

PYTHON

MACHINE LEARNING

THE ULTIMATE BEGINNER'S GUIDE TO LEARN
PYTHON MACHINE LEARNING STEP-BY-STEP
USING SCIKIT-LEARN AND TENSORFLOW



MARK REED

Python Machine Learning

*The Ultimate Beginner's Guide to Learn Python Machine Learning
Step by Step using Scikit-Learn and Tensorflow*

Mark Reed

© Copyright 2020 - All rights reserved.

The content contained within this book may not be reproduced, duplicated or transmitted without direct written permission from the author or the publisher.

Under no circumstances will any blame or legal responsibility be held against the publisher, or author, for any damages, reparation, or monetary loss due to the information contained within this book, either directly or indirectly.

Legal Notice:

This book is copyright protected. It is only for personal use. You cannot amend, distribute, sell, use, quote or paraphrase any part, or the content within this book, without the consent of the author or publisher.

Disclaimer Notice:

Please note the information contained within this document is for educational and entertainment purposes only. All effort has been executed to present accurate, up to date, reliable, complete information. No warranties of any kind are declared or implied. Readers acknowledge that the author is not engaged in the rendering of legal, financial, medical or professional advice. The content within this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.

By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, that are incurred as a result of the use of the information contained within this document, including, but not limited to, errors, omissions, or inaccuracies.

Table of Contents

[Introduction](#)

[What is Machine Learning?](#)

[The Difference Between Machine Learning \(ML\) and Artificial Intelligence \(AI\)](#)

[Some Acronyms to be Aware of When Learning ML:](#)

[Classification of Machine Learning Algorithms](#)

[What is Scikit Learn?](#)

[What is Tensorflow?](#)

[Chapter 1: History of Machine Learning](#)

[Chapter 2: Approaches to Machine Learning](#)

[Machine Learning Terminology](#)

[The Machine Language Process](#)

[Chapter 3: Machine Learning Environment Setup](#)

[Setting Up Python and Anaconda](#)

[Installing Python](#)

[Setting Up the Python Environment](#)

[Installing Anaconda](#)

[Setting Up the Anaconda Environment](#)

[Installing Scikit-Learn](#)

[Installing TensorFlow](#)

[Chapter 4: Using Scikit-Learn](#)

[The Learning Problem](#)

[Loading Data Sets](#)

[Regression](#)

[Chapter 5: K-Nearest Neighbors \(KNN\) Algorithm](#)

[How to Determine the “k” Parameter](#)

[How to Choose the Value of k?](#)

[When to Use KNN Models?](#)

[How the KNN Algorithm Works](#)

[The Workings of the KNN Algorithm](#)

[Implementing KNN Algorithm](#)

[Chapter 6: Using TensorFlow](#)

[Getting Started with TensorFlow](#)

[Working with TensorFlow Variables, Constants, Strings, Updates, Sessions, Placeholders, and Arrays](#)

[Working TensorFlow Data Flow Graph and Programming Structure](#)

[Chapter 7: Machine Learning and Neural Networks](#)

[Neural Networks](#)

[Chapter 8: Machine Learning and Big Data](#)

[The 5 V's of Big Data](#)

[Big Data Uses](#)

[Chapter 9: Machine Learning Classification and Regression](#)

[Chapter 10: Machine Learning and the Cloud](#)

[Benefits of Cloud-Based Machine Learning](#)

[Chapter 11: Machine Learning and the Internet of Things \(IoT\)](#)

[Uses for the Internet of Things](#)

[IoT Security Concerns](#)

[Chapter 12: Machine Learning and Robotics](#)

[Robotic Learning](#)

[Examples of Industrial Robots and Machine Learning](#)

[Neural Networks with Scikit-learn](#)

[Chapter 13: Machine Learning Models](#)

[Machine Learning and Swarm Intelligence](#)

[Chapter 14: Applications of Machine Learning](#)

[Chapter 15: Limitation of Machine Learning](#)

[Problematic Limitation of Machine Learning](#)

[Philosophical Objections and Limitation to Machine Learning](#)

[Chapter 16: Machine Learning and the Future](#)

[Conclusion](#)

[References](#)

[Images](#)

Introduction

If you have worked as a programmer in any other language, you will quickly be able to grasp the concept of Python. As with any type of new knowledge, it will come with a learning curve. However, Python is not a difficult language to learn and it doesn't come with a steep learning curve. This makes Python an ideal language for beginner programmers just starting out.

It is also open-source software, which means it is free and easily accessible for anyone to download. Over the last several years Python has become very popular among the programming community. It has a large and growing list of libraries and open source packages readily available for download across the Internet.

Python is also one of the preferred programming languages for machine learning (ML) and is fast becoming one of the top programming languages for teaching ML. It is a powerful object-oriented, interactive, and interpreted programming language that utilizes concise syntax that is clear and easy to learn and follow. Python has many exceptions, high-level dynamic data types, dynamic typing, modules, and classes. It can also interface easily with other programming languages including C++, C, and Java.

Python is almost ten to fifteen times faster than languages such as C++, C, and JavaScript. Because of the object-orientated nature of Python programming, it makes translation a smooth and straightforward process. It can also be used to create easy automation interfaces and scripting for programming languages as an extension language.

When it comes to machine learning, Python is considered to be one of the preferred languages as it:

- Uses clear and concise syntax
- Has a lot of code libraries
- Can interface with nearly all the available platforms and languages
- Is Open Source and readily available to download from the Internet

What is Machine Learning?

In today's technology-driven world there are so many different disciplines that pop up, it's easy to get confused. For example, a lot of people tend to think machine learning (ML) and artificial intelligence (AI) are the same things. Although ML does get used together with AI, they are not the same thing. However, ML is a subset of AI.

The Difference Between Machine Learning (ML) and Artificial Intelligence (AI)

Artificial intelligence and machine learning are sometimes hard to tell apart, especially as more and more systems now integrate ML with AI systems.

Artificial Intelligence (AI)

Any system such as a phone, a robot, car, refrigerator, or a computer program that is made to be smart is AI.

Some examples of AI systems include:

- Siri
- Alexa
- Video Games
- Smart Phones
- Smart Cars
- Smart Home devices

Machine Learning (ML)

Machine learning systems are systems that start with basic knowledge but get smarter and smarter over time. These machines are designed to learn by themselves and can gain knowledge to get smarter without human intervention. More and more AI systems now integrate ML. They do so to create smarter AI systems that can learn by themselves without having to constantly physically update them.

Some examples of ML systems include:

- Speech recognition systems
- Clinical diagnostic systems
- Financial and banking services
- Virtual personal assistants
- Online customer support applications

- Google and other search engines
- Online fraud detection systems
- Social media services
- Automated product recommendation systems.

Some Acronyms to be Aware of When Learning ML:

While learning ML you will come across quite a few acronyms in reference to various applications.

These acronyms include:

Artificial Neural Networks (ANN)

Although there have been models of this type of networking since the 1940s, they were not very efficient models. These types of networks work well with massively large data sets and in recent years have advanced to become extremely powerful and useful. ANN models are inspired by the brain's biological nervous system and the way it processes information. The most common systems they are used for include:

- Classification systems
- Self-driving cars
- Stock market prediction systems
- Character recognition systems

Automated Speech Recognition (ASR)

ASR is a program integrated into various hardware systems that is designed to recognize voice or speech patterns. It can identify and process a human's voice. This gives humans the ability to interface with a system by talking to it. ASR is limited in that it does not allow for open-ended conversation. Instead, ASR offers a menu of options a person can choose from to answer specific questions. The most common systems they are used for include:

- GPS software
- Phone answering messaging services
- Automated telephone banking
- Customer service interfaces

Deep Learning (DL)

Deep learning systems learn by gathering information from data sets. These are large data sets that continue to grow. DL is part of ML but is the part that deals with much larger data sets or sources. DL systems, like ML systems, get smarter as they learn and grow. Some examples of systems that use DL include:

- Voice-activated assistants
- Self-driving cars
- Voice-activated search engines or facilities
- Language translation services such as French into English
- Automatic text generators
- Prediction software
- Statistical software
- Survey software
- Spam filters
- Advertising suggestion software
- Plagiarism checkers

Natural Language Processing (NLP)

NLP is the more advanced version of ASR. NLP allows for interaction between a human and a device. ASR has a menu of words or options that a person has to choose from to get a response or be directed. NLP uses an open-ended chat format that is close to having a real-time conversation with the machine.

Some systems that use NLP that you may be familiar with include:

- Siri
- Alexa

Classification of Machine Learning Algorithms

An algorithm is used to solve a problem and generate a solution in a computer program. It is a sequence of instructions given to the system which makes it function in the way the programmer designs it to. For instance, something as simple as logging in to a system is causing a sequence of instructions to be carried out within the program. Based on various instructions, rules, and parameters, computer algorithms are designed to process and organize these instructions accordingly.

Machine learning is about computer algorithms that create a new set of rules. This means that instead of a programmer manually teaching the system by writing code after code of updated instructions, the computer has a set of instructions that enable it to learn from data input and grow, allowing for tasks that before could not be manually programmed to be carried out by the system. These are tasks such as a system recognizing and processing voices. Taking basic data sets and using them to learn and grow for systems that require predictive or suggestive outcomes is the basis of machine learning.

Machine learning can be categorized into three general methods or techniques:

Reinforcement Learning

Reinforcement learning is a rewards-based system of learning. The system is either penalized or rewarded for the task it carries out. The programmer may set the basic rules and rewards policies for the system but will not give the system any clues as to how to carry out or solve the task.

For instance, a self-driving car has to figure out how to get the passenger from A to B without harming the passer while keeping the car on the road and avoiding other cars. The car may have been fed a series of maps, general road rules, and information of objects it may encounter along the way. The car has to figure out the best way to perform its task and stick to rules to successfully complete the task it has been given. The car can leverage the power of being able to search through an ever-growing database of parallel scenarios to determine how to complete the task.

Supervised Learning

When children are taught, they are given examples of what they need to learn. Through these examples, children learn to make choices and base outcomes on these choices. For example, you can show a child a round ball and tell them it is a soccer ball. Then you can show an oblong eye-shaped ball and tell them it is a rugby ball. When they are asked to identify or retrieve a rugby ball, the child will pick the ball they were told and shown to be a rugby ball. This is what is called supervised learning. It is the process whereby a system's response to new data input is generated using a set of known data and outcomes to generate a reasonable predicted outcome.

For instance, let's say you want to establish what the average price of a house in a certain neighborhood might be. You would input what a 2 bedroom, 2 bathroom house on the block recently sold for. You would also input what a 3 bedroom, 2 bathroom house on the block recently sold for. Using that information you could find out what a 4 bedroom, 2 bathroom house on the block is estimated to sell for.

For numeric labels in supervised learning the model is referred to as a "classification." The model is known as a "regression" for categorical labels. Supervised learning algorithms include:

- For regression problems, linear/logistic regression are used
- For classification problems, support vectors are used
- Random forest can be used for both regression and classification problems

Unsupervised Learning

Unsupervised learning is where the system has input data but no example outcomes. That machine is given the basic data and must learn more about the data to establish an outcome.

For instance, an automated recommendation system is a prime example of an unsupervised learning system. This type of system uses a person's past viewing or searching history to make suggestions based on the choices the person made in the past. As the person searches more subjects or views more videos, the system will adjust the suggestions accordingly.

Unsupervised learning does not use labels and the algorithms are left to discover and learn on their own. Unsupervised algorithms can be separated into two groups:

- Association, which is where the algorithm bases the outcome on similar problems. For instance, a customer who buys toothpaste will most likely also want a toothbrush.
- Clustering is what groups customers into buyers of toothpaste.

What is Scikit Learn?

Scikit Learn is a Python library that contains tools such as regression, clustering, dimensionality reductions, and

classification for statistical modeling and machine learning. Scikit Learn tools are used for building ML models, unlike Pandas and NumPy which are used for data summarizing, reading, and manipulation.

Scikit learn, like Python, is a library that is free to download and use. It has features that include:

- K-Neighbors
- Random forests
- Support vector machines

What is Tensorflow?

Tensorflow is an open-source library that is free to download and use and was created by Google. It is used for the development of deep learning (DL) models and allows for quick and easy computations of complex numerical operations across multiple platforms. Programmers use it for neural networks that are more large-scale and usually have multiple layers. Its primary use is for discovering, creating, and predicting, as well as classification, understanding, and perception in ML models.

Python is used as a front-end application programming interface (API) for Tensorflow. Tensorflow is used to build, run, and train deep neural networks for systems which include:

- Word embedding
- Natural language processing (NLP)
- Handwritten digit classification
- Image recognition
- Recurrent neural networks.

Chapter 1:

History of Machine Learning

The history of machine learning is embedded in the history of computing and man's curiosity about artificially mimicking the brain. Most computing functions are made up of a series of complex algorithms. So, it should come as no surprise that computers were born from mathematics, in particular, Boolean logic.

In Babylon, which is modern day Iraq, counting tools which are known as abacuses were used as far back as 300 B.C. The Abacus is an important computing reference because it was one of the first counting machines in history. It was also a prelude to the first gear-driven calculating clock developed by Wilhelm Schickard in 1623. It was a calculating machine that could add and subtract up to six-digit numbers.

The dynamics of the workings of computers were figured out by mathematicians over a span of hundreds of years. It was through the expansion of various theories and prototypes developed by the minds of these great mathematicians from times past that computing developed to where it is today.

Computing History Timeline

In 1652, an 18-year-old young man, Blaise Pascal, became one of France's most revered mathematicians and was hailed a child prodigy. Pascal invented the first digital calculator to help his father calculate tax accounting. It was known as the Arithmetique or Pascaline. It was an arithmetic machine that could add, subtract, multiply, and divide.

In 1671 the Step Reckoner was designed by Gottfried Wilhelm Leibniz. It was a calculating machine that updated Pascal's idea and added a new dimension of multiplication to the machine. The machine was built in 1673. Although Leibniz was a staunch advocate of the binary system, his machine used the decimal system.

In 1689, Gottfried Wilhelm Leibniz created the binary number system that is still used in computing today. He developed the binary number system using only ones and zeros to convert logical verbal statements into mathematical ones. In 1703, he wrote an article on how zeros and ones could easily represent numbers (Gonzalez, 2018).

In 1801, a loom created by Joseph Marie Jacquard of France automatically wove fabric designs based on wooden punch cards inserted into the machine. The first computers used a design that was much the same as this loom.

In 1822, English inventor Charles Babbage was funded by the English government to design a steam-driven calculating machine. The machine was supposed to be able to calculate tables of numbers, but the project was a failure. The machine was known as the Difference Engine. Babbage did, however, find ways to try to make the calculator a more viable project, and by **1833** he had begun to develop a better machine, the Analytical Engine. It was the first "fully program controlled, automatic mechanical digital computer" ("Analytical Engine", n.d.). There were four components to the machine, which were the reader, the mill, the store, and the printer. Babbage died before completing the Analytical Engine, but it was a machine that was worthy of being called the first computer.

In 1842, Ada Lovelace, deemed the world's first computer programmer, wrote the first ever machine algorithm. The algorithm was written for Babbage's Analytical Engine, which at the time only existed on paper.

In 1847, George Boole started Boolean algebra which was first called the algebra of logic. Boolean logic is the logic upon which systems like telephone switching is based. It is also logic upon which computers operate and are designed.

In 1936, Alan Turing invented an idea of a "Universal Machine" that would be able to analyze and perform a given set of tasks. In 1930, the death of someone close to him made him become obsessed with the brain and the mind, as he believed his friend's mind did not just die with him. The paper that Turing published in 1936 has become known as the "Foundation of computer science" ("Alan Turing", n.d.) In **1946**, Turing designed the Automatic Computing Engine (ACE).

In 1943, a neurophysiologist, Warren McCulloch, teamed up with a mathematician, Walter Pitts, and co-wrote a paper theorizing how neurons in the human brain might function. They then modeled their theory by building a simple neural network with electrical circuits. Their paper gave a simplified model of a neural network with electrical circuits which has had an important impact on artificial neural networks.

In 1949, Donald Hebb created a model based on the brain cell interaction which he published in a book. The book, "The Organization of Behavior," discussed Hebb's theories on communication and excitement between neurons. The model which Hebb described in his book reveals a way that the relationship between two artificial neurons can be altered.

In 1950, Alan Turing, in one of his philosophical papers, put forward the "Learning Machine." His idea was to compare the outputs of a human against that of a machine. This paper was to become the paper that he would be remembered for and was what the Turing Test was adapted from. The Turing Test is still used today to determine the intelligence behavior of a machine. In order for an AI to be determined as intelligent, it must be able to think

like a human brain and be convincing enough to pass as human.

In 1951, the first artificial neural network, called SNARC (Stochastic Neural Analog Reinforcement Calculator), was created by Marvin Minsky and Dean Edmonds.

In 1952, IBM's Poughkeepsie Laboratory was where Arthur Samuel began working on some of the first machine learning programs. As a pioneer in the fields of artificial intelligence and computer gaming, Samuel developed a system that could play checkers. The program was one of a kind and one of the first programs that improved on its game by being able to learn. In 1959, Arthur Samuel came up with the term "machine learning" for systems that could learn and improve themselves with little to no human intervention.

In 1958, Frank Rosenblatt designed "Perceptron," which was the first artificial neural network with its primary function to recognize shapes and patterns.

In 1959, Bernard Widrow and Marcian Hoff created ADELIN, a neural network that could recognize binary patterns and predict what the next bit would be in a stream of bits. MADELINE was the next generation neural network to follow ADELIN. Both these models were developed at Stanford University. Although MADELINE's technology proved to be extremely useful, as it was able to detect and get rid of phone line echo, it didn't take off until the 1970s. This was due to the more popular Von Neumann architecture which proved a lot simpler to use and understand than complex neural networks. Ever since it took off in the late 1970s, MADELINE neural network technology has been used even into modern days.

In 1982, neural networks once again had the world's attention when Japan started to focus on advanced neural networks. America, not to be left behind, also allocated funding to the research of advanced neural networks. It was in the same year that John Hopfield put forward his suggestion of creating a network that mimicked, as best it could, the working of neurons. This network would work with bidirectional lines much the same as neurons do.

In 1986, Widrow and Hoff's neural network model was expanded on by researchers at Stanford University. The researchers extended the algorithm used by Widrow and Hoff in order to create "slow learners" by allowing for neural networks to use multiple layers. These "slow learners" would allow for the system to learn well into the future.

In 1997, after almost a decade of not much advancement in the field of machine learning, a computer called Deep Blue defeated the world chess champion of the time. Deep Blue was an IBM computer that initially started as a computer called ChipTest. ChipTest was invented by Feng Hsiung Hsu and a classmate of his, Murray Campbell. After the two students were hired by IBM in 1989, they continued their work on ChipTest but renamed it Deep Blue. The 1997 chess match against reigning champion Garry Kasparov lasted for 7 days and consisted of a six-game match. The match drew a lot of attention to the new computer system and machine learning.

The 21st century has seen many advances in machine learning, as business has come to see the advantages of what the concept can bring. With big data comes a need for machines that are able to quickly and accurately process it.

There are many huge machine learning projects that started at the turn of the century, and these projects include:

1. AlexNet
2. Amazon Machine Learning Platform
3. DeepFace
4. DeepMind
5. GoogleBrain
6. OpenAI

Chapter 2:

Approaches to Machine Learning

Although machine learning has been around for a few decades now, it never really took off until the twentieth century. Although some advancements were made before then, in the twenty-first century machine learning has become one of the world's most influential technologies. The advancements in the AI field have come along in huge leaps and bounds with no signs of having reached even a fraction of its potential.

If we look at movies such as the first *Star Trek*, the stars interacted with their computers. The computers were intelligent and could run predictions for the crew. The newer science fiction movies have even more advanced AI systems. These “fictional systems” amaze and fascinate the audiences who mostly think that these systems are just figments of the writer's imagination.

Most people, even without realizing it or taking too much time to think about it, interact with AI and ML every day. Each time you ask Siri or Alexa a question, you are interacting with an AI with ML capabilities. Even when doing a search on Google you are interacting with ML.

Machines inspired by human cognition in order to mimic the human brain systems have to go through a learning process. This learning process is where the machine has to figure out a set of rules, then build off those by trying out various scenarios. The machine learns from how well it performs while trying out different rules in order to successfully complete a scenario. The machine is not so much concerned with how it is being interacted with but rather with what meaningful results were produced from the interaction.

Machine Learning Terminology

Before you can really get into ML, there is some terminology that may be useful to know. The following terms are some of the most important ones to be aware of.

Machine Learning Data Set

The data set is an important part of the machine learning structure. A data set contains examples of data the machine needs to learn in order to solve problems.

Features

The example data will contain features and functions that the machine needs to use to learn from. These features are pieces of information that get fed to the ML system algorithm and are used to help the machine understand and learn.

