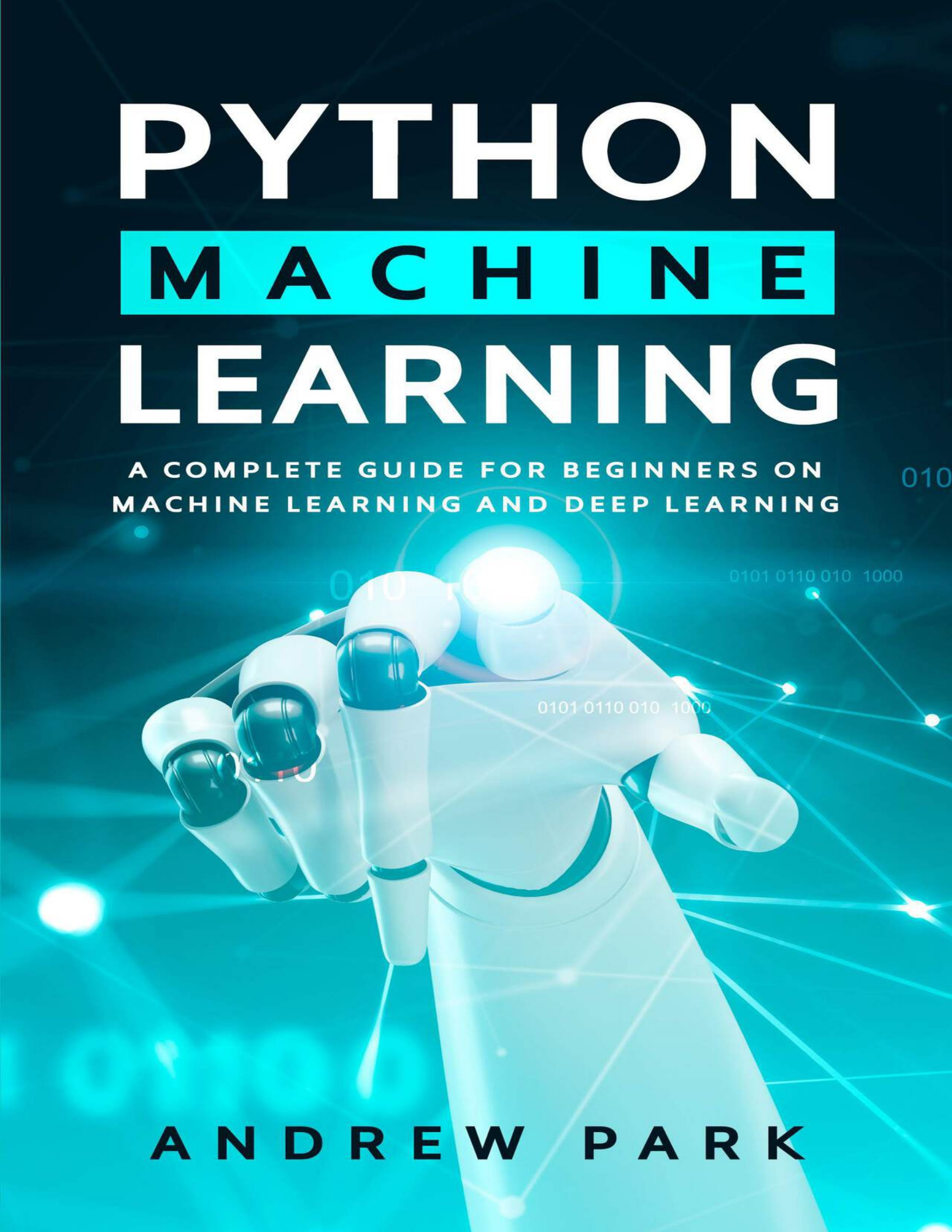
PYTHON MACHINE LEARNING

A COMPLETE GUIDE FOR BEGINNERS ON MACHINE LEARNING AND DEEP LEARNING



ANDREW PARK

Python Machine Learning

A Complete Guide for Beginners on Machine Learning and Deep Learning with Python

Andrew Park

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

Table of Contents

Introduction

For all that we know about Machine Learning, the truth is that we are nowhere close to realizing the true potential of these studies. Machine Learning is currently one of the hottest topics in computer science. If you are a data analyst, this is a field you should focus all your energy on because the prospects are incredible. You are looking at a future where interaction with machines will form the base of our being.

In this installation, our purpose was to address Python Machine Learning from the perspective of an expert. The assumption is that you have gone through the earlier books in the series that introduced you to Machine Learning, Python, libraries, and other important features that form the foundation of your knowledge in Machine Learning. With this in mind, we barely touched on the introductory concepts, unless necessary.

Even at an expert level, it is always important to remind yourself of the important issues that we must look at in Machine Learning. Algorithms are the backbone of almost everything that you will do in Machine Learning. Because of this reason, we introduced a brief section where you can remind yourself of the important algorithms and other elements that help you progress your knowledge of Machine Learning.

Machine Learning is as much about programming as it is about probability and statistics. There are many statistical approaches that we will use in Machine Learning to help us arrive at optimal solutions from time to time. It is therefore important that you remind yourself about some of the necessary probability theories and how they affect outcomes in each scenario.

In our studies of Machine Learning from the beginner books through an intermediary level to this point, one concept that stands out is that Machine Learning involves uncertainty. This is one of the differences between Machine Learning and programming. In programming, you write code that must be executed as it is written. The code derives a predetermined output based on the instructions given. However, in Machine Learning, this is not a luxury we enjoy.

Once you build the model, you train and test it and eventually deploy the model. Since these models are built to interact with humans, you can expect variances in the type of interaction that you experience at every level. Some input parameters might be correct, while others might not. When you build

your model, you must consider these factors, or your model will cease to perform as expected.

The math element of Machine Learning is another area of study that we have to look at. We didn't touch on this so much in the earlier books in the series because it is an advanced level study. Many mathematical computations are involved in Machine Learning for the models to deliver the output we need. To support this cause, we must learn how to perform specific operations on data based on unique instructions.

As you work with different sets of data, there is always the possibility that you will come across massive datasets. This is normal because as our Machine Learning models interact with different users, they keep learning and build their knowledge. The challenge of using massive datasets is that you must learn how to break down the data into small units that your system can handle and process without any challenges. In this case, you are trying to avoid overworking your learning model.

Most basic computers will crash when they have to handle massive data. However, this should not be a problem when you learn how to fragment your datasets and perform computational operations on them.

At the beginning of this book, we mentioned that we will introduce hands-on approaches to using Machine Learning in daily applications. In light of this assertion, we looked at some practical methods of using Machine Learning, such as building a spam filter and analyzing a movie database.

We have taken a careful step-by-step approach to ensure that you can learn along the way, and more importantly, tried to explain each process to help you understand the operations you perform and why.

Eventually, when you build a Machine Learning model, the aim is to integrate it into some of the applications that people use daily. With this in mind, you must learn how to build a simple solution that addresses this challenge. We used simple explanations to help you understand this, and hopefully, as you keep working on different Machine Learning models, you can learn by building more complex models as your needs permit.

There are many concepts in Machine Learning that you will learn or come across over time. You must reckon the fact that this is a never-ending learning process as long as your model interacts with the data. Over time, you will encounter greater datasets than those you are used to working with. In such a scenario, learning how to handle them will help you achieve your

results faster, and without struggling.

What Is Machine Learning?

We live in a world where technology has become an inalienable part of our daily lives. In fact, with all the rapid changes in technology these days, machines enabled with artificial intelligence are now responsible for different tasks like prediction, recognition, diagnosis, and so on.

Data is added or fed to the machines and these machines “learn” from these data. These data are referred to as training data because they are used to train the machines.

Once the machines have the data, they start to analyze any patterns present within the data and then perform actions based on these patterns. Machines use various learning mechanisms for analyzing the data according to the actions that they need to perform. These mechanisms can be broadly classified into two categories- supervised learning and unsupervised learning.

You might wonder why there aren’t any machines designed solely to perform those tasks that they are needed to carry out. There are different reasons why Machine Learning is important. As already mentioned, all research conducted about Machine Learning comes in handy since it helps us understand a couple of aspects of human learning. Also, Machine Learning is quintessential because it helps increase the accuracy, effectiveness, and efficiency of machines.

Here is a real-life example that will help you understand this concept better.

Let us assume that there are two random users A and B who love listening to music and we have access to their history of songs. If you were a music company, then you can use Machine Learning to understand the kind of songs each of these users prefers and thereby you can come up with different ways in which you can sell your products to them.

For instance, you have access to noting down the different attributes of songs like their tempo, frequency, or the gender of the voice, and then use all these attributes and plot a graph. Once you plot a graph, over time, it will become evident that A tends to prefer to listen to songs that have a fast tempo and are sung by male artists, whereas B likes to listen to slow songs sung by female artists, or any other similar insight. Once you obtain these data, you can transfer them to your marketing and advertising teams to make better product decisions.

At present, we have free access to all the historical data that have been collected since the advent of technology. Not only we have access to these data, but we can now store and process such large quantities of them. Technology has certainly evolved, and it has come a long way when you look at the way we can now handle such operations. The technology is so refined these days that it provides access to more data to mine from.

Here are a couple of other reasons why Machine Learning is important.

Even with all the progress that engineers keep making, there will always be some tasks that are incapable of being defined explicitly.

Some tasks must be explained to the machines with the help of examples. The idea is to train the machine with the input of data and then teach it to process it to produce an output. In this manner, the machine will be aware of how it needs to deal with similar inputs of data in the future and process them accordingly to generate the appropriate outputs.

The fields of Machine Learning and data mining are intertwined. Data mining refers to the process of going through tons of data to find any bowlers correlations or relationships that exist within. This is another benefit of Machine Learning in the sense that it helps the machines find any vital information.

There are numerous occasions where humans can’t design machines without having an accurate estimation of the conditions within which such machines will function.

The external conditions tend to have a major effect on the performance of the machine. In such situations, Machine Learning helps to get the machine acclimatized to its environment to ensure optimum performance. It also helps the machine to easily adapt to any changes in the environment without affecting its performance.

There is another problem if someone has to hardcode an extremely elaborate process into the machine, and then it is likely that the programmer will miss a couple of details. If there is any manual error, then it becomes quite tedious to encode all the details all over again. In such instances, it is better to allow the machine to learn the process instead.

The world of technology is in a constant flux of change and changes take place in the languages used as well. It isn’t practical to keep redesigning the systems all over again to accommodate all the possible changes. Instead,

Machine Learning helps the machine to automatically get acclimatized to all the changes.

Applications of Machine Learning

Machine Learning is drastically changing the way businesses are operated these days. It helps operate a large scale of data that's available and enables the users to draw helpful predictions based on the given information.

Certain manual tasks cannot be completed within a short time frame when large amounts of data are involved. Machine Learning is the answer to such problems. In the present times, we are overwrought with data and information and there is no physical way in which anyone can process all this information. Therefore, there is a dire need for an automated process and Machine Learning helps attain this objective.

When the processes of analysis and discovery are fully automated, it becomes simpler to attain useful information. This helps make all the future processes fully automated. The words Big Data, Business Analytics, and Data Science require Machine Learning. Predictive analytics and business intelligence are no longer restricted to just the elite businesses and are now accessible to small businesses and companies too. This allows small businesses to be a part of the process of collection and effective utilization of information. Let us look at a couple of technical applications of Machine Learning and see how these apply to problems in the real world.

Virtual Personal Assistants

Popular examples of virtual assistants available today are Alexa, Siri, and Google Now. As is obvious from the name, they help the user find the necessary information via voice commands. You simply need to activate it and then ask the question you want like, "What is my schedule for the day?" "What are the flights available between London and Germany?" or any other question that you want.

To answer your question, your personal assistant will look for information, recall the question you asked and then give you an answer. It can also be used to set reminders for certain tasks. Machine Learning is an important part of the process since it enables the system to gather and refine the information you need based on any of your previous involvements with it.

Density Estimation

Machine Learning allows the system to use the data that's available to it to suggest similar products. For instance, if you were to pick up a copy of Pride and Prejudice from a bookstore and then run it through a machine, then Machine Learning will help it determine the density of the words and come up with other books that are similar to Pride and Prejudice.

Latent Variables

When you are working with latent variables, the machine will use clustering to determine whether any of the variables present in it are related to one another or not. This comes in handy when you aren't certain of the reason that caused the change in variables and aren't aware of the relationship between the variables. When a large quantity of data is involved, it is easier to look for latent variables because it helps with a better understanding of the data thus obtained.

Reduction of Dimensionality

Usually, the data that is obtained tends to have some variables and dimensions. If there are more than three dimensions involved, then the human mind can't visualize that data. In such situations, Machine Learning helps to reduce these data into manageable proportions so that the user can easily understand the relationship between any variables.

Models of Machine Learning train the machines to learn from all the available data and offer different services like prediction or classification that in turn have multiple real-life applications like self-driving cars, the ability of smartphones to recognize the user's face or how Google Home or Alexa can recognize your accent and voice and how the accuracy of the machines improves if they have been learning for longer.

Advantages and Disadvantages of Machine Learning

Disadvantages

In Machine Learning, we always train the model and then validate that model on a small data set. We then use that model to predict the output for some unseen or new data. You will find it difficult to identify if there was a bias in the model that you have created. If you cannot identify the bias, your inferences will be incorrect.

Some social scientists will begin to rely only on Machine Learning. It is important to remember that improvements should be made to some

unsupervised Machine Learning tasks.

Some of the advantages

Human beings cannot process large volumes of data, let alone analyze that data. There is a lot of real-time data that is being produced, and if there is no automatic system to understand and analyze that data, we cannot reach any conclusion.

Machine Learning is getting better. With the advent of deep learning systems, the costs of data engineering and pre-processing of data are reducing.

Machine Learning – Concepts & Terms

Machine Learning is done by feeding the machine with relevant training data sets. Ordinary systems, that is, systems without any artificial intelligence, can always provide an output based on the input that is provided to the system. A system with artificial intelligence, however, can learn, predict, and improve the results it provides through training.

Let us look at a simple example of how children learn to identify objects, or in other words, how a child will associate a word with an object. Let us assume that there is a bowl of apples and oranges on the table. You, as an adult or parent, will introduce the round and red object as an apple, and the other object as an orange. In this example, the words apple and orange are labels, and the shapes and colors are attributes. You can also train a machine using a set of labels and attributes. The machine will learn to identify the object based on the attributes that are provided to it as input.

