the idea: As usual, we first import the necessary libraries so that we can work on our data (and also for data visualization).

```
import matplotlib.pyplot as plt
import Pandas as pd
%matplotlib inline #so plots can show in our Jupyter Notebook
We then import the dataset and take a peek
dataset = pd.read_csv('Ads_CTR_Optimisation.csv')
dataset.head(10)
```

In each round, the ads are displayed, and it's indicated which one/ones were clicked (0 if not clicked, 1 if clicked). As discussed earlier, the goal is to explore first, pick the winner, and then exploit it. One popular way to achieve this is by Thompson Sampling. Simply, it addresses the exploration-exploitation dilemma (trying to achieve a balance) by sampling or trying promising actions while ignoring or discarding actions that are likely to underperform. The algorithm works on probabilities and this can be expressed in code.
When we run the code and visualize:



Notice that the implementation of Thompson sampling can be very complex.
It's an interesting algorithm which is widely popular in online ad optimization, news article recommendation, product assortment, and other business applications.
There are other interesting algorithms and heuristics such as Upper Confidence Bound. The goal is to earn while learning. Instead of later analysis, our algorithm can perform and adjust in real-time. We're hoping to maximize the reward by trying to balance the tradeoff between exploration and exploitation (maximize immediate performance or "learn more" to improve future performance). It's an interesting topic itself, and if you want to dig deeper, you can read more online resources.