Machine Learning Model

A machine learning model is the result of what a machine learning system has learned from its training. For instance, a self-driving car model would be trained to deduce the best and safest route it should take.

The Machine Language Process

Creating a machine language algorithm is a step-by-step process. The following steps make up the machine language creation process:

1. Collection of Data

This is the first stage to creating a ML algorithm, as it is what the algorithm will be based upon. This is all the data that the ML will need to learn from to deliver the desired results.

2. Preparation of Data

The second stage is sorting the data to find the features that the ML will need in order to perform optimally.

3. Machine Learning Training

At this stage the ML algorithm will be fed the carefully chosen feature-filled data collection. It is at this stage that ML will start to learn from that data it has been fed. It learns how to solve complex problems and produce solutions. For instance, a search engine is fed certain information when it starts out. The more people use the search engine, the more it is able to predict each person's preferences based on the pages they search.

4. Evaluating the Machine Learning Model

Evaluating the ML model is the phase where the ML algorithm gets put to the test in order to determine how successful it is.

5. Performance Enhancements

This is the phase where a person will continuously be fine-tuning and keeping the ML algorithm updated.

Chapter 3:

Machine Learning Environment Setup

To be able to learn and practice machine learning, you will first need to get your Python/Anaconda, Scikit-Learn, and TensorFlow environment set up. The following chapter will take you through how to set up your environment. With your environment set up correctly, you will be able to follow the examples in this book.

Setting Up Python and Anaconda

Scikit-learn and TensorFlow can be used with Python 2.7 and above. They will also work with Anaconda 2.7 and higher. There are a few libraries that need to be installed after you have installed Python and Anaconda which are a requirement for installing Scikit-Learn or TensorFlow. These will be covered in the corresponding sections of this chapter.

Note:

- This book assumes that you are using the **Windows** 7 or higher operating system to download and install Python, Anaconda, Scikit-Learn, and TensorFlow. While MAC and Linux download and installation procedures differ, the exercises can still be used on these operating systems.
- If you already have Python on your system, you can skip the “**Installing Python**” section and go to the “**Installing Anaconda**” section if you wish to use Anaconda.
- If you have Python already installed on your machine, you may want to check if you have the following libraries installed:
 - NumPy v 1.18.4
 - SciPy v 1.4.1
 - Pip
- If you have the above libraries installed, or the version pertaining to the release of Python installed on your machine, skip to the “**Installing Anaconda**” section.
- If you are not installing Anaconda or already have it installed on your system, you can skip to “**Installing Scikit-Learn**” or “**Installing TensorFlow**” section should you already have one or the other installed.
- If you already have your Python, Scikit-Learn, and TensorFlow environment set, you can skip this chapter altogether and proceed to the next chapter “**Using Scikit-Learn.**” However, it will do you no harm checking or even upgrading your current system environment for Scikit-Learn and TensorFlow.

Installing Python

The latest version of Python is 3.8.3 and it can be retrieved from the Python.org website. For the sake of this book, version 3.8.3 will be used to keep the book up to date as at time of print. As it is open source software, it is free to download and use.

If you already have Python installed on your machine you can use it to follow along as long as it is 2.7 or higher. It is good to note that there may be differences between the earlier versions and the current ones. Please check the Python.org website for more details on the differences between the versions.

Download Python

On the Python.Org website go to the “**Downloads**” screen.



There is the option to download earlier releases under the “**Active Python Releases**” section of the page.

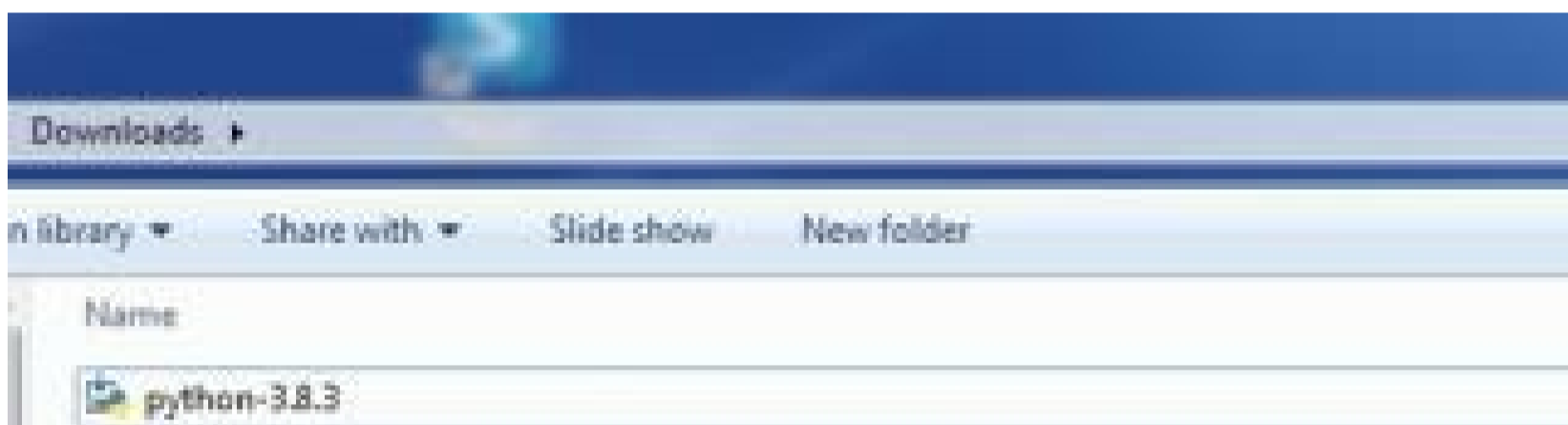
Active Python Releases				
For more information, visit the Python Developer's Guide.				
Python version	Release name	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10-01	Python 3.8
3.7	bugfix	2019-09-17	2022-06-27	Python 3.7
3.6	security	2019-12-18	2021-11-22	Python 3.6
3.5	security	2018-09-18	2020-06-23	Python 3.5
2.7	end of life	2010-07-26	2020-01-01	Python 2.7

Looking for a specific release?		
Python releases by version number		
Release version	Release date	Click for more
Python 3.8.3	May 14, 2020	Download
Python 3.8.2	April 29, 2020	Download

For this exercise you are going to use **Python 3.8.3**.
Choose the version from the “**Active Python Releases**” or click on the “**Download Python 3.8.3**” button beneath the “**Download the latest version for Windows**” section. Make sure it is the 64-bit version, as the current version of TensorFlow works best with the 64-bit.



The file will download into the “**Downloads**” folder on your computer.



Check that the file is there once the download has been completed.

Installing Python on Windows

From the “**Downloads**” folder, double click on the “**python-3.8.3**” executable file.

Click on the “**Run**” button from the “**Open File - Security Warning**” dialogue box that will appear.

Make sure to tick the “**Add Python 3.9 to Path**” at the bottom of the “**Install Python 3.8.3 (xx-bit)**” screen. Keep the default installation directory, unless you are a more advanced user and would like to install the software in another location. For the sake of this training exercise, the default directory has not been changed to simplify the process.

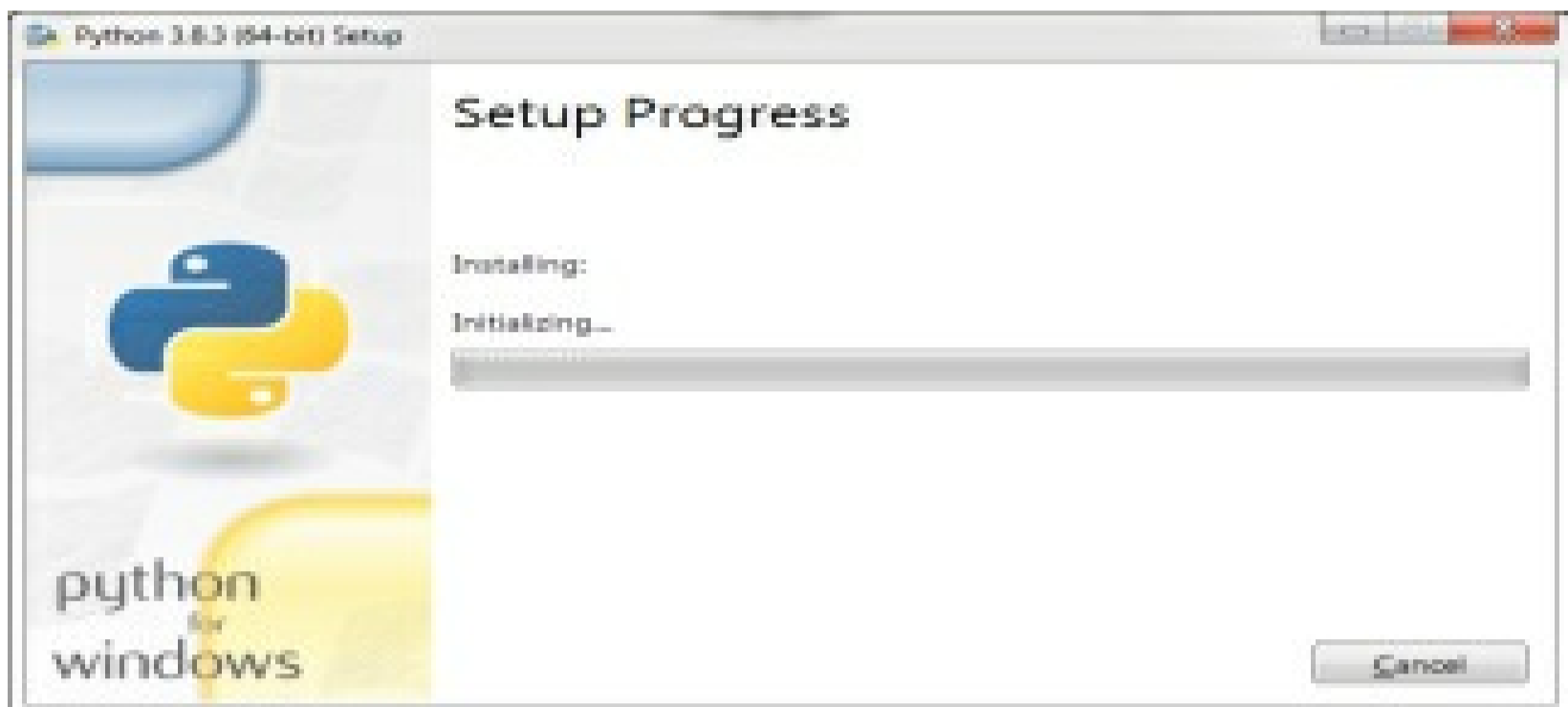
Click the “**Install Now**” option on the “**Install Python 3.8.3 (xx-bit)**” screen.



If the “**Access Control**” warning screen comes up, click on “**Yes**” to allow the application permission to install on

the system.

The “**Setup Progress**” screen will appear with the installation progress for installing Python.

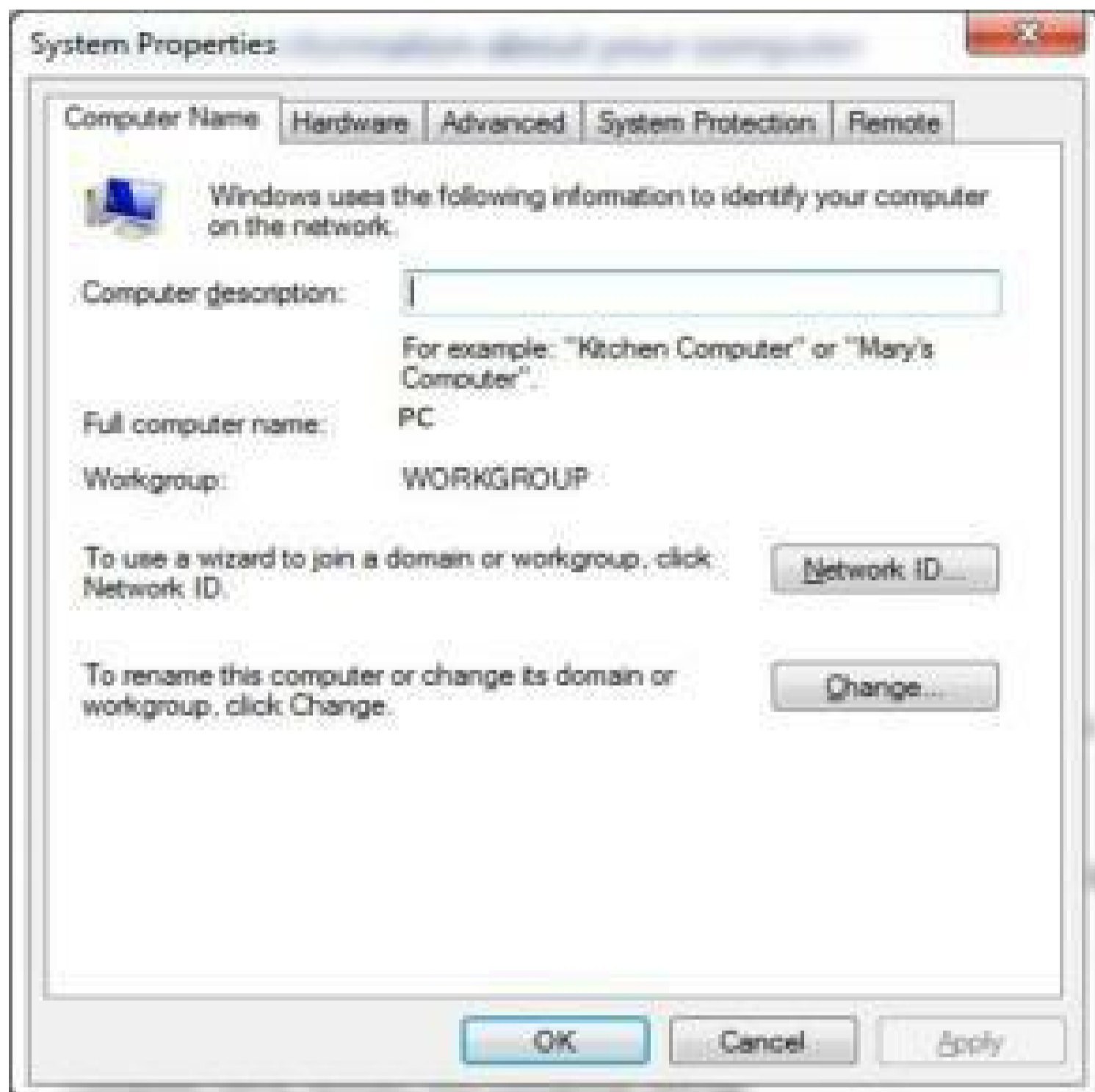


When Python is finished installing the “**Setup was successful**” screen will appear. Python will now be installed on your system.

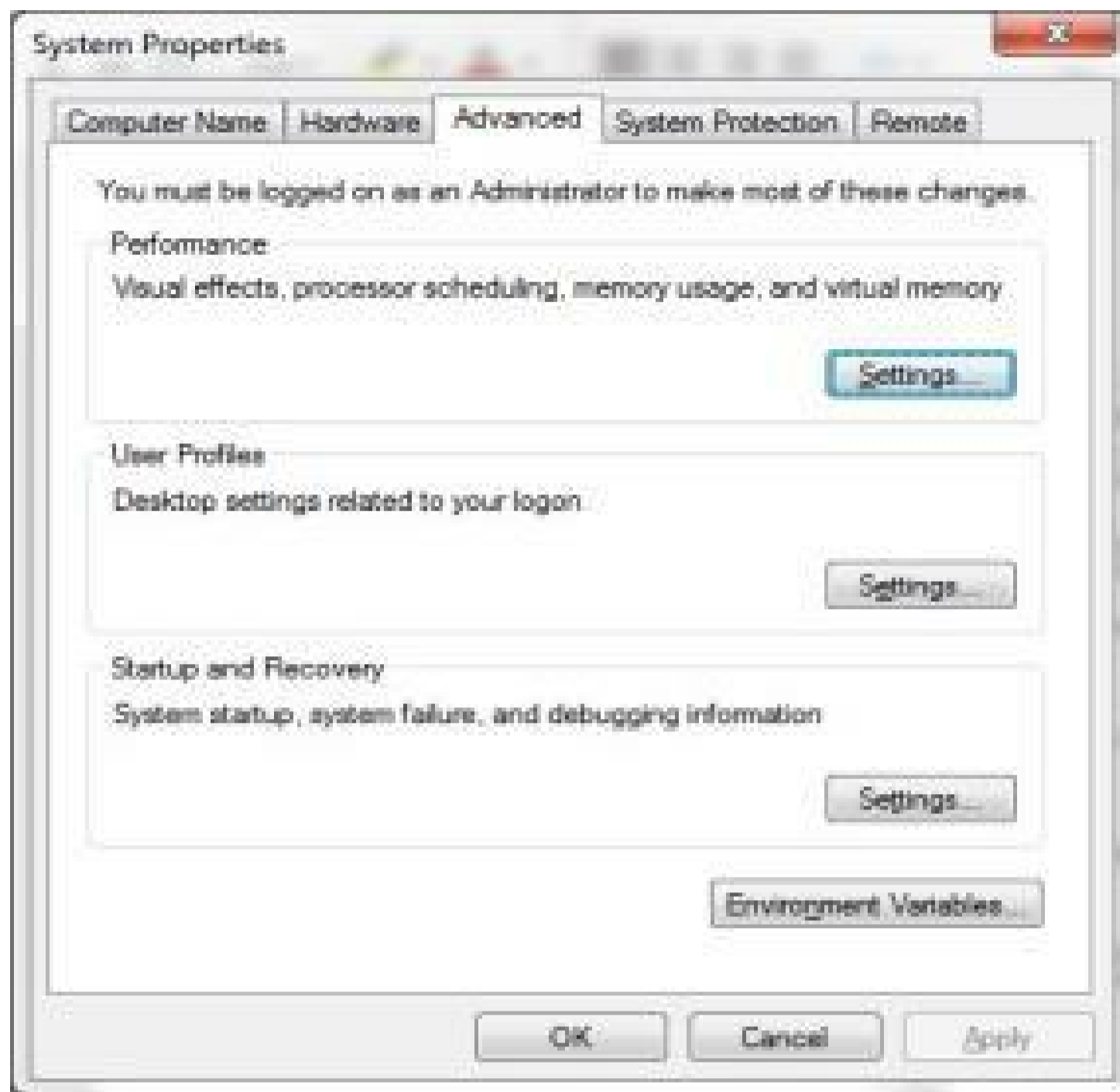


Setting Up the Python Environment

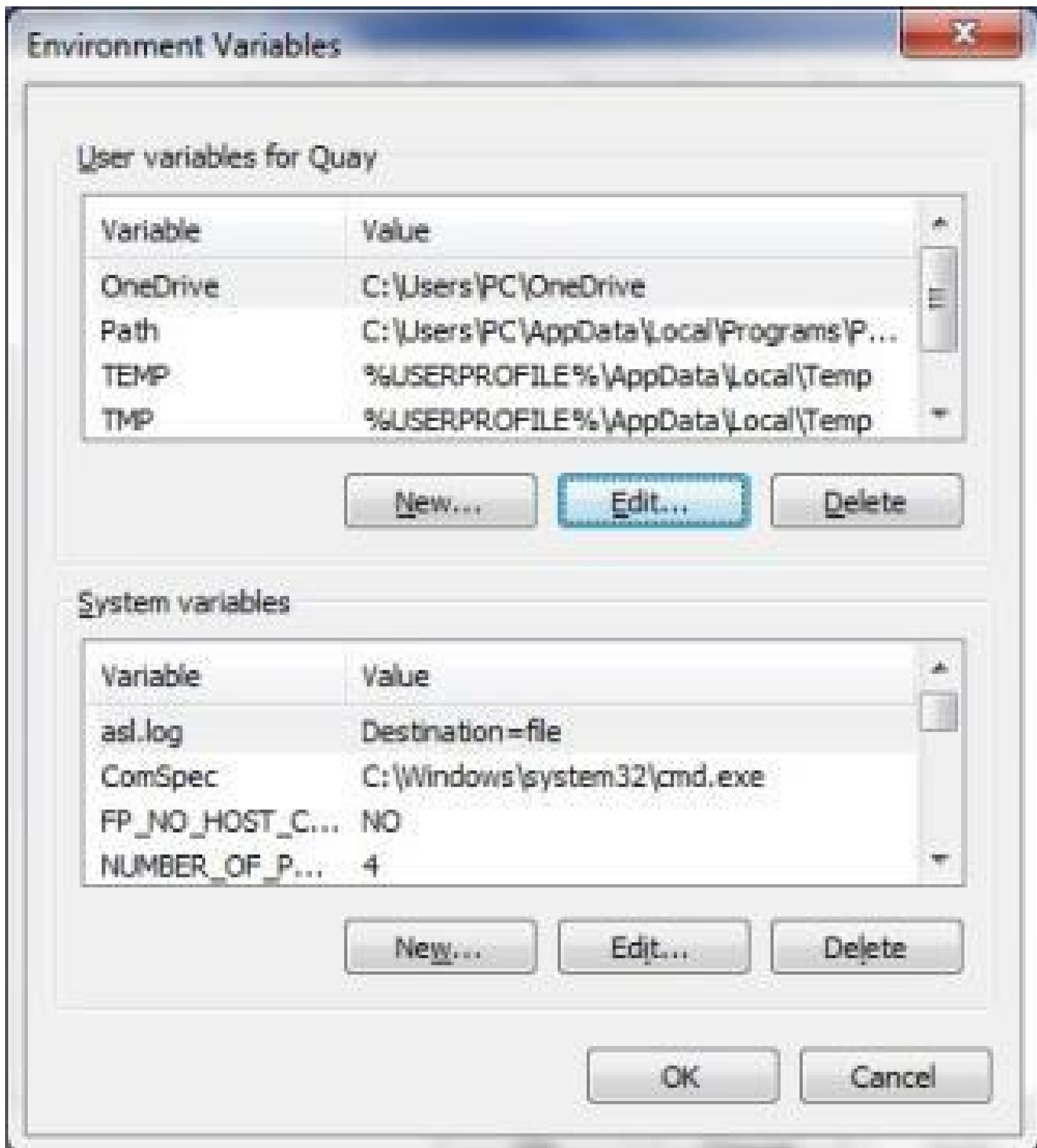
You will need to check to see if Python is running on the system. If you did not tick the “Add Python to Path” box, you will need to add it to the **System Environment Variable** path. To check if it was put into the **System Environment Variable** path, go to the “**Control Panel - System and Security - System Properties.**”



Click on the “**Advanced**” tab.