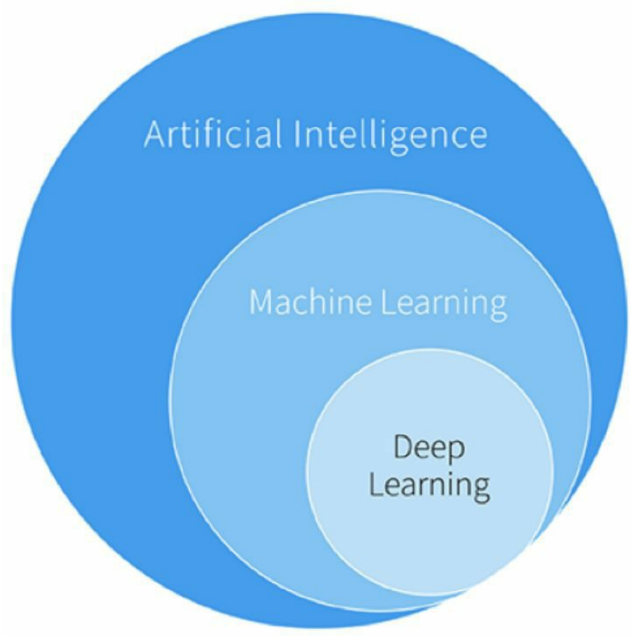
The models that are based on labeled training data sets are termed as supervised Machine Learning models. When children go to school, their teachers and professors give them some feedback about their progress. In the same way, a supervised Machine Learning model allows the engineer to provide some feedback to the machine.

Let us take an example of an input [red, round]. Here, both the child and the machine will understand that any object which is round and red is an apple. Let us now place a cricket ball in front of either the machine or the child. You can feed the machine with the response negative 1 or 0 depending on whether the prediction is wrong or right. You can always add more attributes if necessary. This is the only way that a machine will learn. It is also for this reason that if you use a large-high-quality data set and spend more time training the machine, the machine will give you better and more accurate results.

Before we proceed further, you must understand the difference between the concepts of Machine Learning, artificial intelligence, and deep learning. Most people use these concepts interchangeably, but it is important to know that they are not the same.

Machine Learning, Artificial Intelligence and Deep Learning:

The diagram below will give you an idea of how these terms relate.



An illustration to understand the relationship between Machine Learning, Artificial Intelligence, and Deep Learning.

Artificial intelligence is a technique that is used to make machines mimic any human behavior. The aim is to ensure that a machine can accurately and efficiently mimic any human behavior. Some examples of artificial intelligence machines include deep blue chess and IBM's Watson.

Machine Learning, as defined above, is the use of statistical and mathematical models to help machines learned mimic human behavior. This is done using past data.

Deep learning is a subset of Machine Learning, and it refers to the functions and algorithms that an engineer uses to help a machine to train itself. The machine can learn to take the correct option to derive an output. Neural networks and natural language processing are a part of the deep learning ecosystem.

Objectives of Machine Learning

The system of Machine Learning usually has one of the following objectives.

- Predict a category
- Predict a quantity
- Anomaly Detector Systems
- Clustering Systems

Predict a category

The model of Machine Learning helps analyze the input data and then predicts a category under which the output will fall. The prediction in such cases is usually a binary answer that's based on "yes" or "no." For instance, it helps with answers like, "will it rain today or not?" "Is this a fruit?" "Is this mail spam or not?" And so on. This is attained by referencing a group of data that will indicate whether a certain email falls under the category of spam or not based on specific keywords. This process is known as classification.

Predict a quantity

This system is usually used to predict a value like predicting the rainfall according to different attributes of the weather like the temperature, percentage of humidity, air pressure and so on. This sort of prediction is referred to as regression. The regression algorithm has various subdivisions like linear regression, multiple regression, etc.

Anomaly Detector Systems

The purpose of a model in anomaly detection is to detect any outliers in the given set of data. These applications are used in banking and e-commerce systems wherein the system is built to flag any unusual transactions. All this helps to detect fraudulent transactions.

Clustering Systems

These forms of systems are still in the initial stages, but their applications are numerous and can drastically change the way business is conducted. In this system, the user is classified into different clusters according to various behavioral factors like their age group, the region they live in or even the kind of programs they like to view. According to this clustering, the business can now suggest different programs or shows a user might be interested in watching according to the cluster that the said user belongs to during classification.

Categories of Machine Learning Systems

In the case of traditional machines, the programmer will give the machine a set of instructions and the input parameters, which the machine will use to compute make some calculations and derive an output using specific commands. In the case of Machine Learning systems, however, the system is never restricted by any command that the engineer provides, the machine will

choose the algorithm that it can be used to process the data set and decide the output with high accuracy. It does this, by using the training data set which consists of historical data and output.

Therefore, in the classical world, we will tell the machine to process data based on a set of instructions, while in the Machine Learning setup, we will never instruct a system. The computer will have to interact with the data set, develop an algorithm using the historical data set, make decisions like a human being would, analyze the information and then provide an output. The machine, unlike a human being, can process large data sets in short periods and provide results with high accuracy.

There are different types of Machine Learning algorithms, and they are classified based on the purpose of that algorithm. There are three categories in Machine Learning systems:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforced Learning

Supervised Learning

In this model, the engineers feed the machine with labeled data. In other words, the engineer will determine what the output of the system or specific data sets should be. This type of algorithm is also called a predictive algorithm.

For example, consider the following table:

Currency (label)	**Weight (Feature)**
1 USD	10 gm
1 EUR	5 gm
1 INR	3 gm
1 RU	7 gm

In the above table, each currency is given an attribute of weight. Here, the currency is the label, and the weight is the attribute or feature.

The supervised Machine Learning system with first we fed with this training data set, and when it comes across any input of 3 grams, it will predict that

the coin is a 1 INR coin. The same can be said for a 10-gram coin.

Classification and regression algorithms are a type of supervised Machine Learning algorithms. Regression algorithms are used to predict match scores or house prices, while classification algorithms identify which category the data should belong to.

We will discuss some of these algorithms in detail in the later parts of the book, where you will also learn how to build or implement these algorithms using Python.

Unsupervised Learning

In this type of model, the system is more sophisticated in the sense that it will learn to identify patterns in unlabeled data and produce an output. This is a kind of algorithm that is used to draw any meaningful inference from large data sets. This model is also called the descriptive model since it uses data and summarizes that data to generate a description of the data sets. This model is often used in data mining applications that involve large volumes of unstructured input data.

For instance, if a system is Python input of name, runs and wickets, the system will visualize that data on a graph and identify the clusters. There will be two clusters generated - one cluster is for the batsman while the other is for the bowlers. When any new input is fed, the person will certainly fall into one of these clusters, which will help the machine predict whether the player is a batsman or a bowler.

Name	**Runs**	**Wickets**
Rachel	100	3
John	10	50
Paul	60	10
Sam	250	6
Alex	90	60

Sample data set for a match. Based on this, the cluster model can group the players into batsmen or bowlers.

Some common algorithms which fall under unsupervised Machine Learning

include density estimation, clustering, data reduction and compressing.

The clustering algorithm summarizes the data and presents it differently. This is a technique used in data mining applications. Density estimation is used when the objective is to visualize any large data set and create a meaningful summary. This will bring us the concept of data reduction and dimensionality. These concepts explain that the analysis or output should always deliver the summary of the data set without the loss of any valuable information. In simple words, these concepts say that the complexity of data can be reduced if the derived output is useful.

Reinforced learning

This type of learning is similar to how human beings learn, in the sense that the system will learn to behave in a specific environment, and take actions based on that environment. For example, human beings do not touch fire because they know it will hurt and they have been told that will hurt. Sometimes, out of curiosity, we may put a finger into the fire, and learn that it will burn. This means that we will be careful with fire in the future.

The table below will summarize and give an overview of the differences between supervised and unsupervised Machine Learning. This will also list the popular algorithms that are used in each of these models.

Supervised Learning	**Unsupervised Learning**
Works with labeled data	Works with unlabeled data
Takes Direct feedback	No feedback loop
Predicts output based on input data. Therefore also called “Predictive Algorithm”	Finds the hidden structure/pattern from input data. Sometimes called as “Descriptive Model”
Some common classes of supervised algorithms include: Logistic Regression Linear Regression (Numeric prediction) Polynomial Regression Regression trees (Numeric	Some common classes of unsupervised algorithms include: Clustering, Compressing, density estimation & data reduction K-means Clustering (Clustering) Association Rules (Pattern Detection)

prediction) Gradient Descent Random Forest Decision Trees (classification) K-Nearest Algorithm (classification) Naive Bayes Support Vector Machines	Singular Value Decomposition Fuzzy Means Partial Least Squares Hierarchical Clustering Principal Component Analysis

We will look at each of these algorithms briefly and learn how to implement them in Python. Let us now look at some examples of where Machine Learning is applied. It is always a good idea to identify which type of Machine Learning model you must use with examples. The following points are explained in the next section:

- Facebook face recognition algorithm
- Netflix or YouTube recommending programs based on past viewership history
- Analyzing large volumes of bank transactions to guess if they are valid or fraudulent transactions.
- Uber's surge pricing algorithm

Steps in building a Machine Learning System

Regardless of the model of Machine Learning, here are the common steps that are involved in the process of designing a Machine Learning system.

Define Objective

As with any other task, the first step is to define the purpose you wish to accomplish with your system. The kind of data you use, the algorithm and other factors will primarily depend on the objective or the kind of prediction you want the system to produce.

Collect Data

This is perhaps the most time-consuming steps of building a system of Machine Learning. You must collect all the relevant data that you will use to train the algorithm.

Prepare Data

This is an important step that is usually overlooked. Overlooking this step can prove to be a costly mistake. The cleaner and the more relevant the data you are using is, the more accurate the prediction or the output will be.

Select an Algorithm

There are different algorithms that you can choose, like Structured Vector Machine (SVM), k-nearest, Naive-Bayes, Apriori, etc. The algorithm that you use will primarily depend on the objective you wish to attain with the model.

Train Model

Once you have all the data ready, you must feed it into the machine and the algorithm must be trained to predict.

Test Model

Once your model is trained, it is now ready to start reading the input to generate appropriate outputs.

Predict

Multiple iterations will be performed and you can also feed the feedback into the system to improve its predictions over time.

Deploy

Once you test the model and are satisfied with the way it is working, the said model will be sterilized and can be integrated into any application you want. This means that it is ready to be deployed.

All these steps can vary according to the application and the type of algorithm (supervised or unsupervised) you are using. However, these steps are generally involved in all processes of designing a system of Machine Learning. There are various languages and tools that you can use in each of these stages. In this book, you will learn about how you can design a system of Machine Learning using Python.

Let us understand the scenarios from the previous section below.

Scenario One

In a picture from a tagged album, Facebook recognizes the photo of the friend.

Explanation: This is an instance of supervised learning. In this case, Facebook is using tagged photographs to recognize the person. The tagged

photos will become the labels of the pictures. Whenever a machine is learning from any form of labeled data, it is referred to as supervised learning.

Scenario Two

Suggesting new songs based on someone's past music preferences.

Explanation: This is an instance of supervised learning. The model is training classified or pre-existing labels- in this case, the genre of songs. This is precisely what Netflix, Pandora, and Spotify do – they collect the songs/movies that you like, evaluate the features based on your preferences and then come up with suggestions of songs or movies based on similar features.

Scenario Three

Analyzing the bank data to flag any suspicious or fraudulent transactions.

Explanation: This is an instance of unsupervised learning. The suspicious transaction cannot be fully defined in this case and therefore, there are no specific labels like fraud or not a fraud. The model will try to identify any outliers by checking for anomalous transactions.

Scenario Four

Combination of various models.

Explanation: The surge pricing feature of Uber is a combination of different models of Machine Learning like the prediction of peak hours, the traffic in specific areas, the availability of cabs and clustering is used to determine the usage pattern of users in different areas of the city.

Linear Regression with Python

Linear regression with one variable

The first part of linear regression that we are going to focus on is when we just have one variable. This is going to make things a bit easier to work with and will ensure that we can get some of the basics down before we try some of the things that are a bit harder. We are going to focus on problems that have just one independent and one dependent variable on them.

To help us get started with this one, we are going to use the set of data for car_price.csv so that we can learn what the price of the car is going to be. We will have the price of the car be our dependent variable and then the year of the car is going to be the independent variable. You can find this information in the folders for Data sets that we talked about before. To help us make a good prediction on the price of the cars, we will need to use the Scikit Learn library from Python to help us get the right algorithm for linear regression. When we have all of this setup, we need to use the following steps to help out.

Importing the right libraries

First, we need to make sure that we have the right libraries to get this going. The codes that you need to get the libraries for this section include:

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

You can implement this script into the Jupyter notebook. The final line needs to be there if you are using the Jupyter notebook, but if you are using Spyder, you can remove the last line because it will go through and do this part without your help.

Importing the dataset

Once the libraries have been imported using the codes that you had before, the next step is going to be importing the data sets that you want to use for this training algorithm. We are going to work with the “car_price.csv” dataset. You can execute the following script to help you get the data set in the right place:

car_data = pd.read_csv('D:\Datasets\car_price.csv')

Analyzing the data

Before you use the data to help with training, it is always best to practice and analyze the data for any scaling or any values that are missing. First, we need to take a look at the data. The head function is going to return the first five rows of the data set you want to bring up. You can use the following script to help make this one work:

```
car_data.head()
```

Also, the described function can be used to return to you all of the statistical details of the dataset.

```
car_data.describe ()
```

Finally, let's take a look to see if the linear regression algorithm is going to be suitable for this kind of task. We are going to take the data points and plot them on the graph. This will help us to see if there is a relationship between the year and the price. To see if this will work out, use the following script:

```
plt.scatter(car_data['Year'], car_data['Price'])
plt.title("Year vs Price")
plt.xlabel("Year")
plt.ylabel("Price")
plt.show()
```

When we use the above script, we are trying to work with a scatterplot that we can then find on the library Matplotlib. This is going to be useful because this scatter plot is going to have the year on the x-axis and then the price is going to be over on our y-axis. From the figure for the output, we can see that when there is an increase in the year, then the price of the car is going to go up as well. This shows us the linear relationship that is present between the year and the price. This is a good way to see how this kind of algorithm can be used to solve this problem.

Going back to data pre-processing

Now we need to use that information and have these two tasks come up for us. To divide the data into features and labels, you will need to use the script below to get it started:

```
features = car_data.iloc[:,0:1].values
labels = car_data.iloc[:,1].values
```

Since we only have two columns here, the 0th column is going to contain the feature set and then the first column is going to contain the label. We will then be able to divide up the data so that there are 20 percent to the test set and 80 percent to the training. Use the following scripts to help you get this done:

```
from sklearn.model_selection import train_test_split
train_features, test_features, train_labels
test_labels = train_test_split (features, labels, test_size = 0.2, random_state
= 0)
```

From this part, we can go back and look at the set of data again. And when we do this, it is easy to see that there is not going to be a huge difference between the values of the years and the values of the prices. Both of these will end up being in the thousands each. What this means is that you don't need to do any scaling because you can just use the data as you have it here. That saves you some time and effort in the long run.