Click on the “**Environment Variables**” button.



Click on “**Path**” under “System variables” and then the “**Edit**” button.

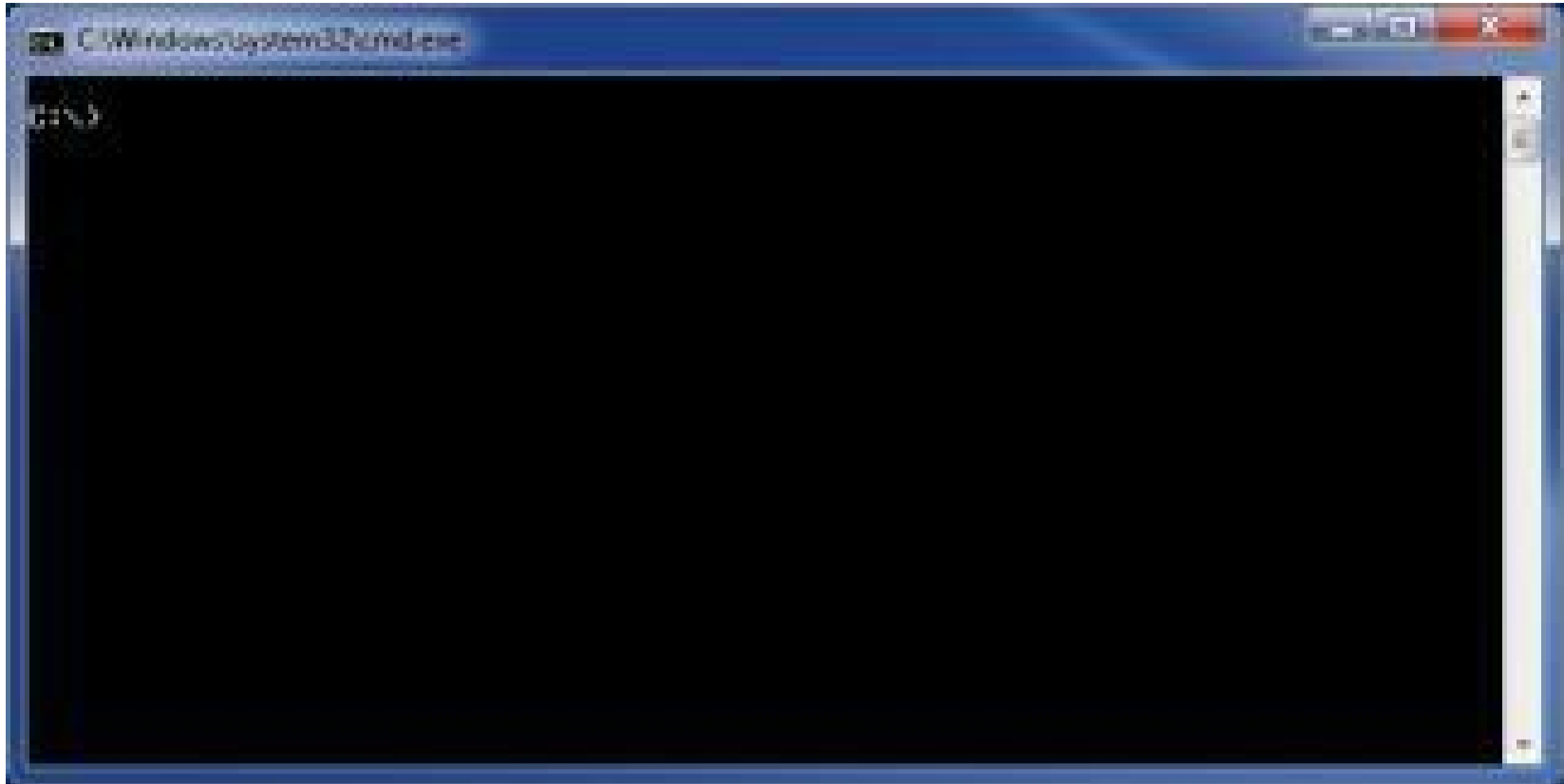
For systems earlier than Windows 10, you will need to place your cursor at the end of the “**Variable value**” path to check if the following line (or similar, depending on where you installed Python) appears. For Windows 10 you will add a new path to the list of variables.

The following variable path should appear in the “**Variable value**” box:

;C:\Python\Python 38-32\

If there is no reference to the Python.exe file, find the path to the Python executable file and type it in here as per the example above. **NOTE:** Do not forget to separate each system path variable entry with a “;” for Windows OS earlier than Windows 10, or it will create problems and not pick up the application. When you are done checking the System path variable click “**OK**” until you have exited the **System Variable** screen. You can also close down the “**Control Panel**” before moving on to test the Python installation.

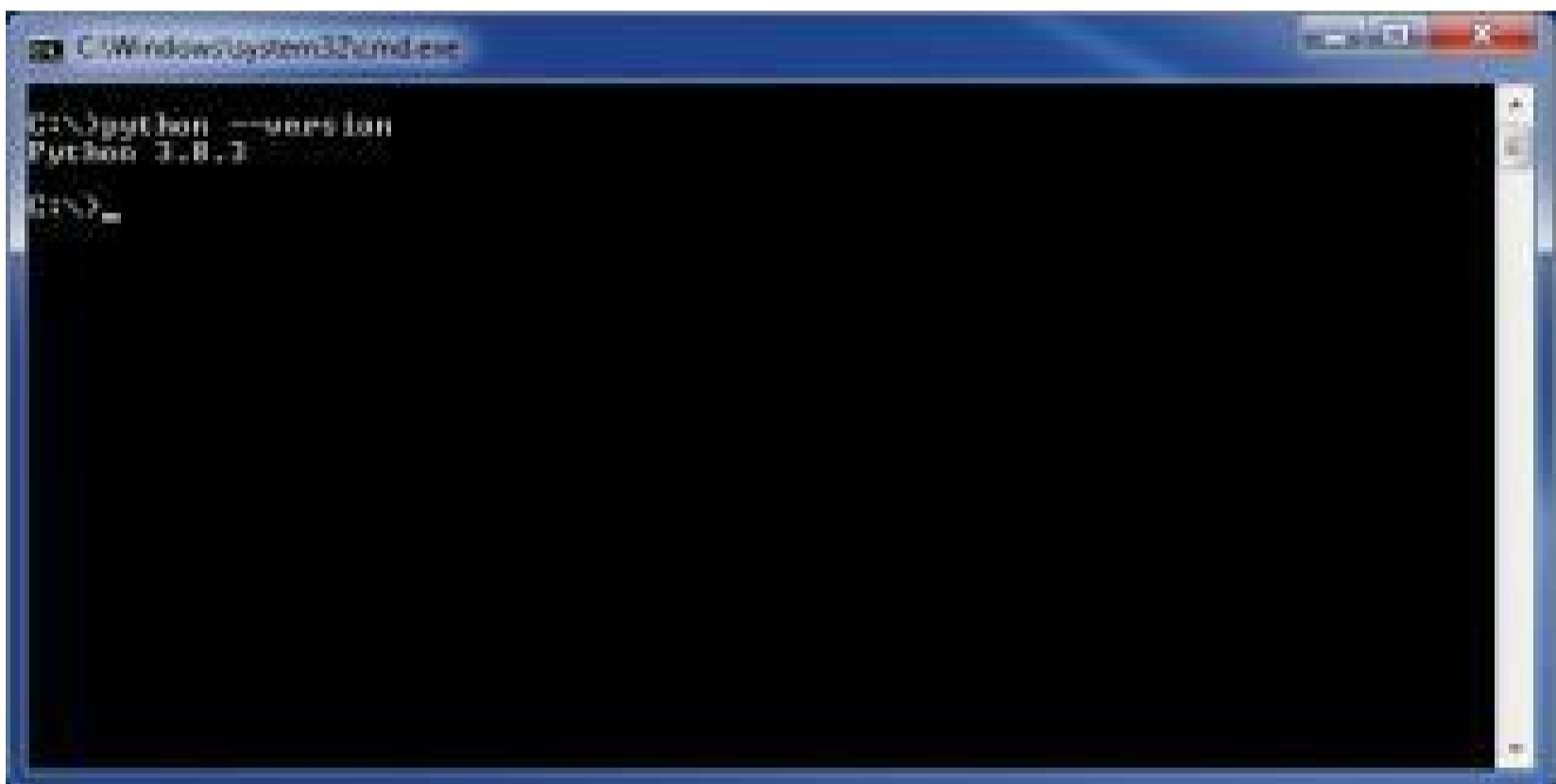
From the Start menu, type **cmd** in the search box and hit enter. This will load the command line screen to test that the Python installation installed correctly and can be accessed. The following screen (or similar) will appear:



To test the Python installation type the following into the command line:

```
python --version
```

The following or similar should appear on the screen:



The version of Python that was recently installed on the machine should appear beneath the command you typed in. If it does not appear there, you will need to check the **System Variable Path**.

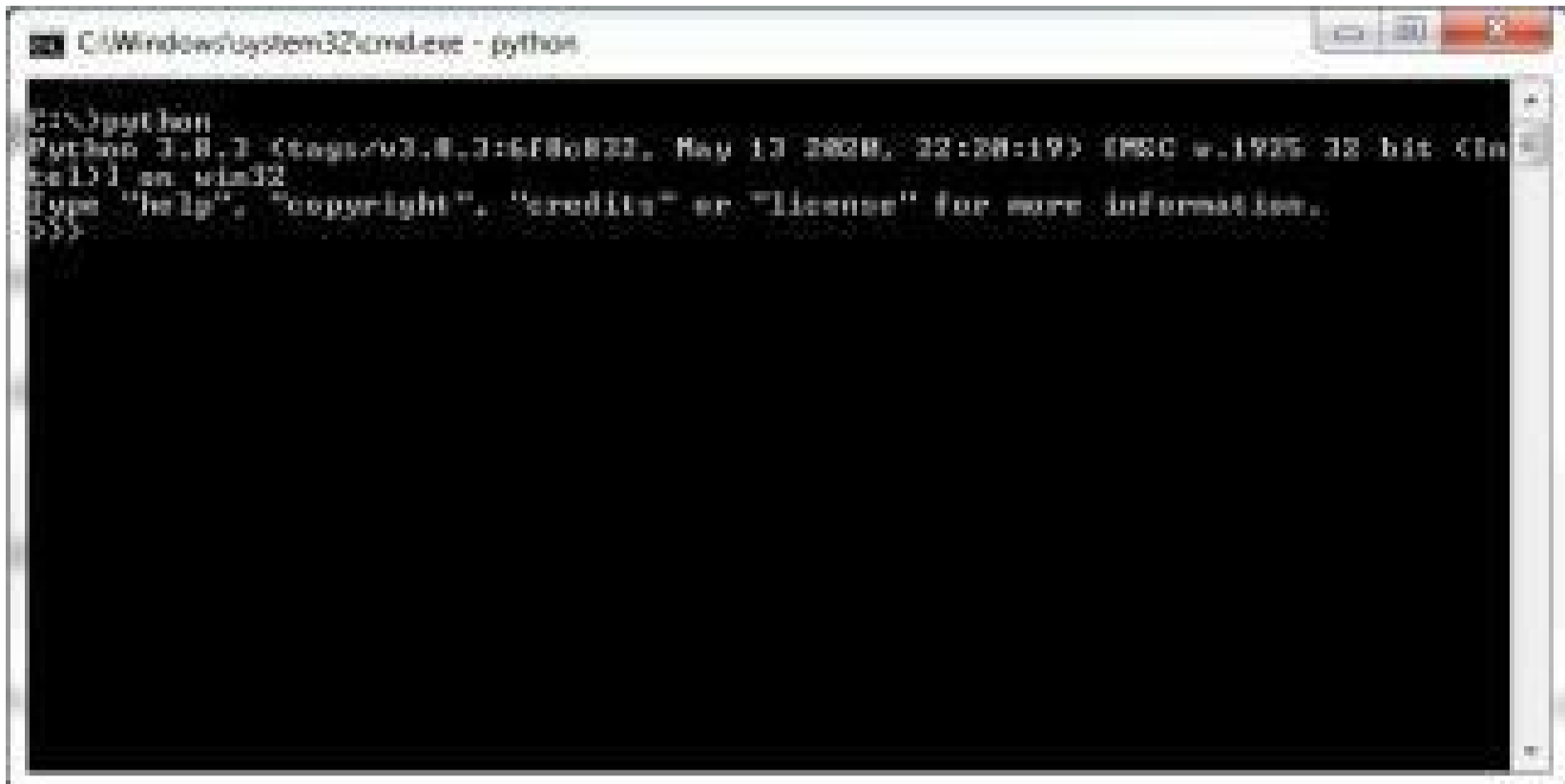
There are a few ways to load the Python command line:

- Create a desktop shortcut
- Access Idle or the Python command line from the “**Start**” menu “**Programs**” option
- Access Python through the **DOS** command line using **cmd** from the “**Start**” menu **run/search** facility.

For the sake of this exercise the **cmd** option is going to be used.

From the cmd command line prompt type:

```
python
```



If you are going to be using Python you will need to use **pip** (Preferred Installer Program). **Pip** is the package installer for Python and if you have Python version 2.7 and higher, it should already be installed with Python. To check that **pip** is installed, type the following into the Python command line:

```
pip --version
```

It should return the version of pip. If it is older than 20.1.1 (the new version at the time of writing this book is version 20.1.1) then **pip** will need to be upgraded. In order to be able to run **pip** from the command line, you can add the **pip** program path to the System Environment Variable Path as you did for the Python executable. The **pip** executable file is usually found in the following Python directory:

```
<Python installation path>\Python\Python38-32\Scripts
```

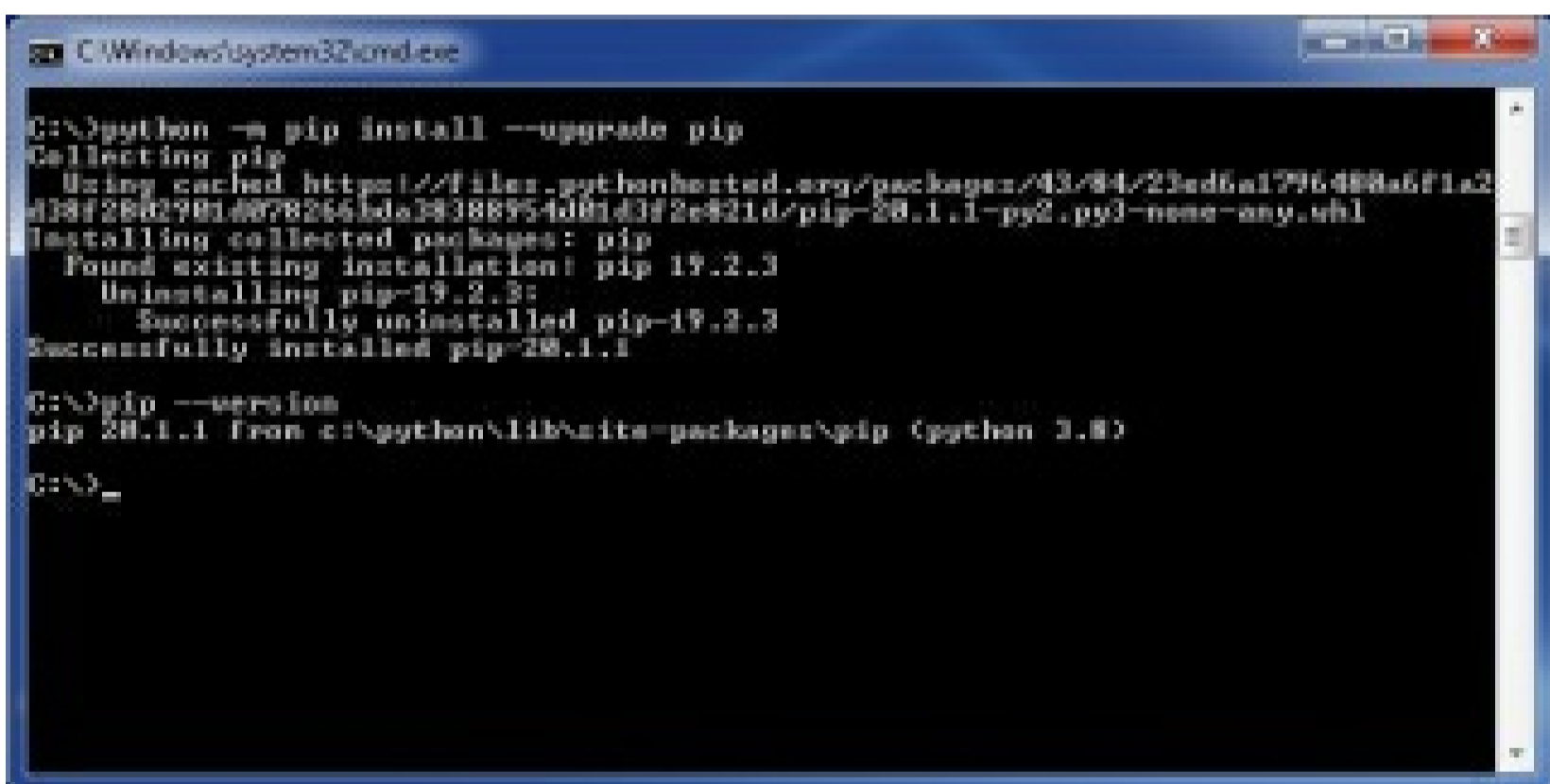
To upgrade **pip** type the following into the command line prompt:

```
python -m pip install --upgrade pip
```

When the **pip** upgrade is finished type:

```
pip --version
```

The version should be 20.1.1



The Python **pip** utility should now be the latest version.

Install NumPy With Pip

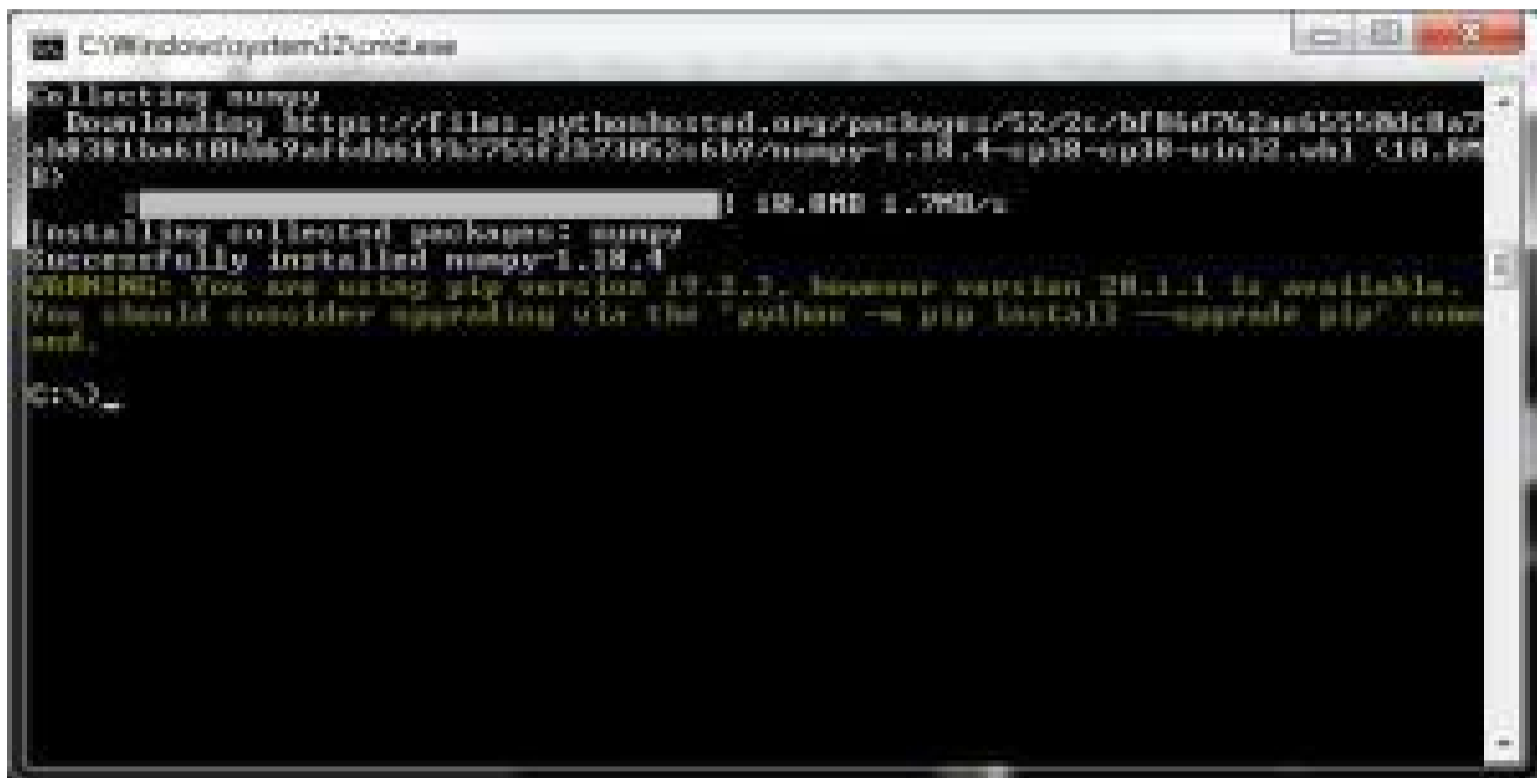
Now you need to check if the NumPy library is installed. NumPy is needed for the support of matrices, high-level math functions, and multi-dimensional arrays that are large. You can check if NumPy was pre-installed with Python by using the following test on the Python command line:

```
import numpy
```

If the command line returns an error “**No module named ‘numpy’**” you will need to get the library. At the command line prompt type the following:

```
pip install numpy
```

If you have pip loaded and referenced correctly in the System Variable Path, pip will download and install the NumPy library as per the image below.



```
C:\Windows\system32\cmd.exe
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/52/2c/bf84d762a655586c8a7
4b8381ba618b69af1db617b2755f2b73852a6b9/numpy-1.18.4-cp38-cp38-win32.whl (18.8M
B)
  |#####| 18.8MB 1.7MB/s
Installing collected packages: numpy
Successfully installed numpy-1.18.4
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' com
mand.
C:\>
```

Install SciPy With Pip

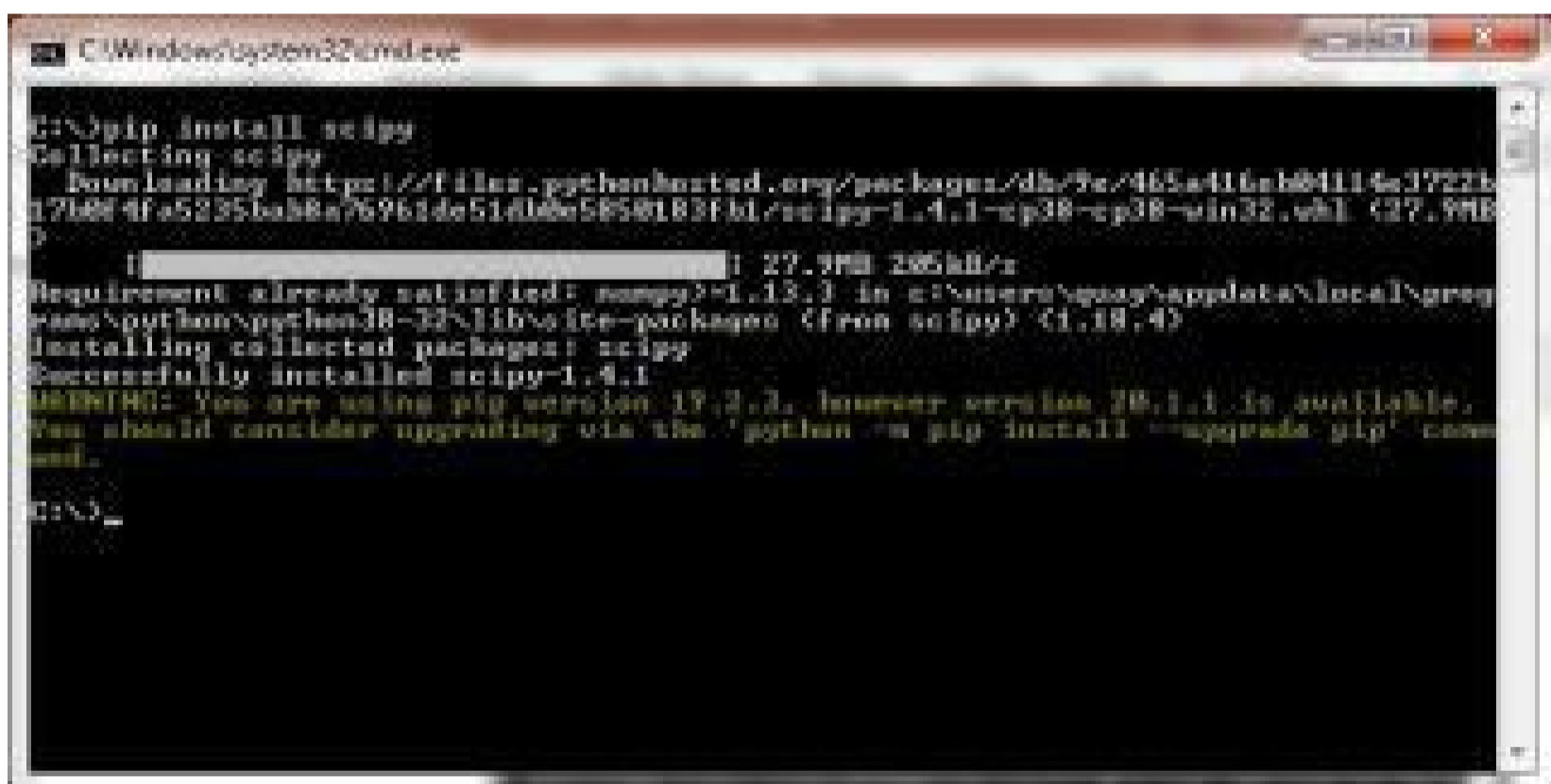
Now you need to check if the SciPy library is installed. You can check if SciPy was pre-installed with Python by using the following test on the Python command line:

```
import scipy
```

If the command line returns an error “**No module named ‘scipy’**” you will need to get the library. At the command line prompt type the following:

```
pip install scipy
```

If you have **pip** loaded and referenced correctly in the System Variable Path, **pip** will download and install the SciPy library as per the image below.



```
C:\Windows\system32\cmd.exe
C:\>pip install scipy
Collecting scipy
  Downloading https://files.pythonhosted.org/packages/d8/9e/465a416ab04114e3722b
17b884fa5235ba38a76961de51d0be5858183f61/scipy-1.4.1-cp38-cp38-win32.whl (27.9M
B)
  |#####| 27.9MB 285kB/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\quay\AppData\Local\prog
rams\python\python38-32\lib\site-packages (from scipy) (1.18.4)
Installing collected packages: scipy
Successfully installed scipy-1.4.1
WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' com
mand.
C:\>
```

Installing Anaconda

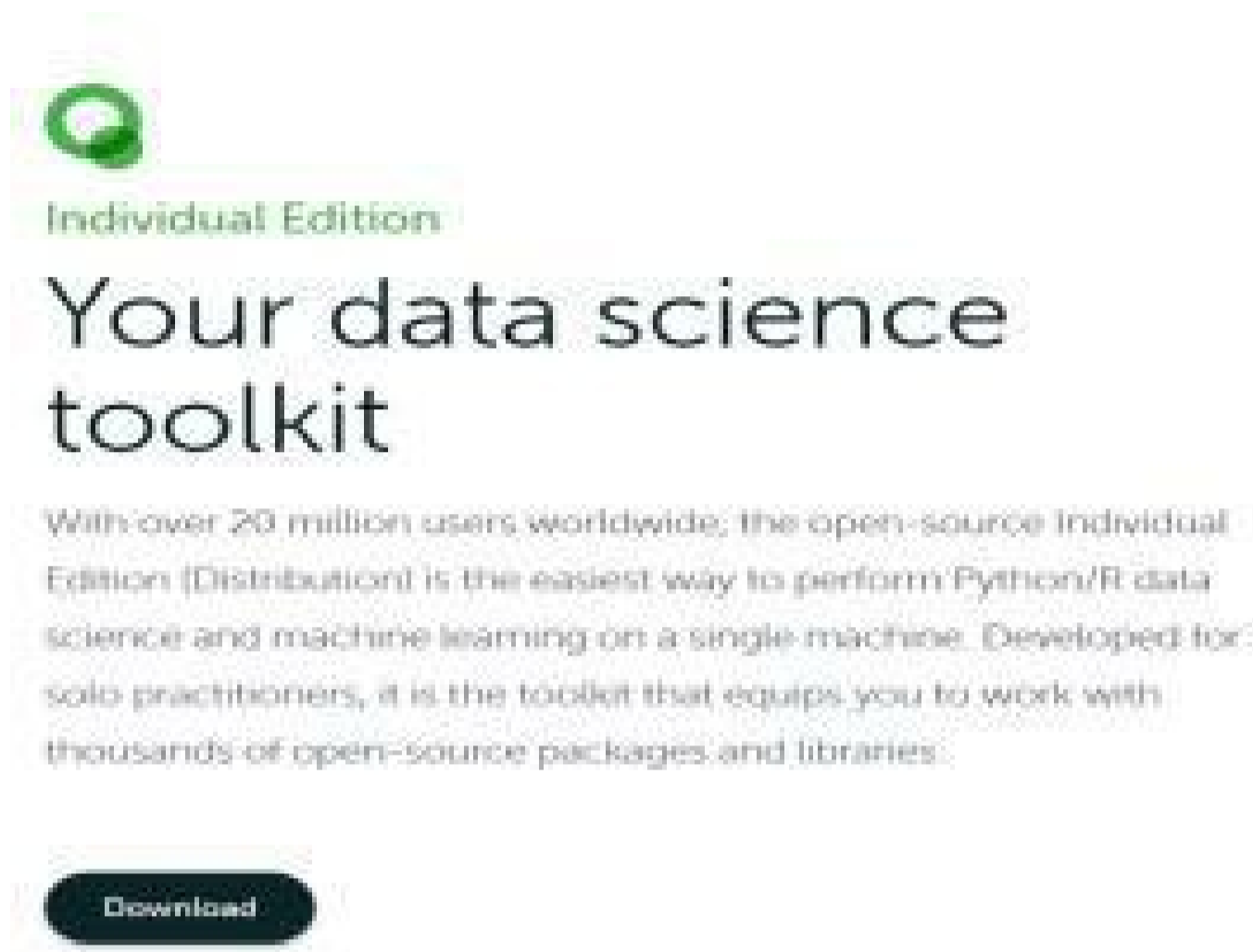
The latest version of Anaconda is 3.7 and it can be retrieved from the Anaconda.com website. For the sake of this book, version 3.7 64-bit will be used to keep the book up to date at time of print. The 64-bit version is advisable as TensorFlow works best with 64-bit. As TensorFlow is open source software, it is free to download and use.

If you already have Anaconda installed on your machine, you can use it to follow along as long as it is 2.7 or

higher. It is good to note that there may be differences between the earlier versions and the current ones. Please check the Anaconda.com website for more details on the differences between the versions.

Download Anaconda

On the Anaconda.com website, scroll down the page until you find the “**Downloads**” button.



There is an option to download earlier releases under the “**Anaconda Installers**” section of the page.



For this exercise you are going to use the **Python 3.7** version which relates to your machine (i.e. 32-bit or 64-bit). Choose the version from the “**Anaconda Installers**” screen and it will start to download the installer file into the “**Downloads**” folder on the machine.



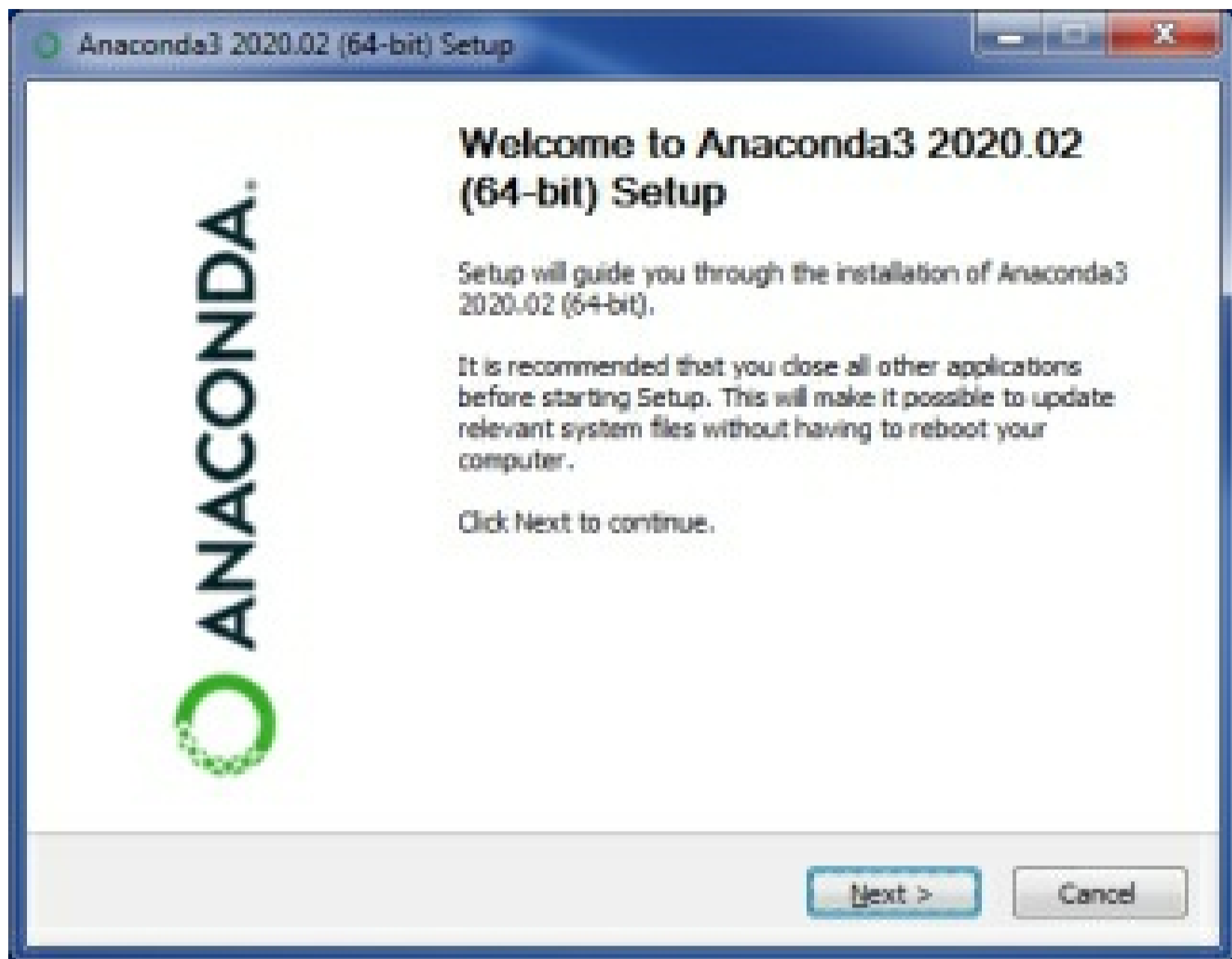
Check that the file is there once the download has been completed.

Installing Anaconda on Windows

From the “**Downloads**” folder, double click on the “**Anaconda3.2020.02-Windows-xx**” executable file.

Click on the “**Run**” button from the “**Open File - Security Warning**” dialogue box that will appear.

Click “**Next**” at the bottom of the “**Welcome to Anaconda3 2020.02 (xx-bit) Setup**” screen.



Click the “**I Agree**” button on the “**ANACONDA License Agreement**” screen.

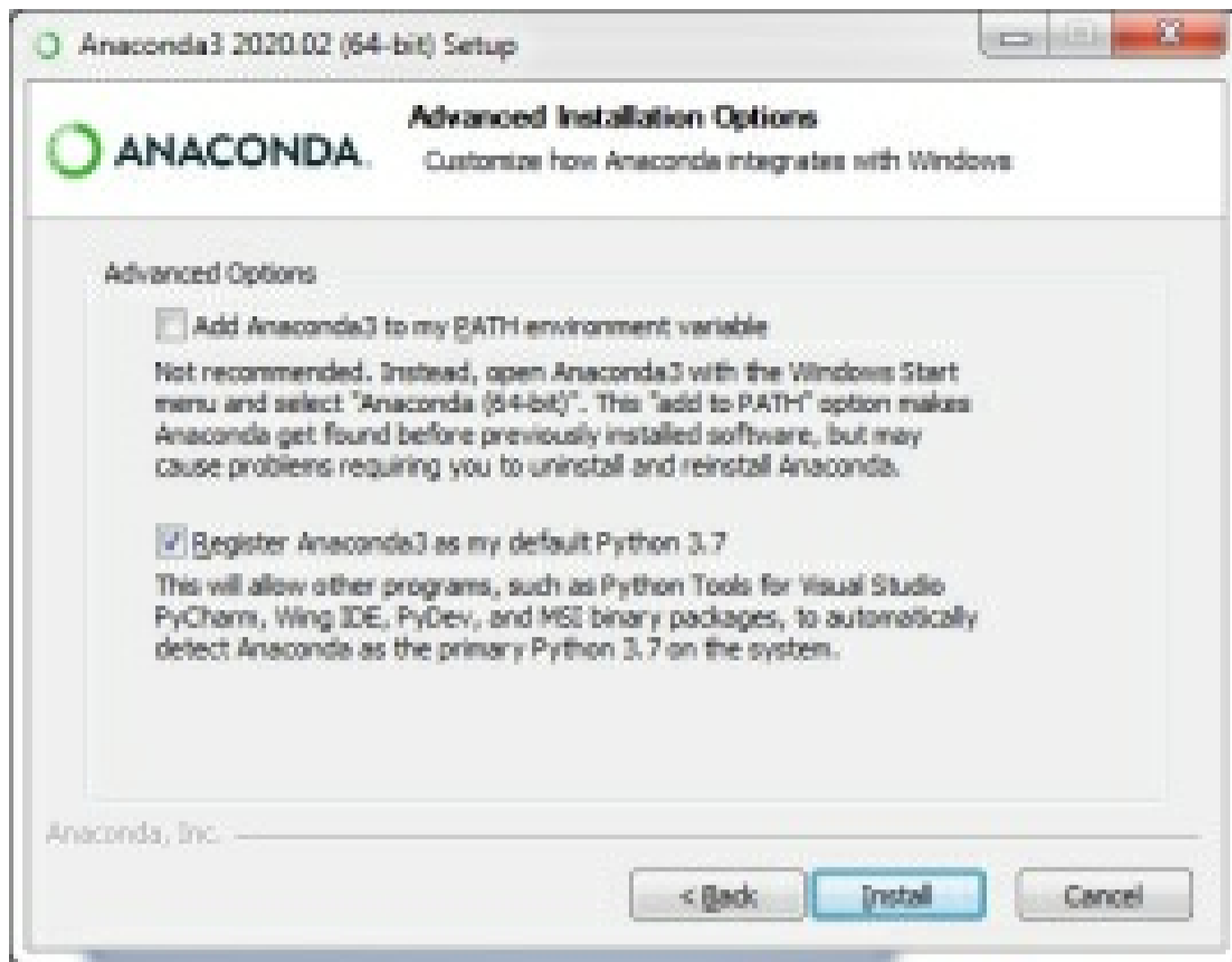
Choose either “**Just me (recommended)**” or “**All Users (requires admin privileges)**” depending on your preferences at the “**Select Installation Type**” screen.