How to train the algorithm and get it to make some predictions

Now it is time to do a bit of training with the algorithm and ensure that it can make the right predictions for you. This is where the LinearRegression class is going to be helpful because it has all of the labels and other training features that you need to input and train your models.

This is simple to do and you just need to work with the script below to help you to get started:

```
from sklearn.linear_model import LinearRegresison
lin_reg = LinearRegression()
lin_reg.fit (train_features, train_labels)
```

Using the same example of the car prices and the years from before, we are going to look and see what the coefficient is for only the independent variable. We need to use the following script to help us do that:

```
print(lin_reg.coef_)
```

The result of this process is going to be 204.815. This shows that for each unit change during the year, the car price is going to increase by 204.815 (at least in this example).

Once you have taken the time to train this model, the final step to use is to predict the new instance that you are going to work with. The predicted method is going to be used with this kind of class to help see this happen. The

method is going to take the test features that you choose and add them in as the input, and then it can predict the output that would correspond with it the best. The script that you can use to make this happen will be the following:

```
predictions = lin_reg.predict( test_features)
```

When you use this script, you will find that it is going to give us a good prediction of what we are going to see in the future. We can guess how much a car is going to be worth based on the year it is produced in the future, going off the information that we have right now. There could be some things that can change with the future, and it does seem to matter based on the features that come with the car. But this is a good way to get a look at the cars and get an average of what they cost each year, and how much they will cost in the future.

So, let's see how this would work. We now want to look at this linear regression and figure out how much a car is going to cost us in the year 2025. Maybe you would like to save up for a vehicle and you want to estimate how much it is going to cost you by the time you save that money. You would be able to use the information that we have and add in the new year that you want it based on, and then figure out an average value for a new car that year.

Of course, remember that this is not going to be 100 percent accurate. Inflation could change prices, the manufacturer may change some things up, and more. Sometimes the price is going to be lower, and sometimes higher. But it at least gives you a good way to predict the price of the vehicle that you have and how much it is going to cost you in the future.

Lists In Python

We create a list in Python by placing items called elements inside square brackets separated by commas. The items in a list can be of mixed data types.

Start IDLE. Navigate to the File menu and click New Window.

Type the following:

```
list_mine=[] #empty list
list_mine=[2,5,8] #list of integers
list_mine=[5,"Happy", 5.2] #list having mixed data types
```

Exercise

Write a program that captures the following in a list: "Best", 26, 89, 3.9

Nested Lists

A nested list is a list as an item in another list.

Example

```
list_mine=["carrot", [9, 3, 6], ['g']]
```

Exercise

Write a nested list for the following elements: [36, 2, 1], "Writer", 't', [3.0, 2.5]

Accessing Elements from a List

In Python, the first element in a vector is always indexed as zero. A list of five items can be accessed by index0 to index4. An index error will occur if you fail to access the items in a list. The index is always an integer, so using other numbers will also create a type error.

Example

```
list_mine=['b','e','s','t']
print(list_mine[0])#the output will be b
print(list_mine[2])#the output will be s
print(list_mine[3])#the output will be t
```

Exercise

Given the following list:

your_collection=[‘t’,‘k’,‘v’,‘w’,‘z’,‘n’,‘f’]

a. Write a Python program to display the second item in the list
b. Write a Python program to display the sixth item in the last
c. Write a Python program to display the last item in the list

Nested List Indexing

```
nested_list=[“Best’,[4,7,2,9]]
print(nested_list[0][1])
```

Python Negative Indexing

For its sequences, Python allows negative indexing. The last item on the list is index-1, index -2 is the second last item and so on.

```
list_mine=[‘c’,‘h’,‘a’,‘n’,‘g’,‘e’,‘s’]
print(list_mine[-1])#Output is s
print(list_mine [-4])##Output is n
```

Slicing Lists in Python

Slicing operator (full colon) is used to access a range of elements in a list.

Example

```
list_mine=[‘c’,’h’,’a’,’n’,’g’,’e’,’s’]
print(list_mine[3:5]) #Picking elements from the fourth to the sixth
```

Example

Picking elements from start to the fifth.

```
print(list_mine[:-6])
```

Example

Picking the third element to the last.

```
print(list_mine[2:])
```

Exercise

Given class_names=[‘John’, ‘Kelly’, ‘Yvonne’, ‘Una’,’Lovy’,’Pius’, ‘Tracy’]

a. Write a Python program using the slice operator to display from the second students and the rest.
b. Write a Python program using the slice operator to display the first

student to the third using the negative indexing feature.

c. Write a Python program using the slice operator to display the fourth and fifth students only.

Manipulating Elements in a List using the Assignment Operator

```
list_yours=[4,8,5,2,1]
list_yours[1]=6
print(list_yours) #The output will be [4,6,5,2,1]
```

Changing a Range of Items in a List

```
list_yours[0:3]=[12,11,10] #Will change first item to fourth item in the list
print(list_yours) #Output will be: [12,11,10,1]
```

Appending/Extending Items in the List

The append() method allows extending the items on the list. The extend() can also be used.

Example

```
list_yours=[4, 6, 5]
list_yours.append(3)
print(list_yours)#The output will be [4,6,5, 3]
```

Example

```
list_yours=[4,6,5]
list_yours.extend([13,7,9])
print(list_yours)#The output will be [4,6,5,13,7,9]
```

The plus operator (+) can also be used to combine two lists. The * operator can be used to iterate a list a given number of times.

Example

```
list_yours=[4,6,5]
print(list_yours+[13,7,9])# Output:[4, 6, 5,13,7,9]
print(['happy']*4)#Output:["happy","happy", "happy","happy"]
```

Removing or Deleting Items from a List

The keyword del is used to delete elements or the entire list in Python.

```
list_mine=['t','r','o','g','r','a','m']
del list_mine[1]
```

```
print(list_mine) #t, o, g, r, a, m
```

Deleting Multiple Elements

```
del list_mine[0:3]
print(list_mine) #a, m
```

Delete Entire List

```
delete list_mine
print(list_mine) #will generate an error of lost not found
```

The remove() method or pop() method can be used to remove the specified item. The pop() method will remove and return the last item if the index is not given and helps implement lists as stacks. The clear() method is used to empty a list.

Example

```
list_mine=['t','k','b','d','w','q','v']
list_mine.remove('t')
print(list_mine)#output will be ['t','k','b','d','w','q','v']
print(list_mine.pop(1))#output will be 'k'
print(list_mine.pop())#output will be 'v'
```

Exercise

Given list_yours=['K','N','O','C','K','E','D']

a. Pop the third item in the list, save the program as list1.
b. Remove the fourth item using remove() method and save the program as list2
c. Delete the second item in the list and save the program as list3.
d. Pop the list without specifying an index and save the program as list4.

Using Empty List to Delete an Entire or Specific Elements

```
list_mine=['t','k','b','d','w','q','v']
list_mine[1:2]=[]
print(list_mine)#Output will be ['b','d','w','q','v']
```

Summary of List Methods in Python

Method	Description

insert()	Inserts an item at the defined index
append()	Adds an element to the end of the list
pop()	Removes and returns an element at the given index
index()	Returns the index of the first matched items
remove()	Removes an item from the list
copy()	Returns a shallow copy of the list
count()	Returns the count of number of items passed as an argument
clear()	Removes all items from the list
sort()	Sorts items in a list in ascending order
extend()	Adds all elements of a list to another list
reverse()	Reverses the order of items in the list

Inbuilt Python Functions to manipulate Python Lists

Method	Description
enumerate()	Returns an enumerated object and contains the index and value of all the items of the list as tuple
sorted()	Returns a new sorted list but does not sort the list itself
sum()	Returns the sum of all the elements in the list
max()	Returns the largest item in the list
len()	Returns the length of the list
any()	Returns True if any element of the list is true – if

	the list is empty, returns False
min()	Returns the smallest item in the list
all()	Returns True if all elements of the list are True

Exercise

Use list access methods to display the following items in reverse order list_yours=[4,9,2,1,6,7]

Use the list access method to count the elements in list_yours.

Use the list access method to sort the items in list_yours in an ascending order/default.

Modules In Python

Modules, also known as packages, are a set of names. This is usually a library of functions and/or object classes that are made available to be used within different programs. We used the notion of modules earlier in this chapter to use some function from the math library. In this chapter, we are going to cover in-depth on how to develop and define modules. To use modules in a Python program, the following statements are used: import, from, reload. The first one imports the whole module. The second allows import only a specific name or element from the module. The third one, reload, allows reloading a code of a module while Python is running and without stopping in it. Before digging into their definition and development, let’s start first by the utility of modules or packages within Python.

Modules Concept and Utility Within Python

Modules are a very simple way to make a system component organized. Modules allow reusing the same code over and over. So far, we were working in a Python interactive session. Every code we have written and tested is lost once we exit the interactive session. Modules are saved in files that make them persistent, reusable, and sharable. You can consider modules as a set of files where you can define functions, names, data objects, attributes, and so on. Modules are a tool to group several components of a system in a single place. In Python programming, modules are among the highest-level unit. They point to the name of packages and tools. Besides, they allow the sharing of the implemented data. You only need one copy of the module to be able to use across a large program. If an object is to be used in different functions and programs, coding it as a module allows share it with other programmers.

To have a sense of the architecture of Python coding, we go through some general structure explanation. We have been using so far in this book very simple code examples that do not have a high-level structure. In large applications, a program is a set of several Python files. By Python files, we mean files that contain Python code and have a .py extension. There is one main high-level program and the other files are the modules. The high-level file consists of the main code that dictates the control flow and executes the application. Module files define the tools that are needed to process elements and components of the main program and maybe elsewhere. The main program makes use of the tools that are specified in the modules.

In their turn, modules make use of tools that are specified in other modules. When you import a module in Python, you have access to every tool that is declared or defined in that specific module. Attributes are the variables or the functions associated with the tools within a module. Hence, when a module is imported, we have access to the attributes of the tools as well to process them. For instance, let's consider we have two Python files named file1.py and file2.py where the file1.py is the main program and file2.py is the module. In the file2.py, we have a code that defines the following function:

```
def Xfactorial (X):
P = 1
for i in range (1, X + 1):
P *= i
return P
```

To use this function in the main program, we should define code statements in the file1.py as follows:

```
import file2
A = file2.Xfactorial (3)
```

The first line imports the module file2.py. This statement means to load the file file2.py. This gives access to the file1.py to all tools and functions defined in file2.py by the name file2. The function Xfactorial is called by the second line. The module file2.py is where this function is defined using the attributes' syntax. The line file2.Xfactorial() means fetch any name value of Xfactorial and lies within the code body of file2. In this example, it is a function that is callable. So, we have provided an input argument and assigned the output result to the variable A. If we add a third statement to print the variable A and run the file file1.py, it would display 6 which is the factorial of 3. Along with Python, you will see the attribute syntax as object.attribute. This allows calling the attributes that might be a function or data object that provides properties of the object.

Note that some modules that you might import when programming with Python are available in Python itself. As we have mentioned at the beginning of this book, Python comes with a standard large library that has built-in modules. These modules support all common tasks that might be needed in programming from operating system interfaces to graphical user interface. They are not part of the language. However, they can be imported and comes with a software installation package. You can check the complete list of

available modules in a manual that comes with the installation or goes to the official Python website: www.Python.org. This manual is kept updated every time a new version of Python is released.

How to Import a Module

We have talked about importing a module without really explaining what happens behind in Python. The Imports are a very fundamental concept in Python programming structure. In this section, we are going to cover in-depth how really Python imports modules within a program. Python follows three steps to import a file or a module within the work environment of a program. The first step consists of finding the file that contains the module. The second step consists of compiling the module to a byte-code if required. Finally, the third step runs the code within the module file to build the objects that are defined. These three steps are run only when the module is imported for the first time during the execution of a program. This module and all its objects are loaded in the memory. When the module is imported further in the program, it skips all three steps and just fetch the objects defined by the module and are saved in memory.

At the very first step of importing a module, Python has to find the module file location. Note that, so far in the examples we presented, we used import without providing the complete path of the module or extension .py. We just used import math, or import file2.py (an example of the previous section). Python import statement omits the extension and the path. We just simply import a module by its name. The reason for this is that Python has a module that looks for paths called 'search path module'. This module is used specifically to find the path of the module files imported by using import statements.

In some cases, you might need to configure the path search of modules to be able to use new modules that are not part of the standard library. You need to customize it to include these new modules. The search path is simply the concatenation of the home directory, directories of PYTHONPATH, directories of the standard library, and optionally if the content of files with extension .pth when they exist. The home directory is set automatically by the system to a directory of Python executable when launched from the interactive session, or it can be modified to the working directory where your program is saved. This directory is the first to be searched when import a module is run without a path. Hence, if your home directory points to a

directory that includes your program along with the modules, importing these modules does not require any path specification.

The directory of the standard library is also searched automatically. This directory contains all default libraries that come with Python. The directories of PYTHONPATH can be set to point toward the directory of new modules that are developed. In fact, PTYHONPATH is an environment variable that contains a list of directories that contains Python files. When PTYHONPATH is set, all these paths are included in the Python environment and the search path directory would search these directories too when importing modules. Python also allows defining a file with .pth extension that contains directories, one in each line. This file serves the same as PTYHONPATH when included appropriately in a directory. You can check the directories' paths included when you run Python using sys.path. You simply print sys.path to get the list of the directories that Python will be searching for.

Remember, when importing a module, we just use the name of the module without its extension. When Python is searching for a module in its environment paths, it selects the first name that matches the module name regardless of the extension. Because Python allows using packages that are coded in other languages, it does not simply select a module with .py extension but a file name or even a zip file name that matches the module name being imported. Therefore, you should name your modules distinctly and configure the search path in a manner that makes it obvious to choose a module.

When Python finds the source code of the module file with a name that corresponds to the name in the import statement, it will compile it into byte code in case it is required. This step is skipped if Python finds an already byte code file with no source code. If the source code has been modified, another byte code file is automatically regenerated by Python while the program runs in other further executions. Byte code files have typically .pyc extension. When Python is searching and finds the module file name, it will load the byte code file that corresponds to the latest version of the source code with .py extension. If the source code is newer than the byte code file, it will generate a new one by compiling the source code file. Note that only imported files have corresponding files with .pyc extension. These files, the byte code files, are stored on your machine to make the imports faster in

future use.

The third step of the import statement is running the module's byte code. Each statement and each assignment in the file are executed. This allows generating any function, data objects, and so on defined in the module. The functions and all attributes are accessed within the program via importers. During this step, you will see print statements if they exist. The 'def ' statement will create a function object to be used in the main program.

To summarize the import statement, it involves searching for the file, compiling it, and running the byte code file. All other import statements use the module stored in memory and ignore all the three steps. When first imported, Python will look in the search path module to select the module. Hence, it is important to configure correctly the path environment variable to point to the directory that contains newly defined modules. Now that you have the big picture and the concept of modules, let's explore how we can define and develop new modules.

How to write and use a module in Python?

Modules in Python can be created very easily and do not require any specific syntax. Modules are simply files with a .py extension that contains Python code. You can use a text editor like Notepad++ to develop and write modules, then save them in files with the .py extension. Then, you just import these files like we have seen in the previous section to make use of the contained code.

When you create a module, all the data object including functions that are defined becomes the module attributes. These attributes are accessed and used via the attribute syntax like follows: module.attribute. For instance, if we define a module named ' MyModule.py ' that has the following function:

```
def Myfct (A):
print (' A by 2 is: ', A * 2)
return A * 2
```

The function 'Myfct' becomes the attribute of the module 'MyModule.py'. You can call a module any Python code that you develop and save in a file with a .py extension if you are importing them in later use. Module names are referenced variables. Hence, when naming a module, you should follow the same rules as for variable naming. You might be able to name your module anything you want. But if the rules are not respected, Python throws an error.

For instance, if you name your module $2P.py, you will not be able to import it and Python would trigger a syntax error. Directory names that contain the module and Python packages should follow the same rules. Also, their names cannot contain any space. In the rest of this section, we are going to provide some code examples of defining and using modules.

Two statements can be employed to make use of a module. The first one is the import statement we have covered in the previous section. Let’s consider again the previous example to illustrate a module 'MyModule.py' that contains ' Myfct' function:

```
def Myfct(A):
print (A, 'by 2 is: ', A * 2)
```

Now, to use this module, we import it using the following statements:

```
>>> import MyModule
>>> MyModule.Myfct(2)
2 by 2 is: 4
```

Now, the MyModule name is being used by Python to load the file and as a variable in the program. The module name should be used to access all its attributes. Another way to import and use a module attribute is by using the 'from import' statement. This statement works in the same manner as the import statement we have been using. Instead of using the module name to fetch for its attributes, we can access the attributes by their names directly. For example:

```
>>> from MyModule import Myfct
>>> Myfct (2)
2 by 2 is: 4
```

This statement makes a copy of the function name without using the module name. There is another form of 'from import' statement that uses an *. This statement allows copying all names that are assigned to objects in the module. For example:

```
>>> from MyModule import *
>>> Myfct (2)
2 by 2 is: 4
```

Because modules names become variables (i.e. references to objects), Python supports importing a module with an alias. Then we can access its attributes using the alias instead of its name. For instance, we can attribute an alias to

our module as follows:

```
>>> import Mymodule as md
>>> md.Myfct(2)
2 by 2 is: 4
```

Data objects other than functions are accessed the same way with attribute syntax. For instance, we can define and initialize data objects in modules than used them later in the program. Let’s consider the following code to create a module named ExModule.py.

```
A = 9
Name = 'John'
```

In this example, we initialize both variables A and Name.

Now, after importing the module, we can get both variables as follows:

```
>>> import ExModule
>>> print ('A is: ', ExModule.A)
A is: 9
>>> print ('Name is: ', Exmodule.Name)
Name is: John
```

Or we can assign attributes to other variables. For instance:

```
>>> import ExModule
>>> B = ExModule.A
>>> print ('B is: ', B)
B is: 9
```

If we use the 'from import' statement to import the attributes, the names of the attributes become variables in the script. For example:

```
>>> from Exmodule import A, Name
>>> print ('A is: ', A, 'and Name is: ', Name)
A is 9 and Name is John
```

Note that the 'from import' statement supports importing multiple attributes in one single line. Python allows changing objects that are sharable. For instance, let’s consider the following code to define the module named ExModul1.py:

```
A = 9
MyList = [90, 40, 80]
```

Now, let’s import this module and try to change the values of the attributes to

see how Python behaves.

```
>>> from ExModule1 import A, MyList
>>> A = 20
>>> myList [0] = 100
```

Now, let’s re-import the module and print those two attributes and see what changes Python has made.

```
>>> import ExModule1
>>> print ('A is: ', ExModule1.A)
A is: 9
>>> print ('My list is: ', ExModule.myList)
My list is: [100, 40, 80]
```

You can notice that Python has changed the value of the first element of the list but did not change the value of the variable 'A' to the value we assigned before. The reason is that when a mutable object like lists is changed locally, the changes apply also in the module from which they were imported. Reassigning a fetched variable name does not reassign the reference in the module from which it was imported. In fact, there is no link between the reference variable name copied and the file it was copied from. To make a valid modification in the script and the module it is imported from, we should use the import statement like follows:

```
>>> import ExModule1
>>> ExModule1.A = 200
```

The difference between changing the attributes 'A' and 'myList' is the fact that 'A' is a variable name and 'myList' is an object data. That is why modification to the variable 'A' should use import to be applied in the module file, too.

We have mentioned that importing a module for the first time in a script implies going through three steps that are searching for the module, compiling the module, and running the module. All other imports of the module later in the script skip all these three steps and access to module loaded in the memory. Now, let’s try an example to see how this works. Consider we have a module with the following code and named ExModule2.py:

```
print (' Hello World\n')
print (' This is my first module in Python')
A = 9
```

Now, let’s import this module and see how Python behaves when importing this module:

```
>>> import ExModule2
Hello World
```

This is my first module in Python

You can notice that when importing this module, it displays both messages. Now, let’s try to reassign a value to the attribute ' A', then re-import the module with the import statement.

```
>>> ExModule.A = 100
>>> import Exmodule2
```

As you can note from the example, Python did not display the messages, ' Hello World' and ' This is my first module in Python' because it did not re-run the module. It just used the module that is already loaded in the memory.

To make Python goes through all steps of importing a module for the second time in a script, we should use the reload statement. When using this statement, we force Python to import the module as it would for the first time. Besides, it helps make modifications in the program while it is running without interrupting it. It also helps to see instantly the modifications that are made. The reload is a function and not a statement in Python that takes as argument a module that is already loaded in memory.

Because reload is a function and expects an argument, this argument should be already assigned an object which is a module object. If for some reason the import statement failed to import a module, you will not be able to reload it. You have to repeat the import statement until it imports the module successfully. Like any other function, the reload takes the module name reference between parentheses. The general form of using reload with import is as follows:

```
import module_name
list of statements that use module attributes
reload(module_name)
list of statements that use module attributes
```

The module object is changed by the reload function. Hence, any reference to that module in your scripts is impacted by the reload function. Those statements that use the module attributes will be using the values of the new attributes if they are modified. The reload function overwrites the module

source code and re-runs it instead of deleting the file and creating a new one. In the following code example, we will see a concrete illustration of the reload functioning. We consider the following code to create a module named ExModule3.py:

```
my_message = 'This is my module first version'
def display ():
print (my_message)
```

This module simply assigns a string to the variable 'my_message' and print it. Now, let’s import this module in Python and call the attribute function:

```
>>> import ExModule3
>>> Exmodule3.display()
This is my module first version
```

Now, go to your text editor and edit the module source code without stopping the Python prompt shell. You can make a change as follows:

```
my_message = 'This is my module second version edited in the text editor'
def display ():
print (my_message)
```

Now, back to the interactive session of Python in the prompt shell, you can try to import the module and call the function:

```
>>> import ExModule3
>>> Exmodule3.display()
This is my module first version
```

As you can notice that the message did not change although the source code file was modified. As said before, all imports after the first import use the already loaded module in memory.

To get the new message and access the modification made in the module, we use the reload function:

```
>>> reload (ExModule3)
<module 'ExModule3)>
>>> Exmodule3.display()
This is my module second version edited in the text editor
```

Note that the reload function re-runs module and returns the module object. Because it was executed in the interactive session, it displays < module name> by default.

Machine Learning Training Model

In Machine Learning, a model is a mathematical or digital representation of a real-world process. To build a good Machine Learning (ML) model, developers need to provide the right training data to an algorithm. An algorithm, on the other hand, is a hypothetical set taken before training begins with real-world data.

A linear regression algorithm, for example, is a set of functions defining similar characteristics or features as defined by linear regression. Developers choose the function that fits most of the training data from a set or group of functions. The process of training for Machine Learning involves providing an algorithm with training data.

The basic purpose of creating any ML model is to expose it to a lot of input, as well as the output applicable to it, allowing it to analyze these data and use it to determine the relationship between it and the results. For example, if a person wants to decide whether to carry an umbrella or not depending on the weather, he/she will need to look at the weather conditions, which, in this case, is the training data.

Professional data scientists spend more of their time and effort on the steps preceding the following processes:

1. Data exploration
2. Data cleaning
3. Engineering new features

Simple Machine Learning Training Model in Python

When it comes to Machine Learning, having the right data is more important than having the ability to write a fancy algorithm. A good modeling process will protect against over-fitting and maximize performance. In Machine Learning, data are a limited resource, which developers should spend doing the following:

- Feeding their algorithm or training their model
- Testing their model

However, they cannot reuse the same data to perform both functions. If they do this, they could over-fit their model and they would not even know. The

effectiveness of a model depends on its ability to predict unseen or new data; therefore, it is important to have separate training and test different sections of the dataset. The primary aim of using training sets is to fit and fine-tune one's model. Test sets, on the other hand, are new datasets for the evaluation of one's model.

Before doing anything else, it is important to split data to get the best estimates of the model's performance. After doing this, one should avoid touching the test sets until one is ready to choose the final model. Comparing training versus test performance allows developers to avoid over-fitting. If a model's performance is adequate or exceptional on the training data, but inadequate on the test data, then the model has this problem.

In the field of Machine Learning, over-fitting is one of the most important considerations. It describes how well the target function's approximation correlates with the training data provided. It happens when the training data provided has a high signal to noise ratio, which will lead to poor predictions.

Essentially, an ML model is over-fitting if it fits the training data exceptionally well while generalizing new data poorly. Developers overcome this problem by creating a penalty on the model's parameters, thereby limiting the model's freedom.

When professionals talk about tuning models in Machine Learning, they usually mean working on hyper-parameters. In Machine Learning, there are two main types of parameters, i.e., model parameters and hyper-parameters. The first type defines individual models and is a learned attribute, such as decision tree locations and regression coefficients.

The second type, however, defines higher-level settings for Machine Learning algorithms, such as the number of trees in a random forest algorithm or the strength of the penalty used in regression algorithms.

The process of training a machine-learning model involves providing an algorithm with training data. The term machine-learning model refers to the model artifact created by the ML training process. These data should contain the right answer, known as the target attribute. The algorithm looks for patterns in the data that point to the answer it wants to predict and creates a model that captures these different patterns.

Developers can use machine-learning models to generate predictions on new data for which they do not know the target attributes. Supposing a developer wanted to train a model to predict whether an email is legitimate or spam, for

example, he/she would give it training data containing emails with known labels that define the emails as either spam or not spam. Using these data to train the model will result in it trying to predict whether a new email is legitimate or spam.

Simple Machine Learning Python Model using Linear Regression

When it comes to building a simple ML model in Python, beginners need to download and install sci-kit-learn, an open-source Python library with a wide variety of visualization, cross-validation, pre-processing, and Machine Learning algorithms using a unified user-interface. It offers easy-to-understand and use functions designed to save a significant amount of time and effort. Developers also need to have Python Version 3 installed in their system.

Some of the most important features of sci-kit-learn include;

1. Efficient and easy-to-use tools for data analysis and data mining
2. BSD license
3. Reusable in many different contexts and highly accessible
4. Built on the top of matplotlib, SciPy, and NumPy
5. Functionality for companion tasks
6. Excellent documentation
7. Tuning parameters with sensible defaults
8. User-interface supporting various ML models

Before installing this library, users need to have SciPy and NumPy installed. If they already have a data set, they need to split it into training data, testing data, and validation data. However, in this example, they are creating their own training set, which will contain both the input and desired output values of the data set they want to use to train their model. To load an external dataset, they can use the Panda library, which will allow them to easily load and manipulate datasets.

Their input data will consist of random integer values, which will generate a random integer N; for instance, a <= N <= b. As such, they will create a function that will determine the output. Recall a function uses some input value to return some output value. Having created their training set, they will

split each row into an input training set and its related output training set, resulting in two lists of all inputs and their corresponding outputs.

Benefits of splitting datasets include:

1. Gaining the ability to train and test the model on different types of data than the data used for training
2. Testing the model's accuracy, which is better than testing the accuracy of out-of-sample training
3. Ability to evaluate predictions using response values for the test datasets

They will then use the linear regression method from Python's sci-kit-learn library to create and train their model, which will try to imitate the function they created for the ML training dataset. At this point, they will need to determine whether their model can imitate the programmed function and generate the correct answer or accurate prediction.

Here, the ML model analyzes the training data and uses it to calculate the coefficients or weights to assign to the inputs to return the right outputs. By providing it with the right test data, the model will arrive at the correct answer.

Conditional or Decision Statements

In programming, we normally set certain conditions and decide which particular action to perform depending on the conditions. To do this, Python uses the “if statement” to check the program current state before responding suitably to that state. However, in this chapter, you will be exposed to various ways to write conditional statements. Furthermore, you will learn basic “if statements,” create complex if statements and write loops to handle items in a list. There is so much more loaded in this chapter for you to learn. Without further ado, let us begin with a simple example.

The program below shows how you can use “if statement” to respond to a particular situation correctly. For instance, we have a list of colors and want to generate an output of different colors. Furthermore, the first letter should be in the title case of the lower case.

```
colors =[“Green”, “Blue”, “Red”, “Yellow”]
for color in colors:
print(color.title())
```

The output will be as follows:

```
Green
Blue
Red
Yellow
```

Consider another example where we want to print a list of cars. We have to print them in the title case since it is a proper name.

Additionally, the value “Kia” must be in uppercase.

```
cars = [“Toyota,” “Kia,” “Audi,” “Infinity”]
for car1 in cars:
if car1 == “kia”:
print(car1.upper())
else:
print(car1.title())
```

The loop first verifies if whether the current value of the car is “Kia.” If that is true, it then prints the value in uppercase. However, if it is not kia, it prints it in title case. The output will look like this:

```
Toyota
```

KIA
Audi
Infinity

The example above combines different concepts, which at the end of this chapter, you will learn. However, let us begin with the various conditional tests.

Conditional Tests in Python

In the center of any if statement lies an expression, which must be evaluated to be either true or false. This is what is normally known as a conditional test because Python uses both values to determine if a particular code should be executed. If the particular statement is true, Python executes the code that follows it. However, if it is false, it ignores the code after it.