At the “Choose Install Location” screen, it is recommended to keep the default installation directory or set it to the directory of your choice. The directory for the sake of this book will be:

C:\anaconda3

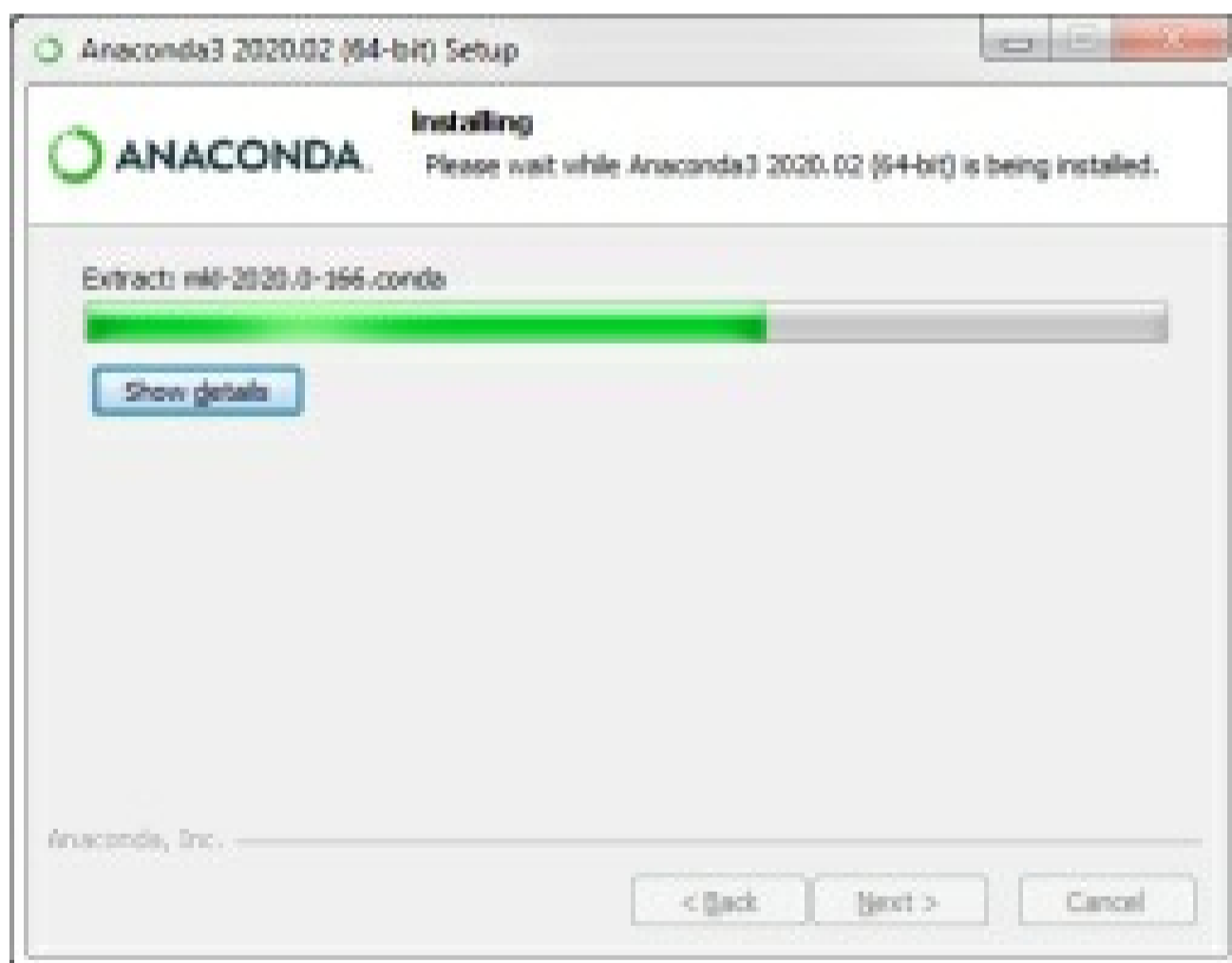
Click the “**Next**” button when you are ready to proceed with the installation.

At the “**Advanced Installation Options**” screen, leave the “Add Anaconda3 to my PATH environment variable” **UNCHECKED**. This will be added later after the installation is complete. You can leave the “**Register Anaconda3 as my default Python 3.7**” **CHECKED**.



Click the “**Install**” button at the bottom of the screen to continue the installation process. The installation will continue with a progress bar indicating the installation progress.

When Anaconda is finished installing the “**Installation Complete**” screen will appear. Anaconda will now be installed on your system.



Click the “**Next**” button which will take you to an informational screen.

Press the “**Next**” button to be taken to the “**Completing Anaconda3 2020.02 (xx-bit) Setup**” screen. Uncheck the “**Anaconda Individual Tutorial**” and “**Learn More About Anaconda**” then click the “**Finish**” button.

Setting Up the Anaconda Environment

There are a few ways to load the Anaconda command line:

- Create a desktop short-cut
- Access Anaconda command line from the “**Start**” menu “**Programs**” option

- Access Anaconda through the **DOS** command line using **cmd** from the “**Start**” menu **run/search** facility.

For the sake of this exercise the “**Start**” menu “**Programs**” option is going to be used.

Go to the “**Start**” menu “**Programs**” and find “**Anaconda3 (xx-bit)**” then open up the folder.

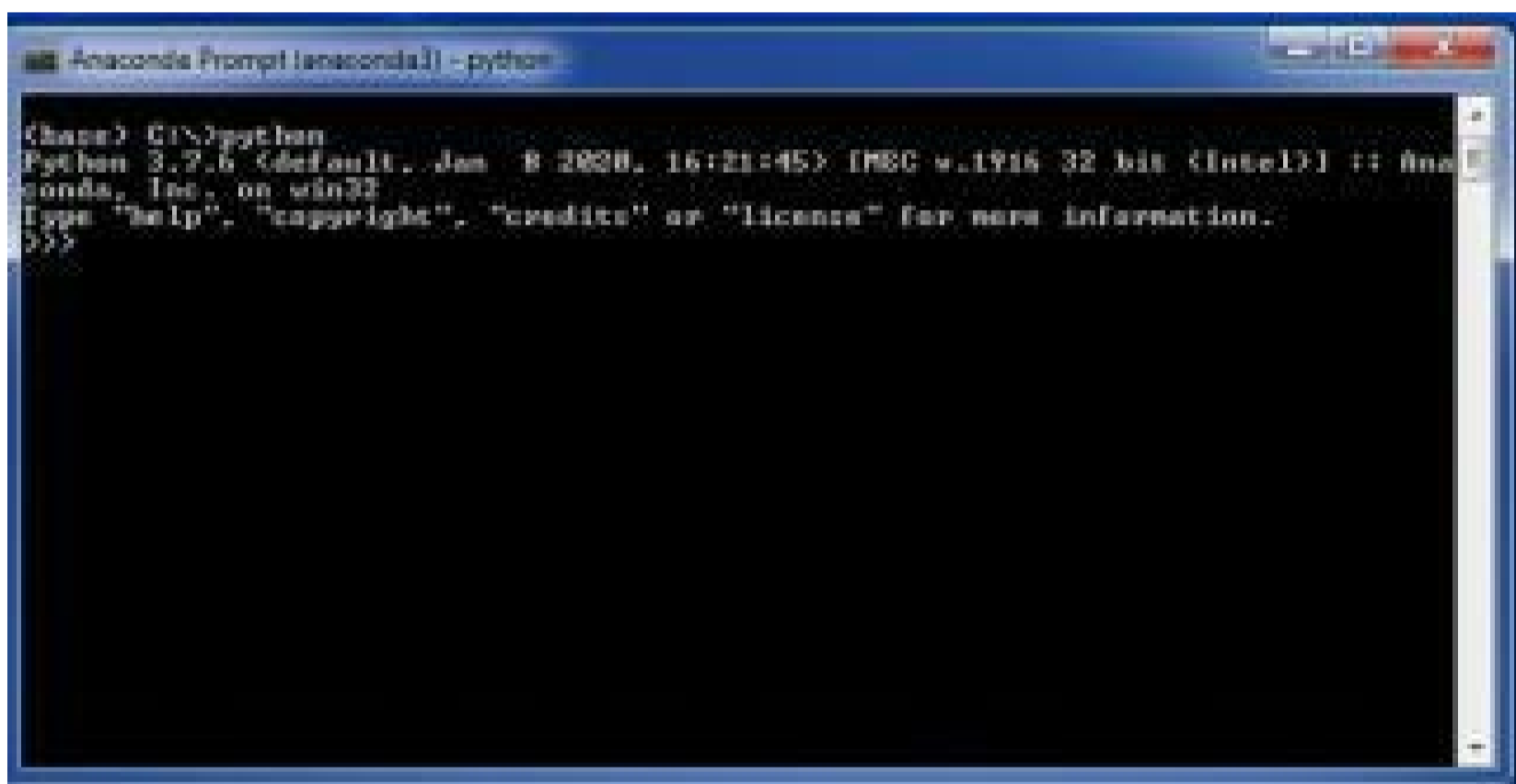
Choose “**Anaconda Prompt (anaconda3).**”



At the command line prompt type:

`python`

The Anaconda Python command line shell will load and look similar to the screen below.



If you want to test the Anaconda installation, import a file called “The Zen of Python” by Tim Peter by typing the following into the command line:

`import this`

Check that “**conda**” is installed in Anaconda, as it is the package installer for Anaconda.

If you have the **Anaconda Prompt** open and **Python** loaded, first exit the Python command line by typing the following:

`exit()`

This will take you back to the **Anaconda Prompt** and you can type in the following:

`conda --version`

This should show the version of conda. At the time of printing this book the latest version was 4.8.3. **Conda.exe** should come preloaded with the newer versions of Anaconda. However, they are not always the latest version of **conda** and may need to be upgraded. If the currently installed version of conda is older than version 4.8.3 then you should upgrade to the latest version.

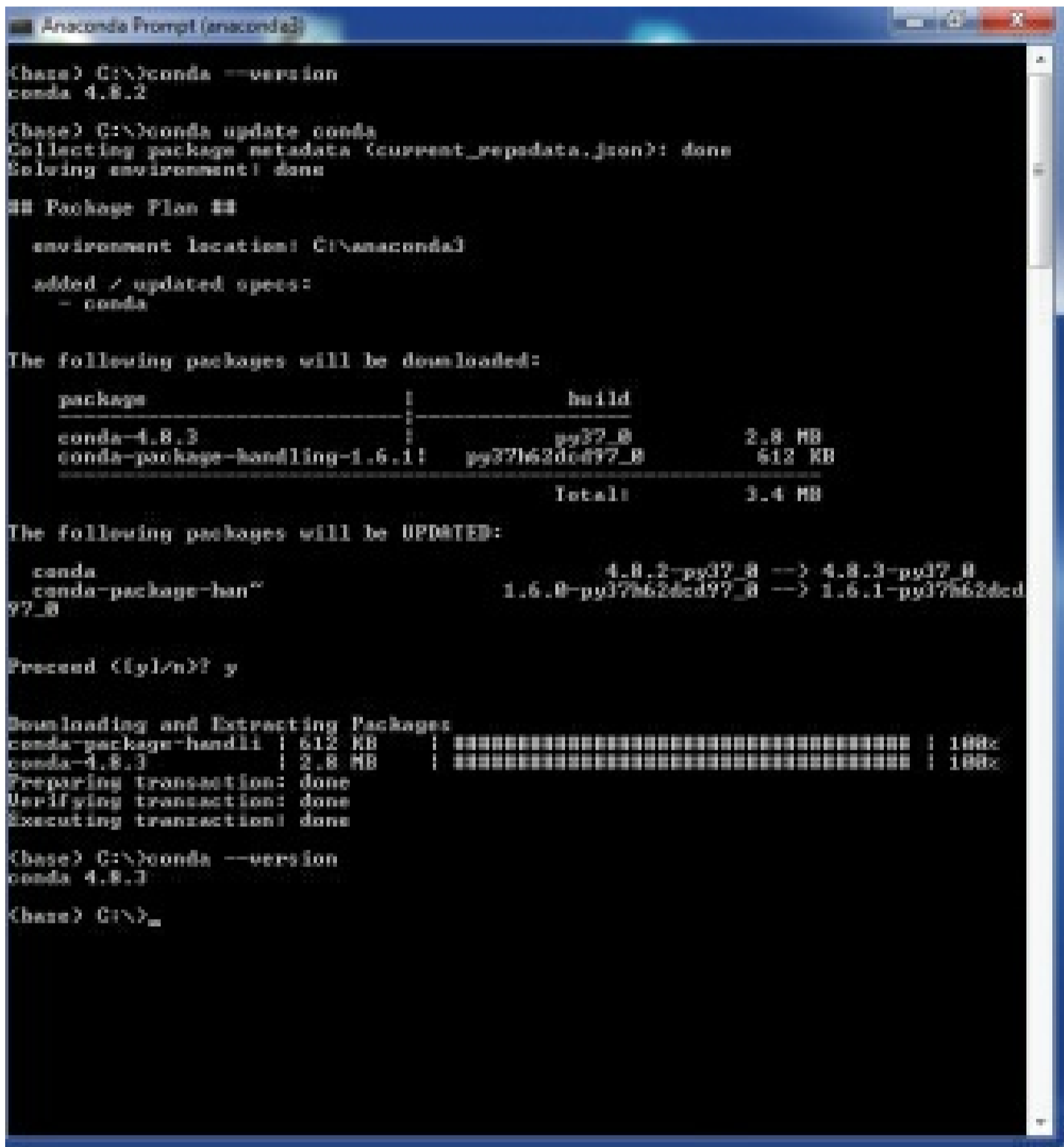
To upgrade **conda**, type the following into the **Anaconda Prompt** command line:

```
conda update conda
```

When the **conda** upgrade is finished type:

```
conda --version
```

The **conda** version should be 4.8.3

A screenshot of the Anaconda Prompt window. The terminal shows the command 'conda --version' returning 'conda 4.8.2'. Then 'conda update conda' is entered, followed by 'Collecting package metadata (current_repodata.json): done' and 'Solving environment: done'. A '## Package Plan ##' section shows the environment location as 'C:\anaconda3' and lists 'conda' as the package to be updated. A table shows the packages to be downloaded: 'conda-4.8.3' (2.8 MB) and 'conda-package-handling-1.6.1' (612 KB), with a total of 3.4 MB. Below this, it shows the packages to be updated: 'conda' from 4.8.2 to 4.8.3 and 'conda-package-handling' from 1.6.0 to 1.6.1. The user is prompted to proceed and types 'y'. The terminal then shows the progress of downloading and extracting these packages, both reaching 100%. Finally, it shows 'Preparing transaction: done', 'Verifying transaction: done', and 'Executing transaction: done'. The command 'conda --version' is run again, returning 'conda 4.8.3'. The prompt ends with a new line.

```
(base) C:\>conda --version
conda 4.8.2

(base) C:\>conda update conda
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\anaconda3

  added / updated specs:
    - conda

The following packages will be downloaded:



| package                      | build          | size   |
|------------------------------|----------------|--------|
| conda-4.8.3                  | py37_0         | 2.8 MB |
| conda-package-handling-1.6.1 | py37h62ded97_0 | 612 KB |
| Total:                       |                | 3.4 MB |



The following packages will be UPDATED:

  conda                4.8.2-py37_0 --> 4.8.3-py37_0
  conda-package-handl 1.6.0-py37h62ded97_0 --> 1.6.1-py37h62ded97_0

Proceed [y/n]? y

Downloading and Extracting Packages:
conda-package-handl | 612 KB | ##### | 100%
conda-4.8.3         | 2.8 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\>conda --version
conda 4.8.3

(base) C:\>
```

The Anaconda **conda** utility should now be the latest version.

Install NumPy and SciPy With Conda

To find out if NumPy and SciPy are loaded, type in the following for a full list of loaded modules:

```
conda list
```

Scroll the list to ensure that Numpy and SciPy are loaded. If they are not loaded, you will have to install them by typing in the following at the **Anaconda Prompt**:

- **To download NumPy**
conda install numpy
- **To download SciPy**
conda install scipy

Installing Scikit-Learn

Either Python **pip** or Anaconda **conda** can be used to install Scikit-learn.

Installing Scikit-Learn with Python Pip Without a Virtual Environment

To install Scikit-Learn with Python and ensure it is the latest version that is compatible with your Python installation, open a **cmd** screen and type the following:

```
pip install -U scikit-learn
```

The following screen will appear when installing Scikit-Learn.



```
C:\Windows\system32\cmd.exe - pip install -U scikit-learn
C:\>pip install -U scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-0.23.1-cp38-cp38-win32.whl (6.8 MB)
    |#####| 6.8 MB 2.2 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in c:\users\quay\appdata\local\programs\python\python38-32\lib\site-packages (from scikit-learn) (1.18.4)
Requirement already satisfied, skipping upgrade: scipy>=0.19.1 in c:\users\quay\appdata\local\programs\python\python38-32\lib\site-packages (from scikit-learn) (1.4.1)
Collecting joblib>=0.11
  Downloading joblib-0.15.1-py3-none-any.whl (298 kB)
    |#####| 298 kB 123 kB/s
Installing collected packages: threadpoolctl, joblib, scikit-learn
```

Allow the installation to run to completion, which could take a few minutes.

Installing Scikit-Learn with Python Pip in a Virtual Environment

If you have an Anaconda installation, it is best to create a different virtual environment to run Scikit-Learn for Python in. To run this from the command line prompt, follow these instructions:

Install the virtual environment for Python:

```
pip install virtualenv
```

Create the virtual environment for Scikit-Learn:

```
python -m venv sklearn-venv
```

Once the environment has been set, type the following into the command line prompt to activate the sklearn-venv environment:

```
sklearn-venv\Scripts\activate
```

Once that script has run, you can then install Scikit-learn with the same commands as used above:

```
pip install -U scikit-learn
```

Allow the package installer to successfully install the library.

Checking the Python Pip Scikit-Learn Installation

To check the Scikit-Learn **pip** installation type the following into the command line once the installation has completed successfully:

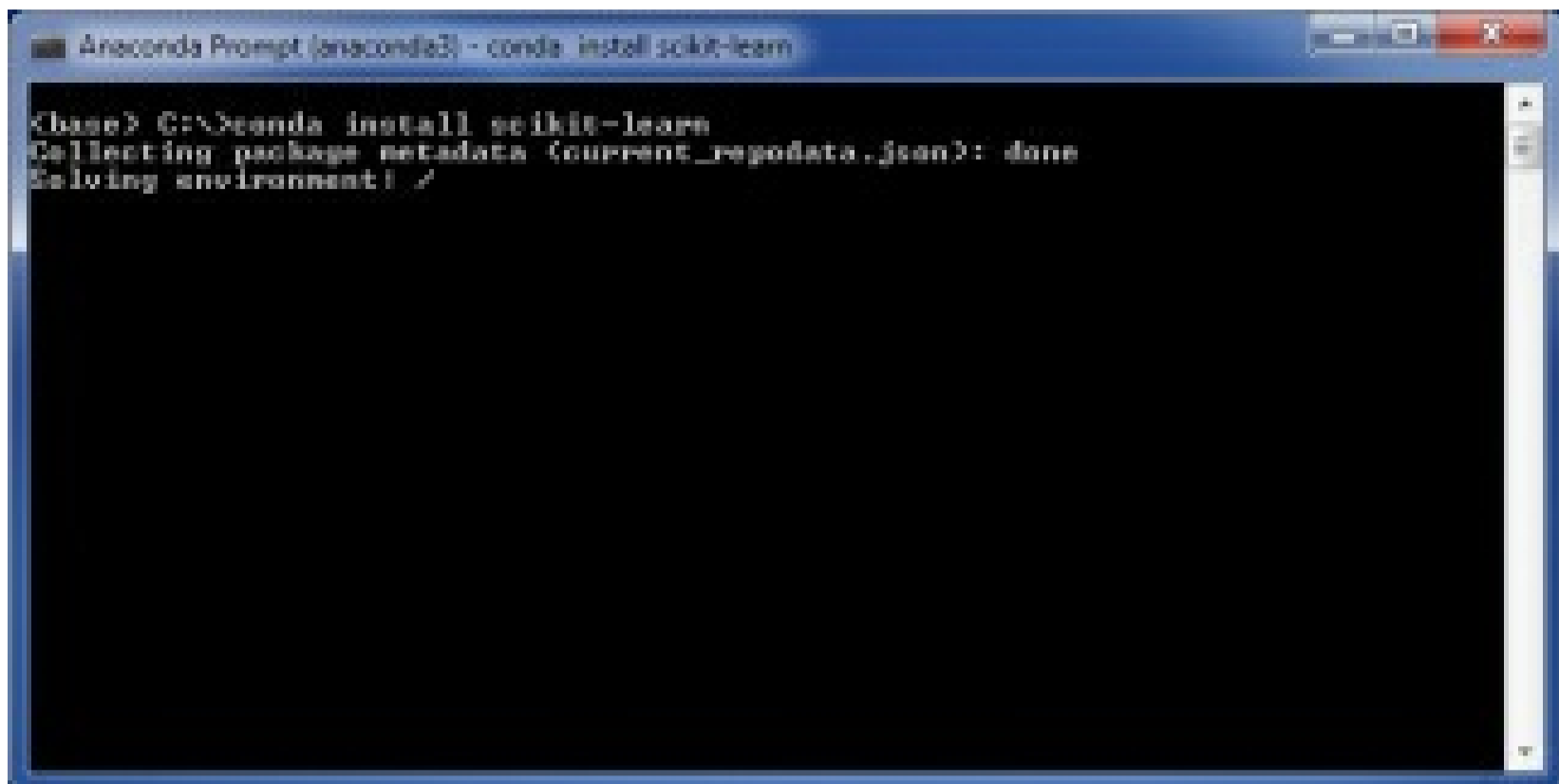
```
python -m pip show scikit-learn
```

This will bring up the version of Scikit-Learn (at the time of writing this book the version was 0.23.1) and where the library was installed.