Checking Equality

At times, we may test for the equality of a particular condition. In this situation, we test if the value of the variable is equal to the other variable we decide. For instance:

>>>color = “green”
>>> color == “green”
True

In this example, we first assign the variable color with the value “green by using the single equal sign. This is not something new, as we have been using it throughout this book. However, the second line checks if the value of color is green, which has a double equal sign. It will return true if the value on the left side and that on the right side are both true. If it doesn’t match, then the result will be false. When the value of the color is anything besides green, then this condition equates to false. The example below will clarify that.

>>>color = “green”
>>> color == “blue”
False

Note: When you test for equality, you should know that it is case sensitive. For instance, two values that have different capitalizations won’t be regarded as equal. For instance,

>>>color = “Green”
>>> color == “green”

False

If the case is important, then this is advantageous. However, if the case of the variable isn't important, and you want to check the values, then you can convert the value of the variable to lowercase before checking for equality.

```
>>>color = "Green"
>>> color.lower() == "green"
True
```

This code will return True irrespective of how to format the value "Green" is because the conditional tests aren't case sensitive. Please note that the lower() function we used in the program does not change the value originally stored in color.

In the same way, we can check for equality; we can also check for inequality in a program code. In checking for inequality, we verify if two values are not equal and then return it as true. To check for inequality, Python has its unique symbol, which is a combination of the exclamation sign with an equal sign (!=). Most programming language uses these signs to represent inequality.

The example below shows the use of if statement to test for inequality:

```
color = "green"
if color != "blue"
print("The color doesn't match")
```

In the second line, the interpreter matches the value of color to that of "blue." If the values match, then Python return false; however, if it is true, Python returns true before executing the statement following it "The color doesn't match"

The color doesn't match

Numerical Comparison in Python

We can also test numerical values in Python, but it is very straightforward. For instance, the code below determines if a person's age is 25 years old:

```
>>>myage = 25
>>>myage == 25
True
```

Additionally, we can also test if two numbers are unequal. Consider the code below.

```
number = 34
```

```
if number != 54:
print(“The number does not match. Please retry!”)
```

The first line declares number as a variable and stores the number “34” in it. The conditional statement begins in line two and passes through the line because the number 34 is not equal to 54. Since the code is indented, the code is then executed to produce

```
The number does not match. Please retry!
```

Besides this, you can perform various mathematical comparison inside your conditional expressions including greater than, greater than or equal to , less than, and less than or equal to.

```
>>> number = 22
>>> number <25
True
>>> number <= 25
True
>>> number > 25
False
>>> number >= 25
 False
```

Every mathematical comparison you want can be included as part of an “if statement” that allows you to detect the particular condition in question.

Creating Multiple Conditions

When writing code, some situations may warrant you to verify multiple conditions simultaneously. For instance, you require conditions to be false to take action. At times, you may want only one condition to be satisfied. In this situation, you can use the keyword “or” and “and.” Let first use the “and” keyword to check multiple conditions in Python programming.

Using “AND”

If you want to verify that two expressions are both true at the same time, the keyword “and” serves that purpose. The expression is evaluated to be true when both conditions test to return true. However, if one of the condition falls, then the expression returns false.

For instance, you want to ascertain if two students in a class have over 45 score marks.

```
>>> score_1 = 46
>>> score_2 = 30
>>> score_1 >=45 and score_2 >= 45
False
>>> score_2 = 47
>>> score_1 >= 45 and score_2 >= 45
True
```

The program looks complicated but lets me explain it step-by-step. In the first two lines, we define two scores, score_1, and score_2. However, in line 3, we perform a check to ascertain if both scores are equal to or above 45. The condition on the right-hand side is false, but that of the left-hand side is true. Then in the line after the false statement, I changed the value of score_2 from 30 to 47. In this instant, the value of score_2 is now greater than 46; therefore, both conditions will evaluate to true.

To make the code more readable, we can use parentheses in each test. However, it is not compulsory to do such but makes it simpler. Let us use parentheses to demonstrate the difference between the previous code and the one below.

```
(score_1 >= 45) and (score_2 >=45)
```

Using "OR"

The "OR" keyword allows you to check multiple conditions as the "AND" keyword. However, the difference here is that the "OR" keyword is used when you want to ascertain that one expression is true for multiple conditions. In this situation, if one of the expression is false, the condition returns true. It returns false when both conditions are false.

Let us consider our previous example using the "OR" keyword. For instance, you want to ascertain if two students in a class have over 45 score mark.

```
>>> score_1 = 46
>>> score_2 = 30
>>> score_1 >=45 or score_2 >= 45
True
>>> score_1 = 30
>>> score_1 >= 45 or score_2 >= 45
False
```

We began by declaring two variables score_1 and score_2 and assign values

to them. In the third line, we test the OR condition using the two variables. The test in that line satisfies the condition because one of the expressions is true. Then, it changed the value of the variable score to 30; however, it fails both conditions and therefore evaluates false.

Besides using the “And” and “OR” conditional statements to check multiple conditions, we can also test the availability of a value in a particular list. For instance, you want to verify if a requested username is already in existence from a list of usernames before the completion of online registration on a website.

To do this, we can use the “in” keyword in such a situation. For instance, let us use a list of animals in the zoo and check if it is already on the list.

```
>>>animals = [“zebra”, “lion”, “crocodile”, “monkey”]
>>> “monkey” in animals
True
>>> “rat” in animals
False
```

In the second and fourth lines, we use the “in” keyword to test if the request word in a double quote exists in our list of animals. The first test ascertains that “monkey” exists in our list, whereas the second test returns false because the rat is not in the animal's list. This method is significant because we can generate lists of important values and check the existence of the values in the list.

There are situations where you want to check if a value isn’t in a list. In such a case, instead of using the “in” keyword to return false, we can use the “not” keyword. For instance, let us consider a list of Manchester United players before allowing them to be part of their next match. In order words, we want to scan the real players and ensure that the club does not field an illegible player.

```
united_player = [“Rashford,” “Young,” “Pogba,” “Mata,” “De Gea”]
player = “Messi”
if player not in united_player:
print(f “{player.title()}, you are not qualified to play for Manchester
United.”)
```

The line “if player, not in united_player:” reads quite clearly. Peradventure, the value of the player isn’t in the list united_player, Python returns the expression to be True and then executed the line indented under it. The player

“Messi” isn’t part of the list united_player; therefore, he will receive a message about his qualification status. The output will be as follow:

Messi, you are not qualified to play for Manchester United.

Boolean Expressions in Python

If you have learned any programming language, you might have come across the term “Boolean Expression” because they are very important. A Boolean expression is another term to describe the conditional test. When evaluated, the outcome can only be either True or False. However, they are essential if your goal is to keep track of specific conditions like if a user can change content or light is switched on or not. For instance,

```
change_content = False
light_on = False
light_off = True
```

Boolean values provide the best means of tracking the particular condition of a program.

Exercises

Conditional Testing – Write various conditional expressions. Furthermore, print a statement to describe each condition and what the likely output of each test will be. for instance, your code can be like this:

```
car = “Toyota”
print(“Is car == ‘Toyota’? My prediction is True.”(
print (car == “Toyota”)
print(“\nIs car == ‘KIA’? My prediction is False.”)
print(car== “KIA”)
```

1. Test the following condition to evaluate either True or False using any things of your choice to form a list.
2. Test for both inequality and equality using strings and numbers
3. Test for the condition using the “or” and “and” keywords
4. Test if an item exists in the above list
5. Test if an item doesn’t exist in the list

If Statements

Since you now know conditional tests, it will be easier for you to under if

statements. There are various types of if statements to use in Python, depending on your choice. In this section, you will learn the different if statements possible and the best situation to apply them, respectively.

Simple If Statements

In any programming language, the "if statement" is the simplest to come across. It only requires a test or condition with a single action following it, respectively. The syntax for this statement is as follows:

```
if condition:
        perform action
```

The first line can contain any conditional statement with the second following the action to take. Ensure to indent the second line for clarity purposes. If the conditional statement is true, then the code under the condition is executed. However, if it is false, the code is ignored.

For instance, we have set a standard that the minimum score for a person to qualify for a football match is 20. We want to test if such a person is qualified to participate.

```
person = 21
if person >= 20
print("You are qualified for the football match against Valencia.")
```

In the first line, we define the person's age to 21 to qualify. Then the second line evaluates if the person is greater than or equal to 20. Python then executes the statement below because it fulfills the condition that the person is above 20.

```
You are qualified for the football match against Valencia.
```

Indentation is very significant when using the "if statement" like we did in the "for loop" situations. All indented lines are executed once the condition is satisfied after the if statement. However, if the statement returns false, then the whole code under it is ignored, and the program halted.

We can also include more code inside the if statements to display what we want. Let us add another line to display that the match is between Chelsea and Valencia at the Standford Bridge.

```
person  =21
if person >= 20
```

```
print(“You are qualified for the football match against Valencia.”)
print(“The match is between Arsenal and Valencia.”)
Print(“The Venue is at the Emirate Stadium in England.”)
```

The conditional statement passes through the condition and prints the indented actions once the condition is satisfied.

The output will be as follow:

```
You are qualified for the football match against Valencia.
The match is between Arsenal and Valencia.
The Venue is at the Emirate Stadium in England.
```

Assuming the age is less than 20, and then there won’t be any output for this program. Let us try another example before going into another conditional statement.

```
name = “Abraham Lincoln”
if name = “Abraham Lincoln”
print(“Abraham Lincoln was a great United State President.”)
print(“He is an icon that many presidents try to emulate in the world.”)
```

The output will be:

```
Abraham Lincoln was a great United State President.
He is an icon that many presidents try to emulate in the world.
```

If-else Statements

At times, you may want to take certain actions if a particular condition isn’t met. For example, you may decide what will happen if a person isn’t qualified to play a match. Python provides the if-else statements to make this possible. The syntax is as follows:

```
if conditional test
        perform statement_1
else
        perform statement_2
```

Let us use our football match qualification to illustrate how to use the if-else statement.

```
person  =18
if person >= 20:
print(“You are qualified for the football match against Valencia.”)
print(“The match is between Arsenal and Valencia.”)
```

```
print("The Venue is at the Emirate Stadium in England.")
else:
print("Unfortunately, you are not qualified to participate in the match.")
print("Sorry, you have to wait until you are qualified.")
```

The conditional test (if person>=20) is first evaluated to ascertain that the person is above 20 before it passes to the first indented line of code. If it is true, then it prints the statements beneath the condition. However, in our example, the conditional test will evaluate to false then passes control to the else section. Finally, it prints the statement below it since it fulfills that part of the condition.

```
Unfortunately, you are not qualified to participate in the match.
Sorry, you have to wait until you are qualified.
```

This program works because of the two possible scenarios to evaluate – a person must be qualified to play or not play. In this situation, the if-else statement works perfectly when you want Python to execute one action in two possible situations.

Let us try another.

```
station_numbers = 10
if station_numbers >=12:
print("We need additional 3 stations in this company.")
else:
print("We need additional 5 stations to meet the demands of our
audience.")
```

The output will be:

```
We need an additional 5 stations to meet the demands of our audience.
```

The if-elif-else Chain

At times, you may want to test three different conditions based on certain criteria. In such a situation, Python allows us to use the if-elif-else conditional statement to execute such a task. We have many real-life situations, require more than two possibilities. For instance, think of a cinema hall with different charge rates for different sets of people.

- Children under 5 years are free
- Children between 5 years and 17 years are $30
- People older than 18 years is $50

As you can see, there are three possible situations because the following set of people can attend the cinema to watch the movie of their choice. In this situation, how can you ascertain a person's rate? Well, the following code will illustrate that point and print out specific price rates for each category of people.

```
person_age = 13
if person_age < 5:
print("Your ticket cost is $0.")
elif person_age < 17:
print("Your ticket cost is $30.")
else:
print("Your ticket cost is $50)
```

The first line declares a variable "person_age" with value 13. Then we perform the first conditional statement to test if the person is below the age of 5. If it fulfills the condition, it prints the appropriate message, and the program halts. However, if it returns false, it passes to the elif line, which tests if the person_age is less than 17. At this post, the person's minimum age must be 5 years and not above 17. If the person is above 17, then Python skips the instruction and goes to the next condition.

In the example, we fix the person_age to 13. Therefore, the first test will evaluate false and won't execute the block of line. It then tests the elif condition, which in this case is true, and will print the message. The output will be:

```
Your ticket cost is $30.")
```

Nevertheless, if the age is above 17, then it will pass through the first two tests because it will evaluate to false. Then the next command will be the else condition, which will print the statement below.

We can rewrite this program in such a way that we won't have to include the message "Your ticket cost is…" all we need is to put the prince inside the if-elif-else chain with a simple print() method to execute after the evaluation of the chain. Look at the line of code below:

```
person_age = 13
if person_age < 5:
cost = 0
elif person_age < 17:
cost =30
```

```
else:
cost = 50
print(f “Your ticket cost is ${cost}.”)
```

In the third, fifth, and seventh lines, we defined the cost based on the person’s age. The cost price is already set within the if-elif-else statement. However, the last line uses the cost of each age to form the final cost of the ticket.

This new code will produce the same result as the previous example. However, the latter is more concise and straightforward. Instead of using three different print statements, our reverse code only uses a single print statement to print the cost of the ticket.

Multiple elif Blocks

You can also have more than one elif block in your program. For instance, if the manager of the cinema decides to implement special discounts for workers, this will require additional, conditional tests to the program to ascertain whether the person in question is qualified for such a discount. Assuming those above 55 years will pay 70% of the initial cost of each ticket. Then the program code will be as follows:

```
person_age = 13
if person_age < 5:
cost = 0
elif person_age < 17:
cost =30
elif person_age < 55
cost = 50
else:
cost = 35
print(f “Your ticket cost is ${cost}.”)
```

The cost is identical to our previous example; however, the only including is the “elif person_age < 55” and is respective else condition. This second elif block checks if the person’s age is less than 55 before assigning them the cost of the ticket for $50. However, the statement after the else needs to be changed. In this situation, it is applicable if the person’s age is above 55 years, which is this situation fulfills the condition we want.

The “else” statement isn’t compulsory because you can omit it and use the elif statement instead. At times, it is better to use additional elif statements to

capture specific interests. Let us see how to implement it without using the else statement.

```
person_age = 13
if person_age < 5:
cost = 0
elif person_age < 17:
cost =30
elif person_age < 55:
cost = 50
elif person_age >= 55:
cost = 35
print(f “Your ticket cost is ${cost}.”)
```

The additional elif statement helps to assign the ticket cost of “$30” to those above 30 years. This format is a bit clearer when compared with the else block.

Performing Multiple Conditions

Using the if-elif-else statement comes handy when especially when you want to pass only one test. Once the interpreter discovers that this test is passed, it skips other tests and halts the program. With this feature, you test a specific condition in a line of code.