Installing Scikit-Learn with Anaconda Conda Without a Virtual Environment

To install Scikit-Learn with Anaconda and ensure it is the latest version that is compatible with your Anaconda installation, run the **Anaconda Prompt** and type the following:

```
conda install scikit-learn
```



```

Anaconda Prompt (anaconda3) - conda: install scikit-learn

(base) C:\>conda install scikit-learn
Collecting package metadata (current_repodata.json): done
Solving environment: /

```

Allow the installation to complete. This can take several minutes.

Installing Scikit-Learn with Anaconda Conda in a Virtual Environment

If you have a Python environment, then it is best to create an environment to install and run Scikit-Learn for Anaconda that will not interfere with your Python installations.

To create a SciKit Learn environment from the **Anaconda Prompt**, do the following:

```
conda create -n sklearn-env
```

The next step is to activate the environment by typing the following into the **Anaconda Prompt**:

```
activate sklearn-env
```

Next, install Scikit-Learn into the Anaconda virtual environment. At the **Anaconda Prompt**, type the following:

```
conda install scikit-learn
```

Allow the installation to run through to completion.

Checking the Anaconda Conda Scikit-Learn Installation

To check the Scikit-Learn **conda** installation, type the following into the command line once the installation has completed successfully:

```
conda list scikit-learn
```

This will bring up the version of Scikit-Learn (at the time of writing this book the version was 0.21.1) and where the library was installed.

Installing TensorFlow

TensorFlow is a framework that is required for deep learning. It comes with a myriad of functionalities that allow a system to carry out deep learning functions. TensorFlow also has APIs that allow for the interaction between nearly all the most popular programming languages. The programming languages include Rust, Haskell, Java, C++, and Go.

As with most Python or Python compatible libraries, TensorFlow is free to download and use from the Internet as it is open source software. TensorFlow can be installed using either Python **pip** or Anaconda **conda** package installation programs. For the sake of this book the CPU-only version of TensorFlow is to be installed.

Installing TensorFlow With Python Pip Without a Virtual Environment

To install TensorFlow with Python and ensure it is the latest version that is compatible with your Python installation, open a **cmd** screen and type the following:

```
pip install -U tensorflow
```

The above command will help you install the **CPU** only version for TensorFlow. The **CPU** version of TensorFlow is faster and is a much easier system for beginners to work with. It is also recommended for the simpler machine learning models. When you are first starting out with machine learning model's, **CPU** is a lot easier to work with to design and train these types of models.

If you need to install a **GPU** version for TensorFlow, type the following command into the **cmd** command line:

```
pip install -U tensorflow-gpu
```

This will install the TensorFlow **GPU** version for your Windows system. **GPU** is a more advanced modelling system of TensorFlow and is used for huge amounts of data and graphics/images. It is also used to work with more complex tasks than **CPU** and can be quite a bit slower than **CPU**.

Installing TensorFlow With Python Pip with a Virtual Environment

If you created the sklearn-venv environment when you installed Scikit-Learn, then you should install TensorFlow into that environment. You can activate that environment by typing in the following at the **cmd** prompt:

```
sklearn-venv\Scripts\activate
```

Follow the same installation process for installing Scikit-Learn as you would when you are installing it without the virtual environment.

Installing TensorFlow With Anaconda Conda Without a Virtual Environment

To install TensorFlow with Anaconda and ensure it is the latest version that is compatible with your Anaconda installation, open the **Anaconda Prompt** screen and type the following:

```
conda install tensorflow
```

The above command will help you install the **CPU** only version for TensorFlow.

If you need to install a **GPU** version for TensorFlow, type the following command into the **Anaconda Prompt** command line:

```
conda install tensorflow-gpu
```

This will install the TensorFlow **GPU** version for your Windows system.

Installing TensorFlow With Anaconda Conda With a Virtual Environment

If you have already created a conda virtual environment for Scikit-Learn, then use the same environment. If you have not, follow the same procedure to create an Anaconda environment as shown in the “**Installing Scikit-Learn in Anaconda with a Virtual Environment**.” Once you have the Anaconda virtual environment, activate it using the following command:

```
activate sklearn-env
```

Install TensorFlow by typing in:

```
conda install tensorflow
```

This will install the TensorFlow CPU version for your Windows system.

Chapter 4:

Using Scikit-Learn

To start using Scikit-Learn you will first need to go into either Python or Anaconda to import the library for use. To import Scikit-learn type in the following at the command line prompt of either **cmd** or **Anaconda Prompt**:

For Python:

```
import sklearn
```

For Anaconda:

```
Python
```

```
Import sklearn
```

Allow the command to complete. You will know if it imported the module successfully as it will return to a new prompt line. If it was not imported correctly there will be an error message.

You can start creating machine learning models with Scikit-learn and use the libraries that are included with the package.

The Learning Problem

Machine Learning is based on a training set and a testing set. The training set is used to learn various properties from the data set. The testing set is used to test what was learned in the training set.

Learning problems in machine language have two categories, as already discussed in a previous chapter. Learning problems need to take into consideration various data sets of n samples. It uses these samples to try and predict the unknown data properties.

The two problem categories are classed as:

- **Supervised Learning**, when the machine algorithm's data has a few more attributes to it that need to have a predictive outcome. For instance, for a human it is when a child gets shown the color red and is told that it is red. Then they are shown a few other colors and are told they are not red. This classifies the problem as being either the color red or not the color red. Supervised learning problems are used to solve either:
 - **Classification problems**, problems that are known to have discrete output values. For instance, it is either the color red or it is not the color red; there is no other possibility or grey areas. Classification data will have the following elements to it:
 - **A predictor** or predictors. For example, based on an age group, predicting how many people enjoy vanilla ice-cream. In this case the age would be the predictor for the output
 - **A label** which is the output, and for a classification it is represented by an integer which could be 1, 2, -2, or 0. When you think of a label think of it as a yes or a no tag, because labels cannot be used to perform mathematical operations. Think about adding Yes, they like ice cream + No, they do not like ice cream together. It makes no sense to do so as they are simply labels.
 - **Regression** is when the machine learning algorithm needs to compare one or more continuous variables in order to predict an outcome. For instance, think of the average weight of a person that is 5 foot 3 inches. It could go one further to predict the weight of a person depending on their height and build. Regression will have one or more independent variables and a dependent variable. For the example above, weight would be the dependent variable which is predicted depending on the height and build of the person.
- **Unsupervised Learning** is when the machine learning algorithm is given a data set that consists only of functions and features. There is no target data set of example outcomes (labels). For example, the machine is only given input data and nothing to base its predictions on. Therefore the machine has to learn the basic structure of the data instead. Some unsupervised learning structures include:
 - **Clustering**, when a group of similar objects are grouped together in a cluster. Clustering is useful in applications such as statistical data analysis and data mining.
 - **Density estimation**, when an estimate is formed on observed data. Histograms, Parzen windows, and vector quantization are examples techniques of density estimation.

Loading Data Sets

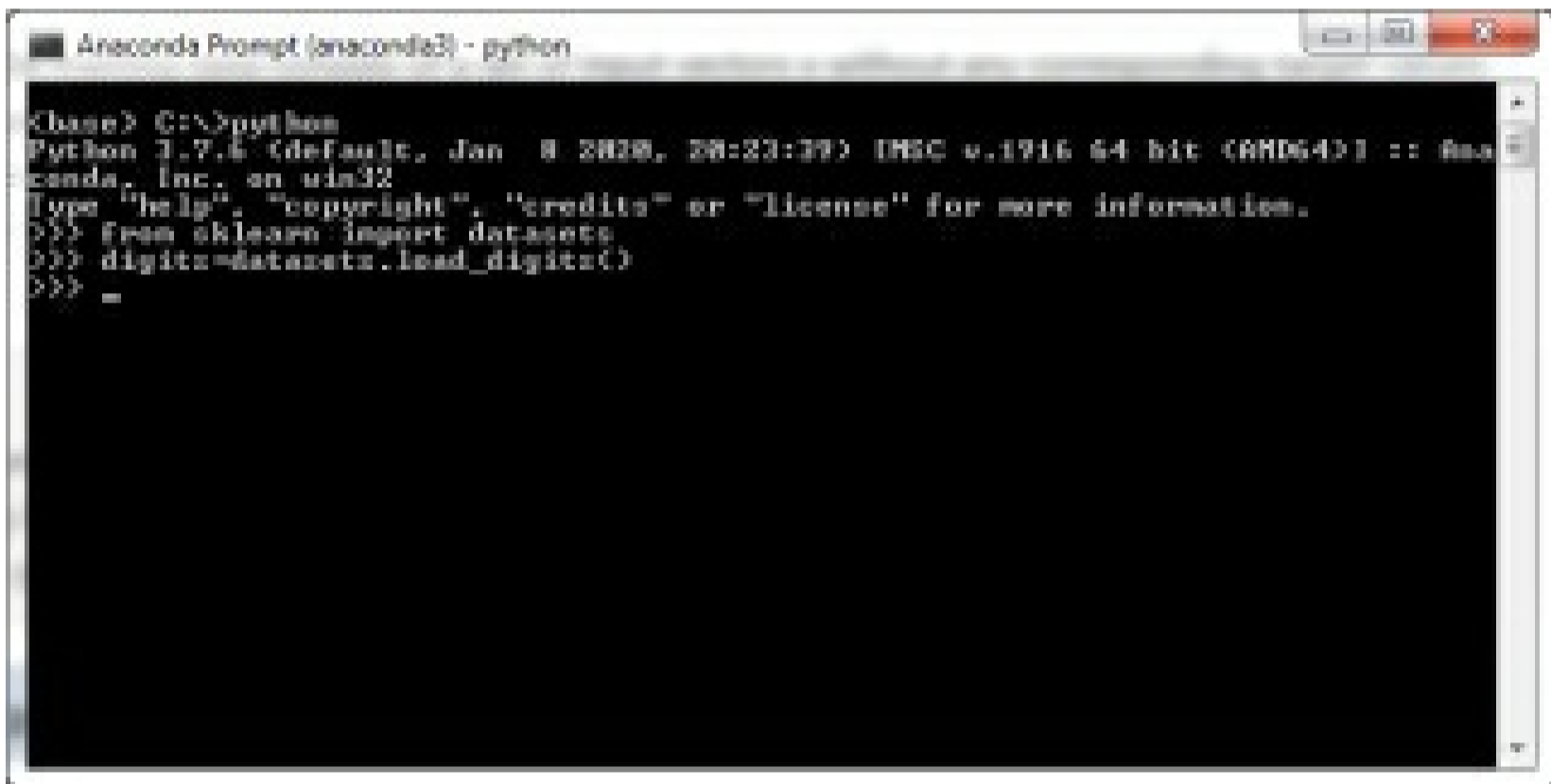
In order to give the ML algorithm something to work with it needs data sets to be loaded. To demonstrate how this is done, follow the command below to load a demonstration data set called iris. To load a Scikit-learn data set, type in the following at the command line prompt of either **cmd** (for Python) or **Anaconda Prompt**:

For Python:

```
from sklearn import datasets
digits = datasets.load_digits()
```

For Anaconda:

```
Python
from sklearn import datasets
digits = datasets.load_digits()
```



The data for data sets are usually stored in the `.data` member of the data set. This data set will contain information in the **n_features** and **n_samples** arrays that show what the data is, and the features that data has. The **.target** member is where some solutions to variables can be found.

For instance, if we had an array that contained the alphabet, all the letters would be stored **.data** member from a to z, whereas a combination of the most common words could be stored in the **.target** member. This would be the variation of words or word combinations.

The iris data set contains an array of numbers. To see what is in the data set, type the following at the command line prompt of either **cmd** (for Python) or **Anaconda Prompt**:

For Python:

```
print(digits.data)
```

For Anaconda:

```
Python
print(digits.data)
```

```
Anaconda Prompt (anaconda3) - python

C:\>python
Python 3.7.4 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from sklearn import datasets
>>> digits=datasets.load_digits()
>>> print(digits.data)
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 13.  1.  0.]]
>>> =
```

To see what some of the predicted outcomes for the arrays of numbers in iris are you can type at the command line prompt of either **cmd** (for Python) or **Anaconda Prompt**:

- For Python:**
digits.target
- For Anaconda:**
Python
digits.target

This will return a 2D array which will feature the **n_samples** array which correspond to the **n_features** array.

```
Anaconda Prompt (anaconda3) - python

C:\>python
Python 3.7.4 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from sklearn import datasets
>>> digits=datasets.load_digits()
>>> print(digits.data)
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 13.  1.  0.]]
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
>>> =
```

You can sort the data into a more usable format. The following example takes part of a target array and shapes it. At the command line prompt of either **cmd** (for Python) or **Anaconda Prompt** type:

- For Python:**
digits.images[2]
- For Anaconda:**
Python
digits.images[2]

```
Anacanda Prompt (anaconda3) - python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from sklearn import datasets
>>> digits=datasets.load_digits()
>>> print(digits.data)
[[ 0.  0.  5. ...  0.  0.  0.1
 [ 0.  0.  0. ... 10.  0.  0.1
 [ 0.  0.  0. ... 16.  7.  0.1
 ...
 [ 0.  0.  1. ...  6.  0.  0.1
 [ 0.  0.  2. ... 12.  0.  0.1
 [ 0.  0. 10. ... 12.  1.  0.11
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
>>> digits.images[2]
array([[ 0.,  0.,  0.,  4., 15., 12.,  0.,  0.1,
 [ 0.,  0.,  3., 16., 15., 14.,  0.,  0.1,
 [ 0.,  0.,  0., 17.,  0., 16.,  0.,  0.1,
 [ 0.,  0.,  1.,  6., 15., 11.,  0.,  0.1,
 [ 0.,  1.,  8., 13., 15.,  1.,  0.,  0.1,
 [ 0.,  9., 16., 16.,  5.,  0.,  0.,  0.1,
 [ 0.,  3., 13., 16., 16., 11.,  5.,  0.1,
 [ 0.,  0.,  0.,  3., 11., 16.,  9.,  0.1])
>>>
```

Regression

Machine learning regression models are used to predict continuous variables based on one or many predictor values. For example, predicting a person's weight based on their height and frame.

Linear regression

This is the most popular type of predictive analysis. Linear regression involves asking the following two things:

- Do the predictor variables forecast the results of an outcome variable accurately?
- Which particular variables are key predictors of the final variable, and in what standard does it impact the outcome variable?

Naming variables

The regression's dependent variable has many different names. Some names include outcome variable, criterion variable, and many others. The independent variable can be called an exogenous variable or repressor.

Functions of the regression analysis include:

- Trend Forecasting
- Determine the strength of predictors
- Predict an effect

Breaking down regression

Linear regression and multiple regression are the two basic states of regression. Linear regression contains an independent variable to be able to forecast the outcome of a dependent variable. To assist multiple regression when predicting a result, there are quite a few independent variables to lend a helping hand.

Regression is a useful tool among the financial and investment institutions. This is because it can be used to predict the sales of a particular product or company based on the previous sales and GDP growth among many other factors. A common regression model used in the finance sector is the capital pricing model.

The example below describes the formula used in linear and multiple regression.

Linear Regression: $Y = a + bX + u$

Multiple Regression: $Y = a + b_1X_1 + b_2X_2 + \dots + u$

This example is broken down as:

Y = the dependent variable, which is what needs to be predicted

X = the independent variable that uses the Y variable upon which to base the predicted outcome

a = the intercept value

b = the slope value

u = the regression residual value

Choosing the best regression model

Selecting the right linear regression model can be very difficult and confusing. Trying to model it with sample data

cannot make it easier. This section reviews some of the most popular statistical methods which one can use to choose models and challenges that you might come across. It also lists some practical advice to use to select the correct regression model.

It always begins with a researcher who would like to expand the relationship between the response variable and predictors. The research team that is accorded with the responsibility to perform investigation essentially measures a lot of variables but only has a few in the model. The analysts will make efforts to reduce the variables that are different and apply the ones which have an accurate relationship. As time moves on, the analysts continue to add more models.

Statistical methods to use to find the best regression model

If you want a great model in regression, then it is important to take into consideration the type of variables which you want to test as well as other variables which can affect the response.

Modified R-squared and Predicted R-squared

Your model should have higher modified and predicted R-squared values. The statistics shown below help eliminate critical issues which revolve around R-squared.

- The adjusted R-squared increases once a new term improves the model.
- Predicted R-squared belongs to the cross-validation that helps define the manner in which your model can generalize remaining data sets.

P-values for the Predictors

When it comes to regression, a low value of P denotes statistically significant terms. The term “reducing the model” refers to the process of factoring in all candidate predictors contained in a model.

Stepwise regression

This is an automated technique which can select important predictors found in the exploratory stages of creating a model.

Real World Challenges

There are different statistical approaches for choosing the best model. However, complications still exist.

- The best model happens when the variables are measured by the study.
- The sample data could be unusual because of the type of data collection method. A false positive and false negative process happens when you handle samples.
- If you deal with enough models, you’ll get variables that are significant but only correlated by chance.
- P-values can be different depending on the specific terms found in the model.
- Studies have discovered that the best subset regression and stepwise regression can’t select the correct model.

Finding the correct Regression Model

- **Theory**

Study research done by other experts and reference it in your model. It is important that before you start regression analysis, you should develop ideas about the most significant variables. Developing something based on outcomes from other people eases the process of collecting data.

- **Complexity**

You may think that complex problems need a complex model. Well, that is not the case, because studies show that even a simple model can provide an accurate prediction. Once there is a model with the same explanatory potential, the simplest model is likely to be a perfect choice. You just need to start with a simple model as you slowly advance the complexity of the model.

How to calculate the accuracy of the predictive model

There are different ways in which you can compute the accuracy of your model. Some of these methods are listed below:

- You divide the data sets into test and training data sets. Next, build the model based on the training set and apply the test set as a holdout sample to measure your trained model with the test data.
- Another method is to calculate the “Confusion Matrix” to the computer False Positive Rate and False Negative Rate. These measures will allow a person to choose whether to accept the model or not. If you consider the cost of the errors, it becomes a critical stage of your decision to reject or accept the

model.