Nevertheless, some situations may warrant you to check all the conditions available. In such a scenario, you can use multiple if statements without adding the elif or else blocks. This method becomes relevant when more than one of the condition returns true. For instance, let us consider the previous example of players in Manchester United to illustrate this. In this, we want to include the players in an upcoming match against their rivals Manchester City.

```
united_players = [“Rashford,” “Young,” “Pogba,” “Mata,” “De Gea”]
if “Young” in united_players:
print(“Adding Young to the team list.”)
if “De Gea” in united_players:
print(“Adding Dea Gea to the team list.”)
if “Messi” in united_players:
print(“Adding Messi to the team list.”)
print( “Team list completed for the match against Manchester City!”)
```

In the first line, we defined united_players as a variable with values Rashford, Young, Pogba, Mata, and De Gea. The second line uses the “if statement” to check if the person requested for Young. The same applies to the lines with the “if statement” and the condition is run regardless of the outcome of the previous tests. For this program above, the output will be:

Adding Young to the team list.
Adding Dea Gea to the team list.
Team list completed for the match against Manchester City!

If we decide to use the if-elif-else block, the code won’t function properly because once a particular test returns true, the program will stop.

Let us try it and see.

```
united_players = [“Rashford,” “Young,” “Pogba,” “Mata,” “De Gea”]
if “Young” in united_players:
print(“Adding Young to the team list.”)
elif  “De Gea” in united_players:
print(“Adding Dea Gea to the team list.”)
elif  “Messi” in united_players:
print(“Adding Messi to the team list.”)
print( “\ Team list completed for the match against Manchester City!”)
```

In this code, Python will evaluate the first condition, and once it is true, the program stops. The output for this program will be:

Adding Young to the team list.
Team list completed for the match against Manchester City!

Exercise

Consider the list of colors we have in the world. Create a variable name-color and assign the following colors to it – blue, red, black, orange, white, yellow, indigo, green.

Use an “if statement” to check if the color is blue. If the color is blue, then print a message indicating a score of 5 points.

Write a problem using the if-else chain to print if a particular selected is green.

Write another program using the if-elif-else chain to determine the scores of students in a class. Set a variable “score” to store the student’s score.

If the student’s score is below 40, indicate an output a message that such

student has failed.

If the student’s score is above 41 but less than 55, print a message that the student has passed.

Essential Libraries for Machine Learning in Python

Many developers nowadays prefer the usage of Python in their data analysis. Python is not only applied in data analysis but also statistical techniques. Scientists, especially the ones dealing with data, also prefer using Python in data integration. That's the integration of Webb apps and other environment productions.

The features of Python have helped scientists to use it in Machine Learning. Examples of these qualities include consistent syntax, being flexible and even having a shorter time in development. It also can develop sophisticated models and has engines that could help in predictions.

As a result, Python boasts of having a series or a set of very extensive libraries. Remember, libraries refer to a series of routines and sorts of functions with different languages. Therefore, a robust library can lead to tackling more complex tasks. However, this is possible without writing several code lines again. It is good to note that Machine Learning relies majorly on mathematics. That's mathematical optimization, elements of probability and also statistical data. Therefore, Python comes in with a rich knowledge of performing complex tasks without much involvement.

The following are examples of essential libraries being used in our present.

Scikit – Learn

Scikit learn is one of the best and a trendy library in Machine Learning. It has the ability to supporting learning algorithms, especially unsupervised and supervised ones.

Examples of Scikit learn include the following.

- *k-means*
- decision trees
- linear and logistic regression
- *clustering*

This kind of library has major components from NumPy and SciPy. Scikit learn has the power to add algorithms sets that are useful in Machine

Learning and also tasks related to data mining. That's, it helps in classification, clustering, and even regression analysis. There are also other tasks that this library can efficiently deliver. A good example includes ensemble methods, feature selection, and more so, data transformation. It is good to understand that the pioneers or experts can easily apply this if at all, they can be able to implement the complex and sophisticated parts of the algorithms.

TensorFlow

It is a form of algorithm which involves deep learning. They are not always necessary, but one good thing about them is their ability to give out correct results when done right. It will also enable you to run your data in a CPU or GPU. That's, you can write data in the Python program, compile it, then run it on your central processing unit. Therefore, this gives you an easy time in performing your analysis. Again, there is no need for having these pieces of information written at C++ or instead of other levels such as CUDA.

TensorFlow uses nodes, especially the multi-layered ones. The nodes perform several tasks within the system, which include employing networks such as artificial neutral, training, and even set up a high volume of datasets. Several search engines such as Google depend on this type of library. One main application of this is the identification of objects. Again, it helps in different Apps that deal with the voice recognition.

Theano

Theano too forms a significant part of Python library. Its vital tasks here are to help with anything related to numerical computation. We can also relate it to NumPy. It plays other roles such as;

- Definition of mathematical expressions
- Assists in the optimization of mathematical calculation
- Promotes the evaluation of expressions related to numerical analysis.

The main objective of Theano is to give out efficient results. It is a faster Python library as it can perform calculations of intensive data up to 100 times. Therefore, it is good to note that Theano works best with GPU as compared to the CPU of a computer. In most industries, the CEO and other

personnel use Theano for deep learning. Also, they use it for computing complex and sophisticated tasks. All these became possible due to its processing speed. Due to the expansion of industries with a high demand for data computation techniques, many people are opting for the latest version of this library. Remember, the latest one came to limelight some years back. The new version of Theano, that's, version 1.0.0, had several improvements, interface changes, and composed of new features.

Pandas

Pandas is a library that is very popular and helps in the provision of data structures that are of high level and quality. The data provided here is simple and easy to use. Again, it's intuitive. It is composed of various sophisticated inbuilt methods which make it capable of performing tasks such as grouping and timing analysis. Another function is that it helps in a combination of data and also offering filtering options. Pandas can collect data from other sources such as Excel, CSV, and even SQL databases. It also can manipulate the collected data to undertake its operational roles within the industries. Pandas consist of two structures that enable it to perform its functions correctly. That's Series, which has only one dimension and data frames that boast of two dimensional. The Pandas library has been regarded as the most strong and powerful Python library over the time being. Its main function is to help in data manipulation. Also, it has the power to export or import a wide range of data. It is applicable in various sectors, such as in the field of Data Science.

Pandas is effective in the following areas:

- Splitting of data
- Merging of two or more types of data
- Data aggregation
- Selecting or subsetting data
- Data reshaping

Diagrammatic explanation

Series Dimensional

A	7
B	8
C	9

D	3
E	6
F	9

Data Frames dimensional

	A	B	C	D
*0	0	0	0	0
*1	7	8	9	3
*2	14	16	18	6
*3	21	24	27	9
*4	28	32	36	12
*5	35	40	45	15

Applications of pandas in a real-life situation will enable you to perform the following:

- You can quickly delete some columns or even add some texts found within the Dataframe
- It will help you in data conversion
- Pandas can reassure you of getting the misplaced or missing data
- It has a powerful ability, especially in the grouping of other programs according to their functionality.

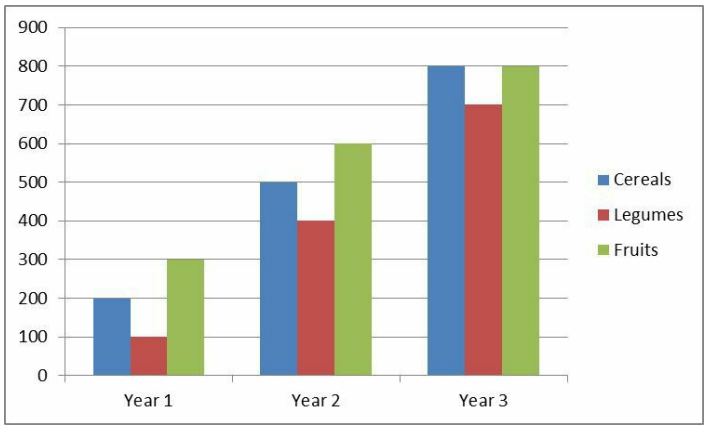
Matplotlib

This is another sophisticated and helpful data analysis technique that helps in data visualization. Its main objective is to advise the industry where it stands using the various inputs. You will realize that your production goals are meaningless when you fail to share them with different stakeholders. To perform this, Matplotlib comes in handy with the types of computation analysis required. Therefore, it is the only Python library that every scientist, especially the ones dealing with data prefers. This type of library has good looks when it comes to graphics and images. More so, many prefer using it in creating various graphs for data analysis. However, the technological world has completely changed with new advanced libraries flooding the industry.

It is also flexible, and due to this, you are capable of making several graphs that you may need. It only requires a few commands to perform this.

In this Python library, you can create various diverse graphs, charts of all kinds, several histograms, and even scatterplots. You can also make non-Cartesian charts too using the same principle.

Diagrammatic explanation



The above graph highlights the overall production of a company within three years. It specifically demonstrates the usage of Matplotlib in data analysis. By looking at the diagram, you will realize that the production was high as compared to the other two years. Again, the company tends to perform in the production of fruits since it was leading in both years 1 and 2 with a tie in year 3. From the figure, you realize that your work of presentation, representation and even analysis has been made easier as a result of using this library. This Python library will eventually enable you to come up with good graphics images, accurate data and much more. With the help of this Python library, you will be able to note down the year your production was high, thus, being in a position to maintain the high productivity season.

It is good to note that this library can export graphics and can change these graphics into PDF, GIF, and so on. In summary, the following tasks can be undertaken with much ease. They include:

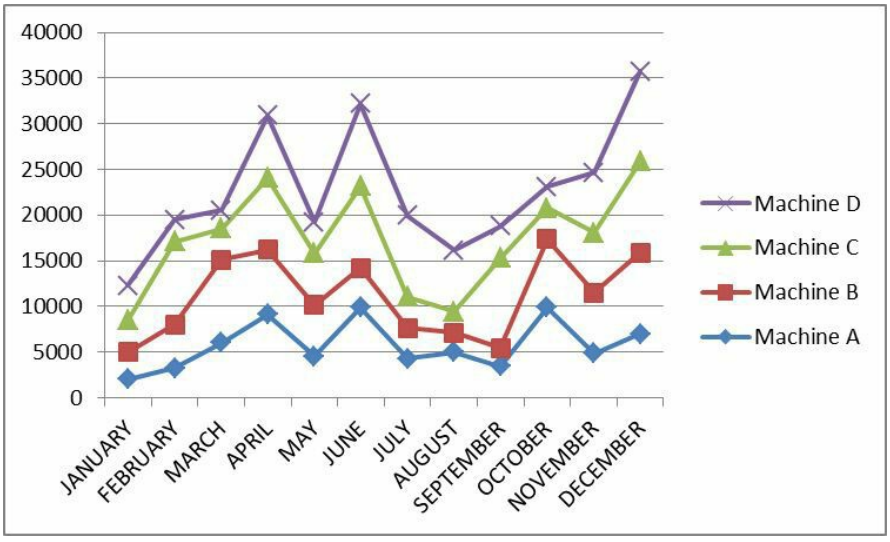
- Formation of line plots
- Scattering of plots
- Creations of beautiful bar charts and building up of histograms
- Application of various pie charts within the industry
- Stemming the schemes for data analysis and computations
- Being able to follow up contours plots
- Usage of spectrograms

- Quiver plots creation

Seaborn

Seaborn is also among the popular libraries within the Python category. Its main objective here is to help in visualization. It is important to note that this library borrows its foundation from Matplotlib. Due to its higher level, it is capable of various plots generation such as the production of heat maps, processing of violin plots and also helping in generation of time series plots.

Diagrammatic illustration



The above line graph clearly shows the performance of different machines the company is using. Following the diagram above, you can eventually deduce and make a conclusion on which machines the company can keep using to get the maximum yield. On most occasions, this evaluation method by the help of the Seaborn library will enable you to predict the exact abilities of your different inputs. Again, this information can help for future reference in the case of purchasing more machines. Seaborn library also has the power to detect the performance of other variable inputs within the company. For example, the number of workers within the company can be easily identified with their corresponding working rate.

NumPy

This is a very widely used Python library. Its features enable it to perform multidimensional array processing. Also, it helps in the matrix processing. However, these are only possible with the help of an extensive collection of

mathematical functions. It is important to note that this Python library is highly useful in solving the most significant computations within the scientific sector. Again, NumPy is also applicable in areas such as linear algebra, derivation of random number abilities used within industries and more so Fourier transformation. NumPy is also used by other high-end Python libraries such as TensorFlow for Tensors manipulation. In short, NumPy is mainly for calculations and data storage. You can also export or load data to Python since it has those features that enable it to perform these functions. It is also good to note that this Python library is also known as numerical Python.

SciPy

This is among the most popular library used in our industries today. It boasts of comprising of different modules that are applicable in the optimization sector of data analysis. It also plays a significant role in integration, linear algebra, and other forms of mathematical statistics.

In many cases, it plays a vital role in image manipulation. Manipulation of the image is a process that is widely applicable in day to day activities. Cases of Photoshops and much more are examples of SciPy. Again, many organizations prefer SciPy in their image manipulation, especially the pictures used for presentation. For instance, wildlife society can come up with the description of a cat and then manipulate it using different colors to suit their project. Below is an example that can help you understand this more straightforwardly. The picture has been manipulated:

The original input image was a cat that the wildlife society took. After manipulation and resizing the image according to our preferences, we get a tinted image of a cat.

Keras

This is also part and parcel of Python library, especially within Machine Learning. It belongs to the group of networks with high level neural. It is significant to note that Keras has the capability of working over other libraries, especially TensorFlow and even Theano. Also, it can operate nonstop without mechanical failure. In addition to this, it seems to work better on both the GPU and CPU. For most beginners in Python programming, Keras offers a secure pathway towards their ultimate understanding. They will be in a position to design the network and even to build it. Its ability to prototype faster and more quickly makes it the best Python library among the learners.

PyTorch

This is another accessible, but open-source kind of Python library. As a result of its name, it boasts of having extensive choices when it comes to tools. It is also applicable in areas where we have computer vision. Computer vision and visual display, play an essential role in several types of research. Again, it aids in the processing of Natural Language. More so, PyTorch can undertake some technical tasks that are for developers. That's enormous calculations and data analysis using computations. It can also help in graph creation which mainly used for computational purposes. Since it is an open-source Python library, it can work or perform tasks on other libraries such as Tensors. In combination with Tensors GPU, its acceleration will increase.

Scrapy

Scrapy is another library used for creating crawling programs. That's spider bots and much more. The spider bots frequently help in data retrieval purposes and also applicable in the formation of URLs used on the web. From the beginning, it was to assist in data scrapping. However, this has undergone several evolutions and led to the expansions of its general purpose. Therefore, the main task of the scrappy library in our present-day is to act as crawlers for general use. The library led to the promotion of general usage, application of universal codes, and so on.

Statsmodels

Statsmodels is a library with the aim of data exploration using several methods of statistical computations and data assertions. It has many features such as result statistics and even characteristic features. It can undertake this role with the help of the various models such as linear regression, multiple estimators, and analysis involving time series, and even using more linear models. Also, other models, such as discrete choice are applicable here.

What is the TensorFlow Library

The next thing that we need to spend some time looking at is the TensorFlow Library. This is another option that comes from Python, and it can help you to get some Machine Learning done. This one takes on a few different options of what you can do when it comes to Machine Learning, so it is definitely worth your time to learn how to use this option along with the algorithms that we talked about with the Scikit-Learn library.

TensorFlow is another framework that you can work within Python Machine Learning, and it is going to offer the programmer a few different features and tools to get your project done compared to the others. You will find that the framework that comes with TensorFlow is going to come from Google, and it is helpful when you are trying to work on some models that are deep learning related. TensorFlow is going to rely on graphs of data flow for numerical computation. And it can make sure that some of the different things that you can do with Machine Learning are easier than ever before.

TensorFlow is going to help us out in many different ways. First, it can help us with acquiring the data, training the models of Machine Learning that we are trying to use, helps to make predictions, and can even modify a few of the future results that we have to make them work more efficiently. Since each of these steps is going to be important when it comes to doing some Machine Learning, we can see how TensorFlow can come into our project and ensure we reach that completion that we want even better.

First, let’s take a look at what TensorFlow is all about and some of the background that comes with this Python library. The Brain team from Google was the first to develop TensorFlow to use on large scale options of Machine Learning. It was developed in order to bring together different algorithms for both deep learning and Machine Learning, and it is going to make them more useful through what is known as a common metaphor. TensorFlow works along with the Python language that we talked about before. In addition to this, it is going to provide the users with a front-end API that is easy to use when working on a variety of building applications.

It makes it a bit further, though. Even though you can work with TensorFlow and it matches up with the Python coding language while you do the coding and the algorithms, it is going to be able to change these up. All of the applications that you use with the help of TensorFlow are going to be

executed using the C++ language instead, giving them an even higher level of performance than before.

TensorFlow can be used for a lot of different actions that you would need to do to make a Machine Learning project a success. Some of the things that you can do with this library, in particular, will include running, training, and building up the deep neural networks, doing some image recognition, working with recurrent neural networks, digit classification, natural language processing, and even word embedding. And this is just a few of the things that are available for a programmer to do when they work with TensorFlow with Machine Learning.

Installing TensorFlow

With this in mind, we need to take some time to learn how to install TensorFlow on a computer before we can use this library. Just like we did with Scikit-Learn, we need to go through and set up the environment and everything else so that this library is going to work. You will enjoy that with this kind of library; it is already going to be set up with a few APIs for programming (we will take a look at these in more depth later on), including Rust, Go, C++ and Java to name a few. We are going to spend our time here looking at the way that the TensorFlow library is going to work on the Windows system, but the steps that you have to use to add this library to your other operating systems are going to be pretty much the same.

Now, when you are ready to set up and download the TensorFlow library on your Windows computer, you will be able to go through two choices on how to download this particular library. You can choose either to work with the Anaconda program to get it done, or a pip is going to work well, too. The native pip is helpful because it takes all of the parts that go with the TensorFlow library and will make sure that it is installed on your system. And you get the bonus of the system doing this for you without needing to have a virtual environment set up to get it done.

However, this one may seem like the best choice, but it can come with some problems along the way. Installing the TensorFlow library using a pip can be a bit faster and doesn’t require that virtual environment, but it can come with some interference to the other things that you are doing with Python. Depending on what you plan to do with Python, this can be a problem so consider that before starting.

The good thing to remember here is that if you do choose to work with a pip and it doesn't seem like it is going to interfere with what you are doing too much, you will be able to get the whole TensorFlow library to run with just one single command. And once you are done with this command, the whole library, and all of the parts that you need with it, are going to be set up and ready to use on the computer with just one command. And the pip even makes it easier for you to choose the directory that you would like to use to store the TensorFlow library for easier use.

In addition to using the pip to help download and install the TensorFlow library, it is also possible for you to use the Anaconda program. This one is going to take a few more commands to get started, but it does prevent any interference from happening with the Python program, and it allows you to create a virtual environment that you can work with and test out without a ton of interference or other issues with what is on your computer.

Though there are a few benefits to using the Anaconda program instead of a pip, it is often recommended that you install this program right along with a pip, rather than working with just the conda install. With this in mind, we will still show you some of the steps that it takes to just use the conda install on its own so you can do this if you choose.

One more thing that we need to consider here before moving on is that you need to double-check which version of Python is working. Your version needs to be at Python 3.5 or higher for this to work for you. Python 3 uses the pip 3 program, and it is the best and most compatible when it comes to working with a TensorFlow install. Working with an older version is not going to work as well with this library and can cause some issues when you try to do some of your Machine Learning code.

You can work with either the CPU or the GPU version of this library based on what you are the most comfortable with. The first code below is the CPU version and the second code below is going to be the GPU version.

```
pip 3 install – upgrade tensorflow
pip 3 install – upgrade tensorflow-gpu
```

Both of these commands are going to be helpful because they are going to ensure that the TensorFlow library is going to be installed on your Windows system. But another option that you can use is with the Anaconda package itself. The methods above were still working with the pip installs, but we talked about how there are a few drawbacks when it comes to this one.

Pip is the program that is already installed automatically when you install Python onto your system as well. But you may find out quickly that Anaconda is not. This means that if you want to ensure that you can get TensorFlow to install with this, then you need to first install the Anaconda program. To do this, just go to the website for Anaconda and then follow the instructions that come up to help you get it done.

Once you have had the time to install the Anaconda program, then you will notice that within the files there is going to be a package that is known as conda. This is a good package to explore a bit at this time because it is going to be the part that helps you manage the installation packages, and it is helpful when it is time to manage the virtual environment. To help you get the access that you need with this package, you can just start up Anaconda and it will be there.

When Anaconda is open, you can go to the main screen on Windows, click the Start button, and then choose All programs from here. You need to go through and expand things out to look inside of Anaconda at the files that are there. You can then click on the prompt that is there for Anaconda and then get that to launch on your screen. If you wish to, it is possible to see the details of this package by opening the command line and writing in "conda info." This allows you to see some more of the details that you need about the package and the package manager.

The virtual environment that we talk about with the Anaconda program is going to be pretty simple to use, and it is pretty much just an isolated copy of Python. It will come with all of the capabilities that you need to maintain all of the files that you use, along with the directories and the paths that go with it too. This is going to be helpful because it allows you to do all of your coding inside the Python program, and allows you to add in some different libraries that are associated with Python if you choose.

These virtual environments may take a bit of time to adjust to and get used to, but they are good for working on Machine Learning because they allow you to isolate a project, and can help you to do some coding, without all of the potential problems that come with dependencies and version requirements. Everything you do in the virtual environment is going to be on its own, so you can experiment and see what works and what doesn't, without messing up other parts of the code.

From here, our goal is to take the Anaconda program and get it to work on

creating the virtual environment that we want so that the package from TensorFlow is going to work properly. The conda command is going to come into play here again to make this happen. Since we are going through the steps that are needed to create a brand new environment now, we will need to name it tensorenviron, and then the rest of the syntax to help us get this new environment created includes:

```
conda create -n tensorenvrion
```

After you type this code into the compiler, the program is going to stop and ask you whether you want to create the new environment, or if you would rather cancel the work that you are currently doing. This is where we are going to type in the "y" key and then hit enter so that the environment is created. The installation may take a few minutes as the compiler completes the environment for you.

Once the new environment is created, you have to go through the process of actually activating it. Without this activation in place, you will not have the environment ready to go for you. You just need to use the command of "activate" to start and then list out the name of any environment that you want to work with to activate. Since we used the name of tensorenviron earlier, you will want to use this in your code as well. An example of how this is going to look includes:

Activate tensorenviron

Now that you have been able to activate the TensorFlow environment, it is time to go ahead and make sure that the package for TensorFlow is going to be installed too. You can do this by using the command below:

```
conda install tensorflow
```

When you get to this point, you will be presented with a list of all the packages that are available to install in case you want to add in a few others along with TensorFlow. You can then decide if you want to install one or more of these packages, or if you want to just stick with TensorFlow for right now. Make sure to agree that you want to do this and continue through the process.

The installation of this library is going to get to work right away. But it is going to be a process that takes some time, so just let it go without trying to backspace or restart. The speed of your internet is going to make a big determinant of whether you will see this take a long time or not.

Soon though, the installation process for this library is going to be all done, and you can then go through and see if this installation process was successful or if you need to fix some things. The good news is the checking phase is going to be easy to work with because you can just use the import statement of Python to set it up.

This statement that we are writing is then going to go through the regular terminal that we have with Python. If you are still working here, like you should, with the prompt from Anaconda, then you would be able to hit enter after typing in the word Python. This will make sure that you are inside the terminal that you need for Python so you can get started. Once you are in the right terminal for this, type in the code below to help us get this done and make sure that TensorFlow is imported and ready to go:

```
import tensorflow as tf
```

At this point, the program should be on your computer and ready to go and we can move on to the rest of the guidebook and see some of the neat things that you can do with this library. There may be a chance that the TensorFlow package didn't end up going through the way that it should. If this is true for you, then the compiler is going to present you with an error message for you to read through and you need to go back and make sure the code has been written in the right format along the way.

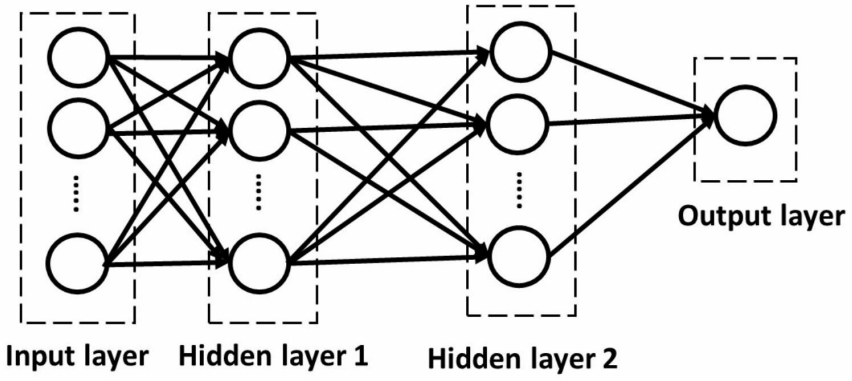
The good news is if you finish doing this line of code above and you don't get an error message at all, then this means that you have set up the TensorFlow package the right way and it is ready to use! With that said, we need to explore some more options and algorithms that a programmer can do when it comes to using the TensorFlow library and getting to learn how they work with the different Machine Learning projects you want to implement.

Artificial Neural Networks

This chapter discusses the integral aspect of artificial neural networks. It also covers their component in particular activation functions and how to train an artificial neural network, as well as the different advantages of using an artificial neural network.

Definition of artificial neural network

The employment of artificial neural networks is a widely used approach in Machine Learning. It is inspired by the brain system of humans. The objective of neural networks is to replicate how the human brain learns. The neural network system is an ensemble of input and output layers and a hidden layer that transforms the input layer into useful information to the output layer. Usually, several hidden layers are implemented in an artificial neural network. The figure below presents an example of a neural network system composed of 2 hidden layers:



Example of an artificial neural network

Before going further and explaining how neural networks work, let's first define what a neuron is. A neuron is simply a mathematical equation expressed as the sum of the weighted inputs. Let's consider $X = \{x_1, x_2, \ldots, x_M\}$ a vector of M inputs, the neuron is a linear combination of all inputs defined as follows:

$$F(X = \{x_1, x_2, \ldots, x_M\}) = w_1x_1 + w_2x_2 + \ \ldots + w_Mx_M,$$

being $w_1, w_2, \ldots, w_M$ the weights assigned to each input. The function F can also be represented as:

$$F(X) = WX$$

Where W is a weight matrix and X a vector of data. The second formulation is very convenient when programming a neural network model. The weights are determined during the training procedure. Training an artificial neural network means finding the optimal weights W that provide the most accurate output.

To each neuron, an activation function is applied the resulted weighted sum of inputs X. The role of the activation function is deciding whether the neuron should be activated or not according to the model's prediction. This process is applied to each layer of the network. In the next sub-sections, we will discuss in detail the role and types of activation functions as well as the different types of neural networks.

What is an activation function and its role in neural network models?

Activation functions are formulated as mathematical functions. These functions are a crucial component of an artificial neural network model. For each neuron, an activation function is associated. The activation function decides whether to activate the neuron or not. For instance, let's consider the output from a neuron, which is:

$$Y = \sum \text{weight} \times \text{input} + \text{bias}$$

The output Y can be of any value. The neuron does not have any information on the reasonable range of values that Y can take. For this purpose, the activation function is implemented in the neural network to check Y values and make a decision on whether the neural connections should consider this neuron activated or not.

There are different types of activation functions. The most instinctive function is the step function. This function sets a threshold and decides to activate or not activate a neuron if it exceeds a certain threshold. In other words, the output of this function is 1 if Y is greater than a threshold and 0 otherwise. Formally, the activation function is:

$$F = \begin{cases} 1, & \text{if } Y > \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

where 1 means 'activated' and 0 means 'not-activated'.

This activation function can be used for a classification problem where the output should be yes or no (i.e., 1 or 0). However, it has some drawbacks. For example, let's consider a set of several categories (i.e., class1, class2, …, etc.) to which input may belong to. If this activation function is used and more than one neuron is activated, the output will be 1 for all neurons. In this case, it is hard to distinguish between the classes and decide into which class the input belongs to because all neuron outputs are 1. In short, the step function does not support multiple output values and classification into several classes.

Linear activation function, unlike the step function, provides a range of activation values. It computes an output that is proportional to the input. Formally:

$$F(X) = WX,$$

where X is the input.

This function supports several outputs rather than just 1 or 0 values. This function, because it is linear, does not support backpropagation for model training. Backpropagation is the process that relies on function derivative or gradient to update the parameters, in particular, the weights. The derivative (i.e., gradient) of the linear activation function is a constant which is equal to W and is not related to changes in the input X. Therefore, it does not provide information on which weights applied to the input can give accurate predictions.

Moreover, all layers can be reduced to one layer when using the linear function. The fact that all layers are using a linear function, the final layer is a linear function of the first layer. So, no matter how many layers are used in the neural network, they are equivalent to the first layer, and there is no point in using multiple layers. A neural network with multiple layers connected with a linear activation function is just a linear regression model that cannot support the complexity of input data.

The majority of neural networks use non-linear activation functions because, in the majority of real-world applications, relations between the output and the input features are non-linear. The non-linear functions allow the neural network to map complex patterns between the inputs and the outputs. They also allow the neural network to learn the complex process that governs complex data or high dimension data such as images, audios, among others.