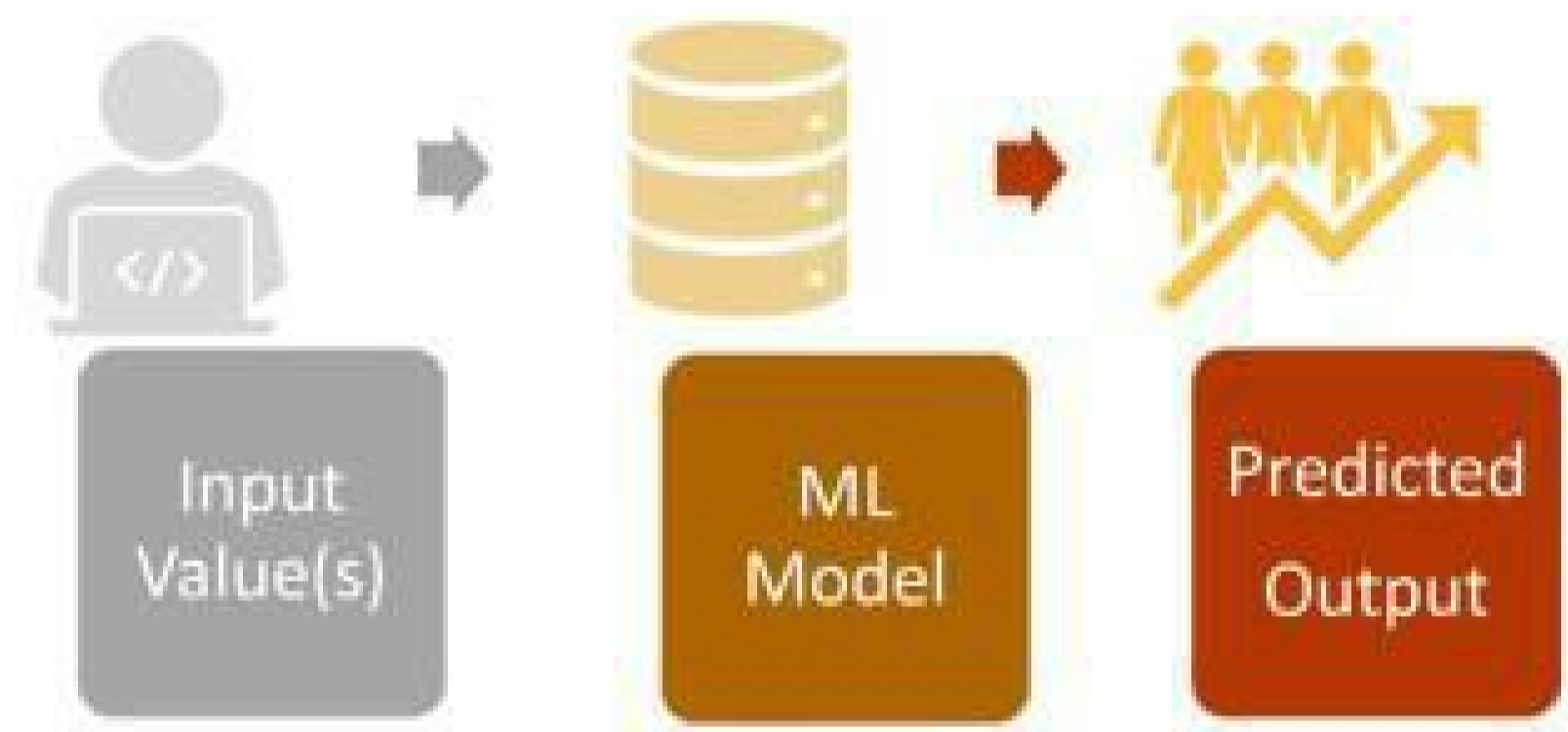
- Computing Receiver Operating Characteristic Curve (ROCC), the Lift Chart, or Area Under the Curve (AUC) are other methods that you can use to decide whether to reject or accept a model.

Chapter 5:

K-Nearest Neighbors (KNN) Algorithm

To build more complex classifiers, the KNN algorithm is the most popular. Having outperformed many powerful classifiers, it remains one of the most simple algorithms. The simplicity of the KNN algorithm is the reason why it is used in numerous applications of data compression, economic forecasting, and genetics. The algorithm can be used for solving both regression and classification problems. It is a supervised learning algorithm.

The KNN algorithm is the best place to start when you are learning machine language as it is an easy concept to grasp and is the basis for a lot of ML concepts.



Machine learning models use available data of past examples to learn from and then to make predicted outcomes based on certain input criteria. For instance, when you are teaching a child to learn, you teach them how to differentiate between two similar objects by their characteristics. An example would be teaching a child the difference between a chicken and a duck. You would show them a picture then add a few characteristics for each. These characteristics could include things such as:

- A chicken has feathers, claws, and a beak.
- A duck has feathers, a bill, and webbed feet.
- A chicken clucks.
- A duck quacks.

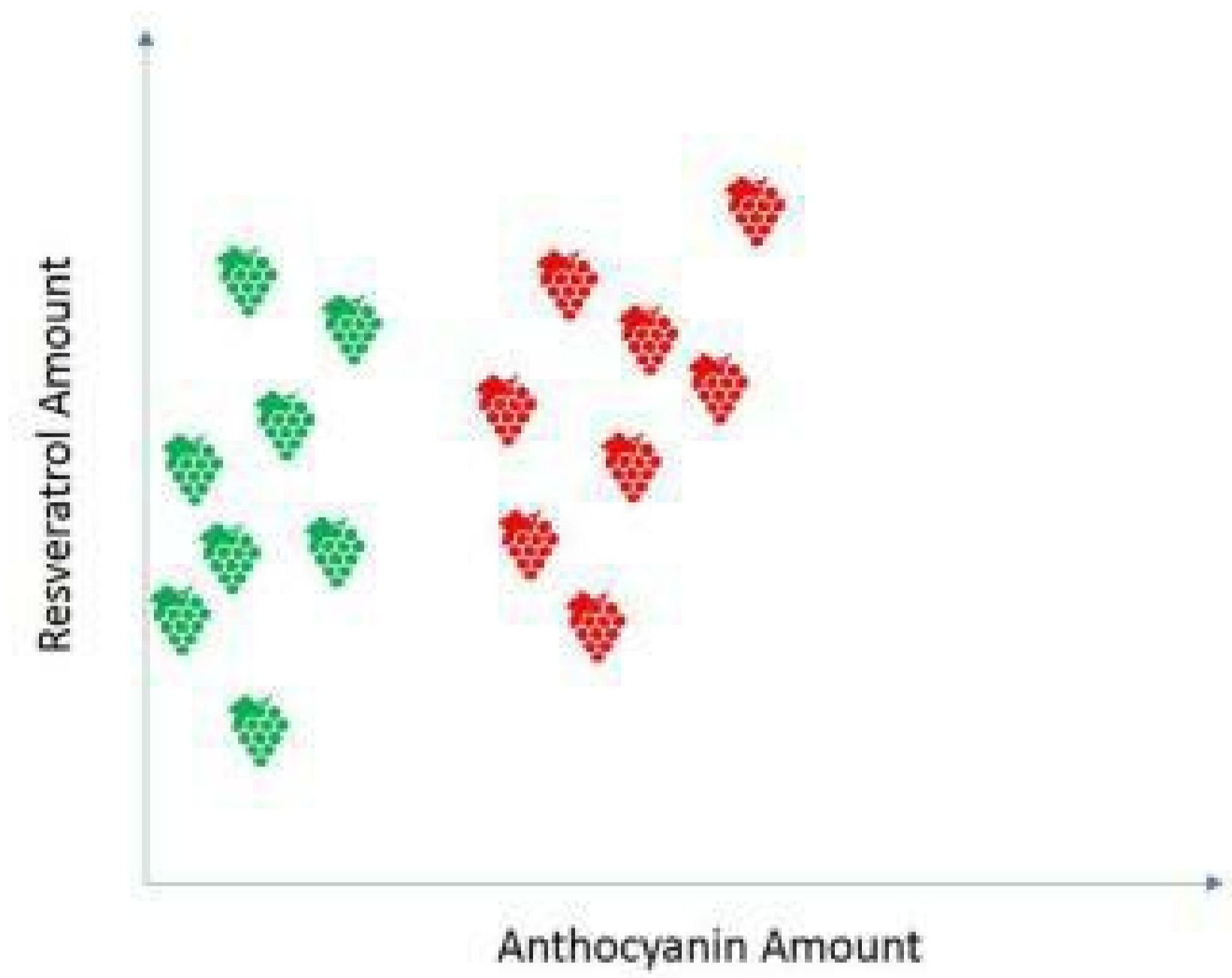
If you showed the child a picture of a duck, they would then be able to use those characteristics to determine that the picture is of a duck.

Similar to teaching a child how to differentiate and recognize objects, the KNN model data set is made up of training observations (x, y). Using information that has been input or made available for the ML model, the KNN model has to determine the relationship between x and y to predict an outcome. For instance, if x is given a value the algorithm needs to predict what the corresponding value for y should be. If you input the value quack (x), the KNN algorithm uses the list of characteristics available in its data sets to determine it should look at data for a duck (y).

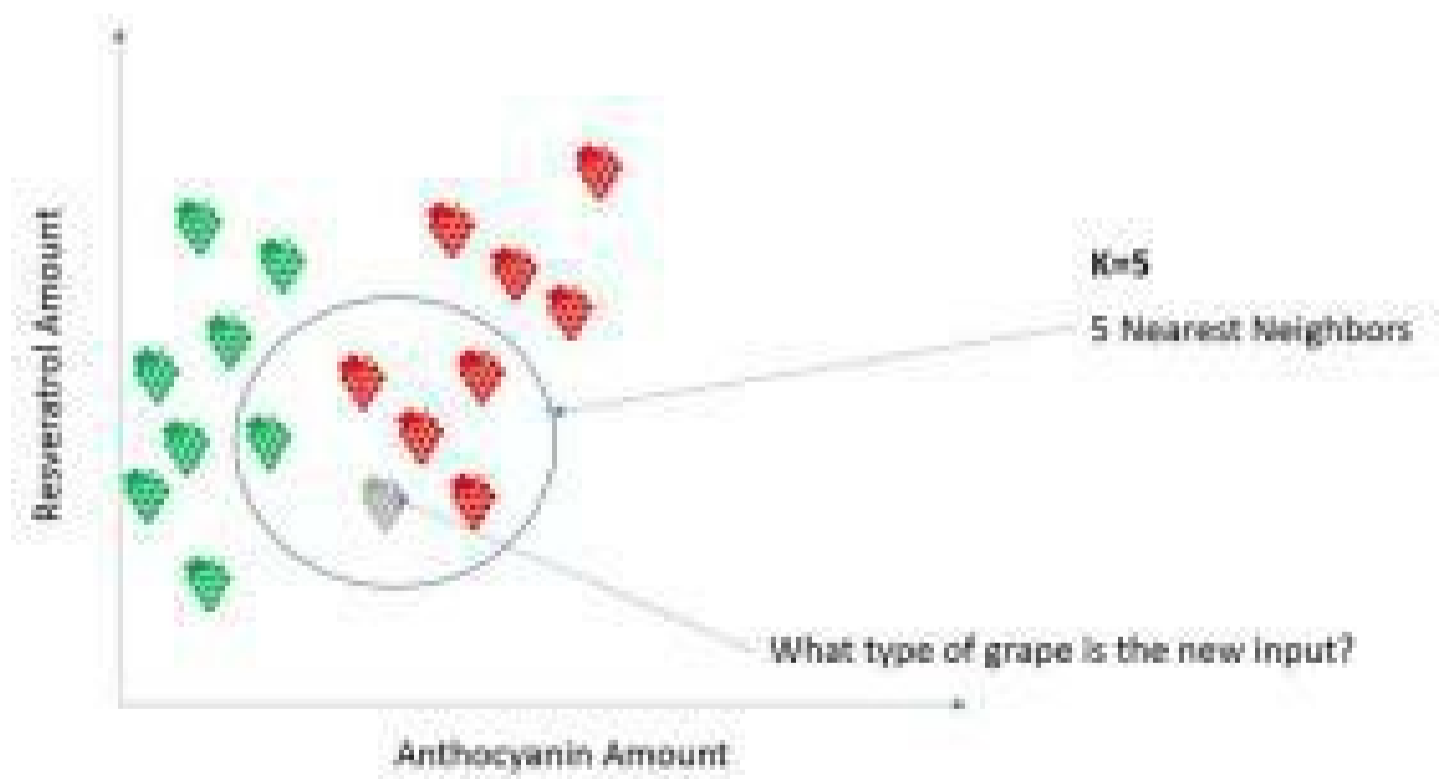
The KNN model will get to the predicted outcome of the “quack” being a duck because the model is based on feature similarities. It will group characteristics based on their similarities, which is why a KNN classifier can be used for classification of features. For instance, a duck and a chicken have feathers. So, they can be grouped as birds. However there are different species and subspecies of birds. While most birds have feathers, there are a few species that have webbed feet, and most water birds have them. But, only a duck quacks, has webbed feet, and feathers. Therefore the predicted outcome would be a duck based on **feature similarity**.

The KNN model is most commonly used to classify a data point. It does this based on the classification of its nearest neighbors. For example, is it a chicken or not? The KNN model builds on its current data set by classifying

new information based on its similarity to the current data set.



The above chart groups grapes or grape products by the amount of chemicals called resveratrol and anthocyanin found in them. These are two natural compounds usually found in grapes or grape products such as juices, wines, and supplements. The amount of these compounds in the different types of grapes is what distinguishes them from a red or white grape. The grapes are placed on the graph based on what amount of each of these compounds is found in the different variety of grapes or grape products.

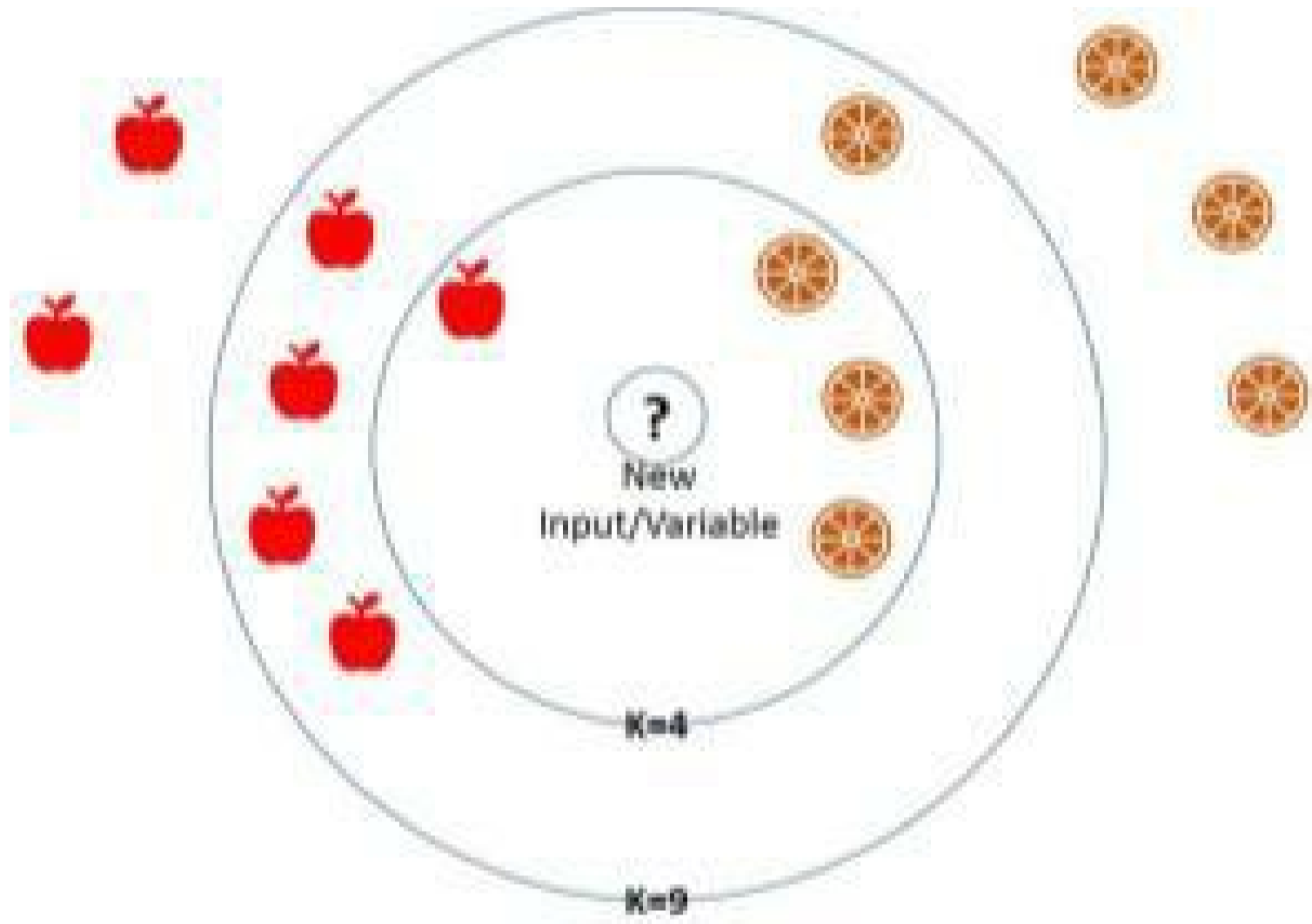


Using the features similarity process, KNN will set up a parameter which includes the 5 nearest neighbors (**K=5**), to determine a new type of grape (grey bunch). It is a majority voting system whereby the majority votes will determine the classification of the new data. The “k” in the KNN sets up the number of nearest neighbors that need to be included in the process of determining the new data point type. If you look at the image above, you will see that the grey bunch of grapes has more red neighbors than it does green ones. Therefore, the grey bunch of grapes will be classified as red because 4 of the 5 neighbors included in the k parameter are red.

How to Determine the “k” Parameter

Parameter tuning is what the process of determining the correct value for “k” in the KNN model is called. As KNN is a feature similarity-based algorithm, setting up the correct parameter for “k” is of the utmost importance to ensure predicted output is as accurate as possible.

The KNN model below determines whether or not a fruit is an apple or an orange.



The headache in choosing the correct “k” factor is trying to eliminate too much bias. For instance, in the diagram above, if you determine $k=4$, then the new variable would be an orange based on the 4 nearest neighbor votes. However, if you determined that $k=9$, the new variable would then be an apple. The trick to determining the “k” factor is to limit bias as much as possible.

How to Choose the Value of k?

If the “k” factor is too low, there is not going to be enough information and if it is too high, there is going to be too much data to process.

The easiest method for choose a “k” value:

- Use the square root of “n” where n = total number of data points.
- If the square root is an even number, either subtract or add 1 to the value to ensure that “k” values are odd values. If a “k” value is an even number, there is always the risk of failure. For example, the apples and oranges diagram could have ended up with an undetermined new value. The new value’s nearest 4 neighbors could have been 2 apples and 2 oranges instead of 2 apples and 3 oranges.
- Round the “k” value to the nearest whole number.
- For the diagram above with the apples and oranges, the “k” value would be:
 - $k = \text{Square Root}\{14\}$
 - $k = 3.74$
 - $k = 4$
 - $k = 4 + 1$
 - $k = 5$

When to Use KNN Models?

You can use the KNN algorithm when the following criteria is met:

- All data has a label — apples, oranges, grapes, etc.
- Data sets are small and not too complex — KNN doesn’t learn discriminative function from the

training sets of the model. It is known as a lazy learner and is used for data sets that are no more than about 1GB.

- Noise-free or clean data sets — This means that data must be clearly defined without too many variables. For instance, it is a dog or a cat, not a small dog, or small dog with long hair, or a cat with no fur, etc.

How the KNN Algorithm Works

A data set that determines if a person was **overweight** or **normal weight** would have two main variables:

- Height
- Weight

Consider the weight chart table below as the data set to determine if a person is overweight or normal weight by plotting their height variable and their weight variable.

Female		
Height (Feet & Inches)	Weight (Pounds)	Group/Class
4'9"	63 lb.	Normal Weight
4'10"	110 lb.	Overweight
4'11"	115 lb.	Overweight
5'0"	110 lb.	Normal Weight
5'1"	116 lb.	Normal Weight
5'2"	121 lb.	Normal Weight
5'3"	127 lb.	Normal Weight
5'4"	160 lb.	Overweight

Using the data from the table above along with the KNN algorithm, you have to determine if a person is overweight or normal weight using their height and weight as the criteria.



For instance, say Mary weighs 110 lbs. and is 5'1" tall. How can you determine if she is overweight or normal weight?