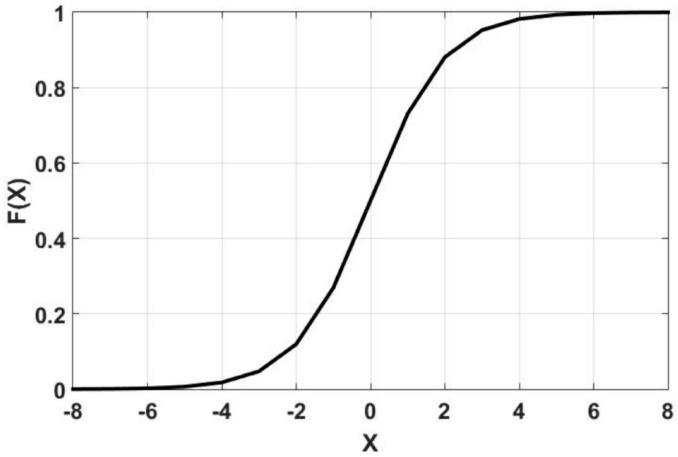
The non-linear functions allow overcoming the drawbacks of linear functions and step functions. They support backpropagation (i.e., the derivative is not a constant and depends on the changes of the input) and stacking several layers (i.e., the combination of non-linear functions is non-linear). Several non-linear functions exist and can be used within a neural network. In this book, we are going to cover the most commonly used non-linear activation functions in Machine Learning applications.

The sigmoid function

The sigmoid function is one of the most used activation functions within an artificial neural network. Formally, a sigmoid function is equal to the inverse of the sum of 1 and the exponential of inputs:

$$F(X) = \frac{1}{1 + \exp(-X)}$$

Outputs of a sigmoid function are bounded by 0 and 1. More precisely, the outputs take any value between 0 and 1 and provide clear predictions. In fact, when the X is greater than 2 or lower than -2, the value of Y is close to the edge of the curve (i.e., closer to 0 or 1).



The disadvantage of this activation function, as we can see from the figure above, is the small change in the output for input values under -4 and above 4. This problem is called ‘vanishing gradient’ which means that the gradient is very small on horizontal extremes of the curve. This makes a neural network using the sigmoid function, learning very slowly when they approach the edges and computationally expensive.

The tanh function

The tanh function is another activation function used that is similar to the

sigmoid function. The mathematical formulation of this function is:

$$F(X) = \tanh(X) = \frac{2}{1 + \exp(-2X)} - 1$$

This function is a scaled sigmoid function. Therefore, it has the same characteristics as the sigmoid function. However, the outputs of this function range between -1 and 1, and the gradient are more pronounced than the gradient of the sigmoid function. Unlike the sigmoid function, the tanh function is zero-centered, which makes it very useful for inputs with negative, neutral, and positive values. The drawback of this function, as for the sigmoid function, is the vanishing gradient issue and computationally expensive.

The ReLu function

The Rectified Linear Unit function or what is known as ReLu function, is also a widely used activation function, which is computationally efficient. This function is efficient and allows the neural network to converge quickly compared to the sigmoid and tanh function because it uses simple mathematical formulations. ReLu returns X as output if X is positive or 0 otherwise. Formally, this activation function is formulated as

$$F(X) = \max(0, X)$$

This activation function is not bounded and takes values from 0 to +inf. Although it has a similar shape as a linear function (i.e., this function is equal to identity for positive values), the ReLu function has a derivative. The drawback of the ReLu is that the derivative (i.e., the gradient) is 0 when the inputs are negative. This means as for the linear functions, the backpropagation cannot be processed, and the neural network cannot learn unless the inputs are greater than 0. This aspect of the ReLu, gradient equal to 0 when the inputs are negative, is called the dying ReLu problem.

To prevent the dying ReLu problem, two ReLu variations can be used, namely the Leaky ReLu function and the Parametric ReLu function. The Leakey ReLu function returns as output the maximum of X and X by 0.1. In other words, the leaky ReLu is equal to the identity function when X is greater than 0 and is equal to the product of 0.1 and X when X is less than zero. This function is provided as follows:

$$F(X) = \max(0.1X, X)$$

This function has a small positive gradient which is 0.1 when X has negative

values, which make this function support backpropagation for negative values. However, it may not provide a consistent prediction for these negative values.

The parametric ReLu function is similar to the Leaky ReLu function, which takes the gradient as a parameter to the neural network to define the output when X is negative. The mathematical formulation of this function is as follows:

$F(X) = \max(aX, X)$

There are other variations of the ReLu function such as the exponential linear ReLu. This function, unlike the other variations of the ReLu the Leaky ReLu and parametric ReLu, has a log curve for negative values of X instead of the linear curves like the Leaky ReLu and the parametric ReLu functions. The downside of this function is it saturates for large negative values of X. Other variations exist which all rely on the same concept of defining a gradient greater than 0 when X has negative values.

The Softmax function

The Softmax function is another type of activation function used differently than the one presented previously. This function is usually applied only to the output layer when a classification of the inputs into several different classes is needed. In fact, the Softmax function supports several classes and provides the probability of input to belong to a specific class. It normalizes outputs of every category between 0 and 1 then divides by their sum to provide that probability.

Given all these activation functions, where each one has its pros and cons, the question now is: which one should be used in a neural network? The answer is: simply having a better understanding of the problem in hand will help guide into a specific activation function, especially if the characteristics of the function being approximated are known beforehand. For instance, a sigmoid function is a good choice for a classification problem. In case the nature of the function being approximated is unknown, it is highly recommended to start with a ReLu function rather than trying other activation functions. Overall, the ReLu function works well for a wide range of applications. It is an ongoing research, and you may try your activation function.

An important aspect of choosing an activation function is the sparsity of the activation. Sparsity means that not all neurons are activated. This is a desired

characteristic in a neural network because it makes the network learns faster and less prone to overfitting. Let's imagine a large neural network with multiple neurons if all neurons were activated; it means all these neurons are processed to describe the final output. This makes the neural network very dense and computationally exhaustive to process. The sigmoid and the tanh activation functions have this property of activating almost all neurons, which makes them computationally inefficient unlike the ReLu function and its variations that cause the inactivation of some negative values. That is the reason why it is recommended to start with the ReLu function when approximating a function with unknown characteristics.

What are the types of artificial neural networks?

Several categories of artificial neural networks with different properties and complexities exist. The first and simplest neural network developed is the perceptron. The perceptron computes the sum of the inputs, applies an activation function, and provides the result to the output layer.

Another old and simple approach is the feedforward neural network. This type of artificial neural network has only one single layer. It is a category that is fully connected to the following layer where each node is attached to the others. It propagates the information in one direction from the inputs to the outputs through the hidden layer. This process is known as the front propagated wave that usually uses what is called the activation function. This activation function processes the data in each node of the layers. This neural network returns a sum of weights by the inputs calculated according to the hidden layer's activation function. The category of feedforward neural network usually uses the backpropagation method for the training process and the logistic function as an activation function.

Several other neural networks are a derivation of this type of network. For example, the radial-basis-function *neural* networks. This is a feedforward neural network that depends on the radial basis function instead of the logistic function. This type of neural networks have two layers, wherein the inner layer, the features, and radial basis function are combined. The radial function computes the distance of each point to the relative center. This neural network is useful for continuous values to evaluate the distance from the target value.

In contrast, the logistic function is used for mapping arbitrary binary values

(i.e., 0 or 1; yes or no). Deep feedforward neural networks are a multilayer feedforward neural network. They became the most commonly used neural network types used in Machine Learning as they yield better results. A new dydydytype of learning called deep learning has emerged from these types of neural networks.

Recurrent neural networks are another category that uses a different type of nodes. Like a feedforward neural network, each hidden layer processes the information to the next layer. However, outputs of the hidden layers are saved and processed back to the previous layer. The first layer, comprised of the input layer, is processed as the product of the sum of the weighted features. The recurrent process is applied in hidden layers. At each step, every node will save information from the previous step. It uses memory, while the computation is running. In short, the recurrent neural network uses forward propagation and backpropagation to self-learn from the previous time steps to improve the predictions. In other words, information is processed in two directions, unlike the feedforward neural networks.

A multilayer perceptron, or multilayer neural network, is a neural network that has at least three or more layers. This category of networks is fully connected where every node is attached to all other nodes in the following layers.

Convolutional neural networks are typically useful for image classification or recognition. The processing used by this type of artificial neural network is designed to deal with pixel data. The convolutional neural networks are a multi-layer network that is based on convolutions, which apply filters for neuron activation. When the same filter is applied to a neuron, it leads to an activation of the same feature and results in what is called a feature map. The feature map reflects the strength and importance of a feature of input data.

Modular neural networks are formed from more than one connected neural network. These networks rely on the concept of ‘divide and conquer.’ They are handy for very complex problems because they allow combining different types of neural networks. Therefore, they allow combining the strengths of a different neural network to solve a complex problem where each neural network can handle a specific task.

How to train an artificial neural network?

As explained at the beginning of this chapter, neural networks compute a

weighted sum of inputs and apply an activation function at each layer. Then it provides the final result to the output layer. This procedure is commonly named *forward propagation*. To train these artificial neural networks, weights need to be optimized to obtain the optimal weights that produce the most accurate outputs. The process of the training an artificial neural network is as follows:

1. Initialize the weights
2. Apply the forward propagation process
3. Evaluate the neural network performance
4. Apply the backward propagation process
5. Update the weights
6. Repeat the steps from step 2 until it attains a maximum number of iterations, or neural network performance does not improve.

As we can see from the steps of training an artificial neural network presented above, we need a performance measure that describes how accurate the neural network is. This function is called the loss function or cost function. This function can be the same as the cost function we presented in the previous chapter:

$$J = \frac{1}{N}\sum(y_{predicted} - y_{target})^2$$

Where N is the number of outputs, $y_{predicted}$ is the output and y_{target} is the true value of the output. This function provides the error of the neural network. Small values of J reflect the high accuracy of the neural network.

So far, we defined loss function and how the neural network works in general. Now, let's go into the details for each step of the training process.

Let's consider a set of inputs X and outputs Y. We initialize W (i.e., weights) and B (i.e., bias) as a null matrix. The next step is to apply the feed-forward propagation that consists of feeding each layer of the artificial neural network with the sum of the weights by the inputs and the bias. Let's consider that we have two layers. We can calculate the first hidden layer's output using the following equation:

$$Z_1 = W_1 X + b_1$$

Where W1 and b1 are the parameters of the neural network as the weights and bias of the first layer, respectively.

Next, we apply the activation function F1, that can be any activation from the function presented previously in this chapter:

$$A_1 = F_1(Z_1)$$

The result is the output of the first layer, which is then fed to the next layer as:

$$Z_2 = W_2A_1 + b_2$$

With W2 and b2 are the weights and bias of the second layer, respectively.

To this result, we apply an activation function F2:

$$A_2 = F_2(Z_2)$$

Now A2 is supposed to be the output of the artificial neural network. The activation function F1 and F2 might be the same activation function or different activation function depending on the dataset and the expected output.

After the feedforward propagation, we compare the neural network output against the target output with the loss function. It is highly likely the difference between the estimated output and the actual values at this stage is very high. Therefore, we have to adjust the weights through the backpropagation process. We calculate the gradient of each activation function concerning biases and weights. We start by evaluating the derivative of the last layer, then the layer before this layer on so on until the input layer. Then update the weights according to the gradient or the derivative of the activation function. Applying these steps to our example of two layers neural network it provides:

$$W_2 = W_2 - \alpha\frac{d}{dW}F_2(W,b)$$

$$b_2 = b_2 - \alpha\frac{d}{db}F_2(W,b)$$

$$W_1 = W_1 - \alpha\frac{d}{dW}F_2(W,b)$$

$$b_1 = b_1 - \alpha \frac{d}{db} F_2(W,b)$$

The parameter α is the learning rate parameter. This parameter determines the rate by which the weights are updated. The process that we just describe here is called the gradient descent algorithm. The process is repeated until it attains a pre-fixed maximum number of iterations. In chapter 4, we will develop an example to illustrate a perceptron and multi-layer neural network by following similar steps using Python. We will develop a classifier based on an artificial neural network. Now, let’s explore the pros of using an artificial neural network for Machine Learning applications.

Artificial neural network: pros and cons of use

Nowadays, artificial neural networks are applied in almost every domain. Research in the domain of artificial neural networks is very active, and several neural networks immerged to take advantage of the full potential of this Artificial intelligence approach. Artificial neural networks have several advantages.

Artificial neural networks are able to map structures and learn from the data faster. They are also able to map the complex structure and connections that relate the outputs to the input datasets, which is the case in many real-life applications. Once an artificial neural network is developed and trained, it can be generalized. In other words, it can be applied to map relationships between data that it has not been exposed to or to make predictions for new datasets. Moreover, the artificial neural network does not make any assumptions of the structure or the distribution of the input data. It does not impose specific conditions on the data or assumptions on the relationship in the data, unlike traditional statistical methods. The fact that artificial neural networks can handle a large amount of data makes them an appealing tool. Artificial neural networks are a non-parametric approach that allows developing a model with a reduced error that is caused by the estimation of the parameters. Although these appealing characteristics of artificial neural networks, they suffer from some drawbacks.

The downside of artificial neural networks is that they often operate as a black box. This means that we cannot fully understand the relationship between the inputs and outputs and the interdependence between specific input variables and the output. In other words, we cannot detect how much

each input variable impacts the output. The training process can be computationally inefficient. We can overcome this problem by using parallel computing and taking advantage of the computation power of computers by using proper coding.

Conclusion

Thanks for reading to the end!

Python Machine Learning may be the answer that you are looking for when it comes to all of these needs and more. It is a simple process that can teach your machine how to learn on its own, similar to what the human mind can do, but much faster and more efficient. It has been a game-changer in many industries, and this guidebook tried to show you the exact steps that you can take to make this happen.

There is just so much that a programmer can do when it comes to using Machine Learning in their coding, and when you add it together with the Python coding language, you can take it even further, even as a beginner.

The next step is to start putting some of the knowledge that we discussed in this guidebook to good use. There are a lot of great things that you can do when it comes to Machine Learning, and when we can combine it with the Python language, there is nothing that we can't do when it comes to training our machine or our computer.

This guidebook took some time to explore a lot of the different things that you can do when it comes to Python Machine Learning. We looked at what Machine Learning is all about, how to work with it, and even a crash course on using the Python language for the first time. Once that was done, we moved right into combining the two of these to work with a variety of Python libraries to get the work done.

If you have ever wanted to learn how to work with the Python coding language, or you want to see what Machine Learning can do for you, then this guidebook is the ultimate tool that you need! Take a chance to read through it and see just how powerful Python Machine Learning can be for you.