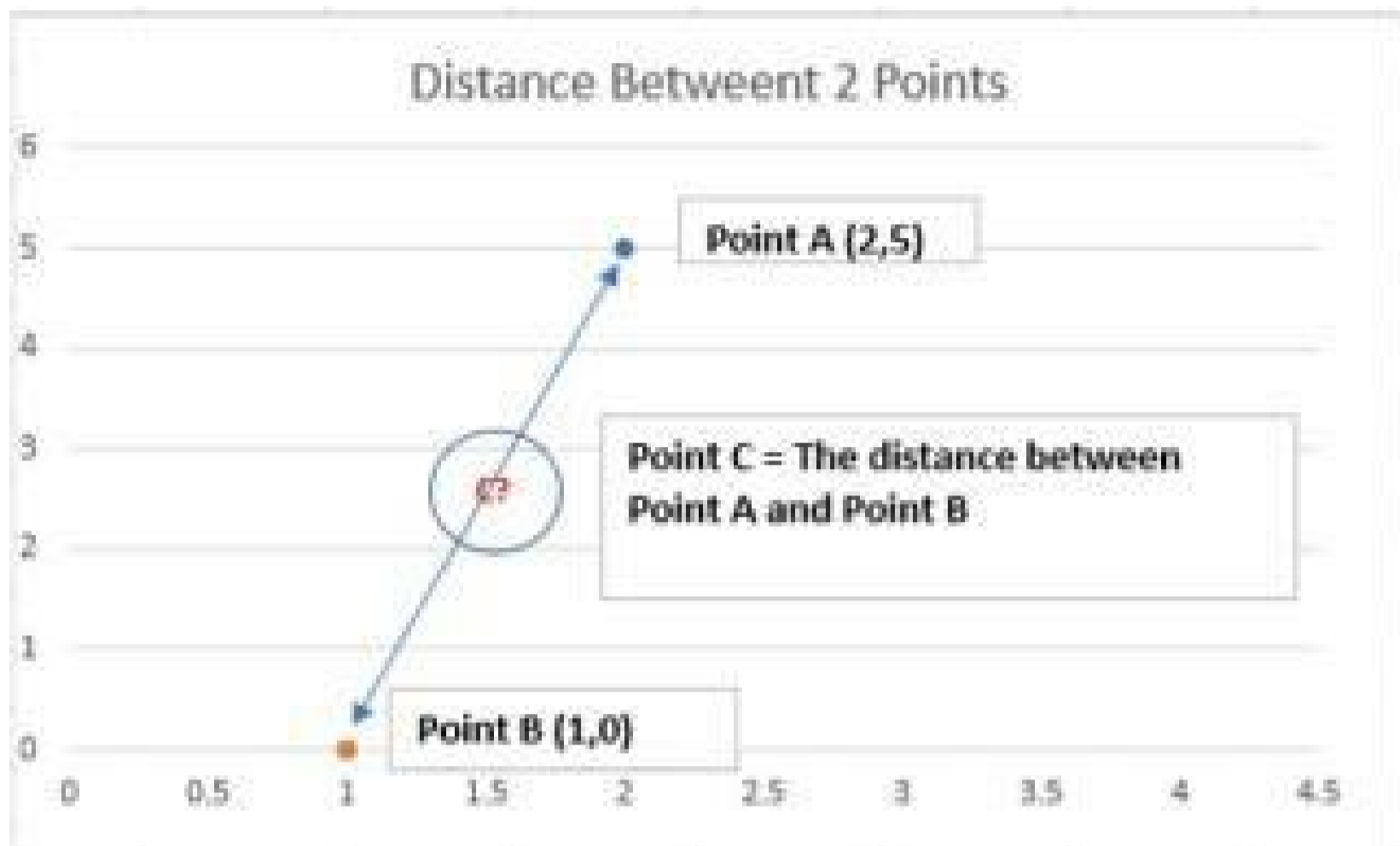
The KNN algorithm assumes that similar things are closer together. Therefore, in order for the algorithm to be true, similar data points should exist in close proximity to each other. For example, the classification of the green and red grapes used in an example above.

Using the assumption that similar data points are closely related, to find the nearest neighbors you would need to calculate the Euclidean distance, or what you may have learned in math as calculating the distance between two points.

The Distance Between Two Points/Euclidean Distance Calculation

When you were at school you would have learned to calculate the distance between two points. In a nutshell, this calculation assumes that when you know the vertical and horizontal positions of two points, you can calculate a straight line between these two points.

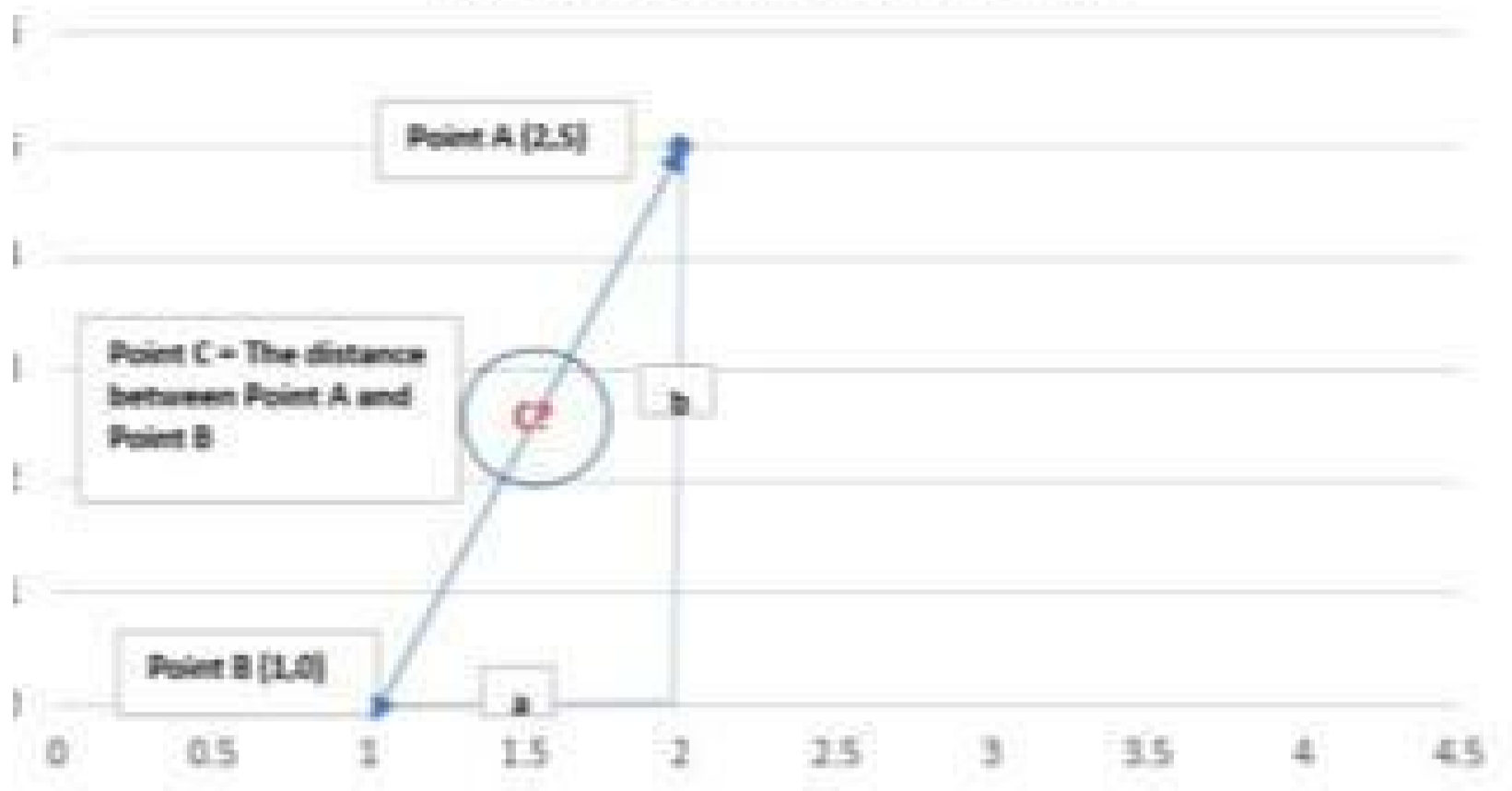
For everyone who needs a fast recap on the distance between two points calculation:



- Taking the graph above as an example, you would know the location of **Point A** and the location of **Point B** on the graph.
 - **Point A:** X = 2 and Y = 5 (2,5)
 - **Point B:** X = 1 and Y = 0 (1,0)
- What needs to be calculated is **Point C**, which is the **distance** between **Point A** and **Point B**.
- The calculation would look like the following formula:

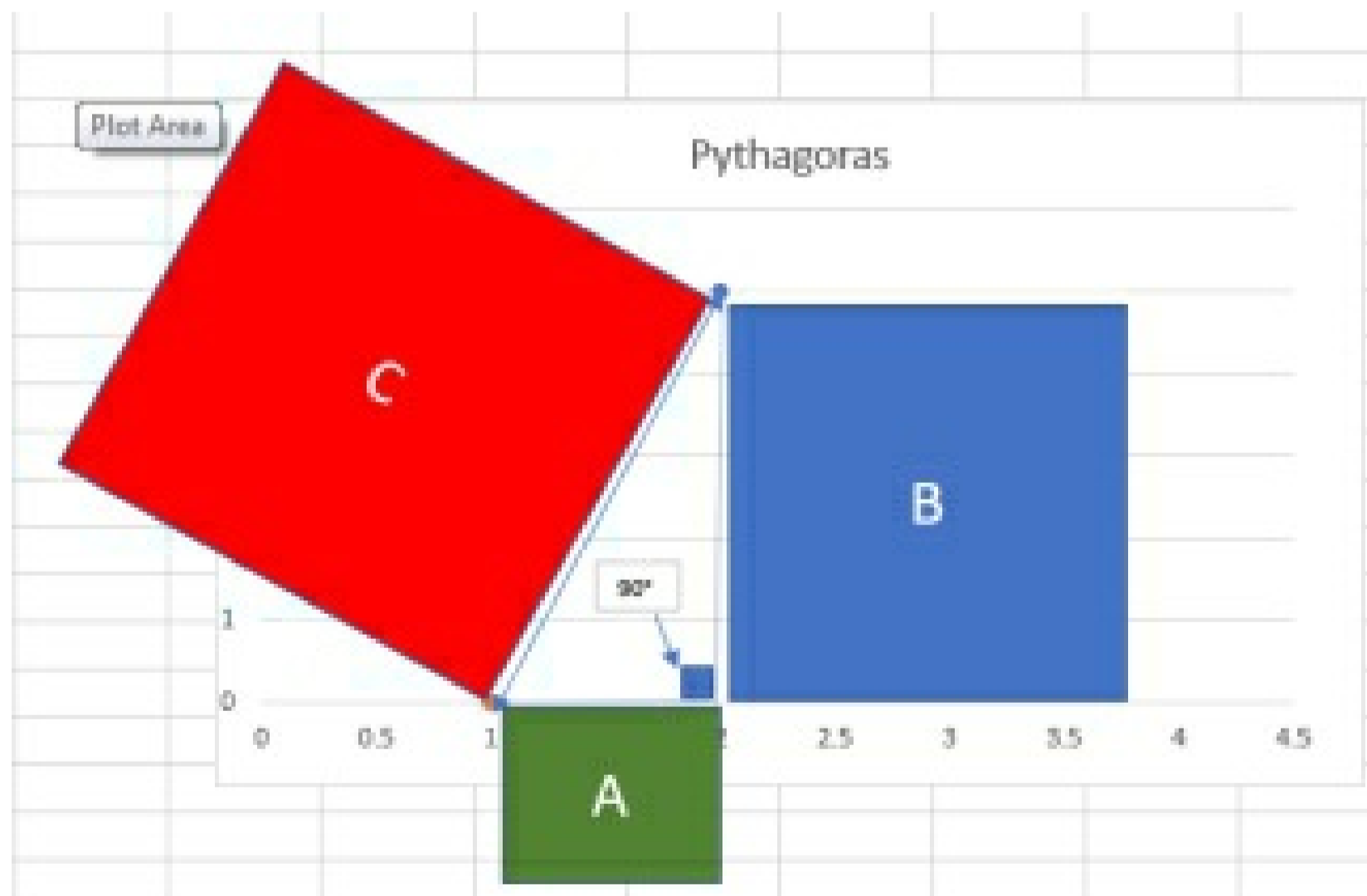
$$\text{Distance} = \sqrt{a^2 + b^2}$$

Distance Between 2 Points

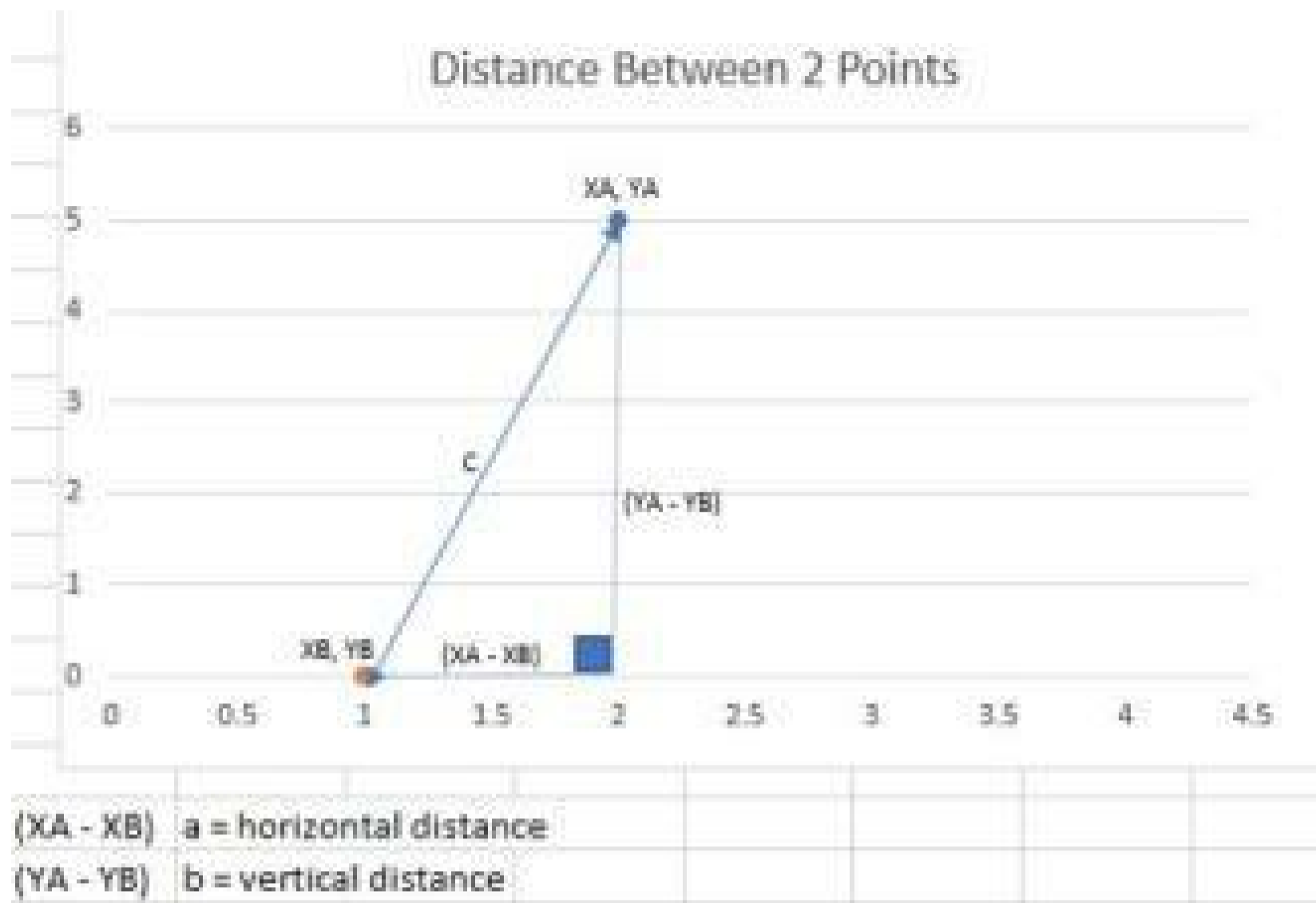


- To solve Point C you would need to plot a dotted line from Point A (b) to the x-axis and from Point B (a) to connect with the dotted line from Point A.
- These dotted lines form a right-angled triangle from the known points on the graph.
- When we form a right-angle triangle where the **Pythagoras theorem** states:

The biggest side of the triangle (the hypotenuse) is equal to the sum of the other two sides (“Pythagoras Theorem”, 2009).

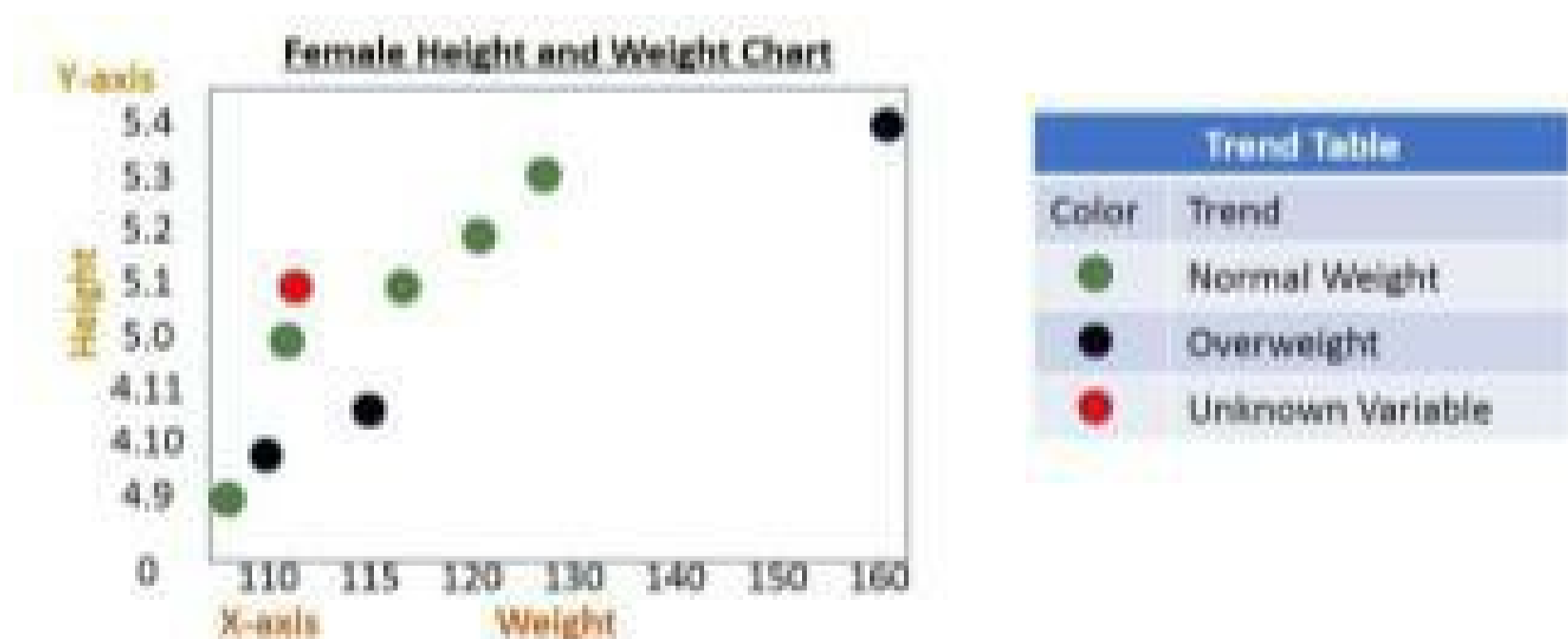


- The Pythagoras formula is:
 - $a^2 + b^2 = c^2$
- To calculate C:



- $a = (X_A - X_B)$ —Horizontal distance
- $b = (Y_A - Y_B)$ —Vertical distance
- Putting the formula together to calculate C:
 - $c^2 = a^2 + b^2$
 - $c^2 = (X_A - X_B)^2 + (Y_A - Y_B)^2$
 - $c = \text{Square Root}\{(X_A - X_B)^2 + (Y_A - Y_B)^2\}$
- Putting the values into the formula to calculate C:
 - $c = \text{Square Root}\{(2 - 1)^2 + (5 - 0)^2\}$
 - $c = \text{Square Root}\{1^2 + 5^2\}$
 - $c = \text{Square Root}\{1 + 25\}$
 - $c = \text{Square Root}\{26\}$
 - $c = 5.099$

KNN Distance Between Two Points/Euclidean Distance Calculation



To understand how to calculate the Euclidean distance for the KNN algorithm, take a look at the chart above. Using

the measurements from the previous height to weight chart, you can clearly see the weights that are normal and those that are overweight.

To determine if Mary is overweight or normal weight, the KNN algorithm will calculate the distance between each data point from the unknown data point within the existing data set.

Distance	Euclidean Distance Calculation Between the Unknown Data Point and Known Data Points
d1=	Square Root $\{[110 - 116]^2 + (5.1 - 5.1)^2\} = (-6)^2 + (0)^2 = 2 + 0 = 2$
d2=	Square Root $\{[110 - 121]^2 + (5.1 - 5.2)^2\} = (-11)^2 + (-0.1)^2 = 3 + 0 = 3$
d3=	Square Root $\{[110 - 127]^2 + (5.1 - 5.3)^2\} = (-17)^2 + (-0.2)^2 = 4 + 0 = 4$



Once the KNN algorithm has calculated the distance between the known data points in the data set, it then votes on how to classify the unknown data point by its closest neighbors based on $k=3$, which makes the unknown variable a normal weight. Thus, Mary is normal weight based on her height and current weight.

The Workings of the KNN Algorithm

The KNN algorithm works along the following procedure:

- Data is loaded into the algorithm
- K is then initialized to the required number of neighbors ($k=n$)
- Each data point query fed into the algorithm will:
 - Calculate the distance between the example data and the query
 - Create an ordered collection from the index and distance data
- Sort the data from the smallest entries to the largest by distance
- The K entries will be assigned labels
 - Regression models will return mean labels
 - Classification models will return mode labels

Implementing KNN Algorithm

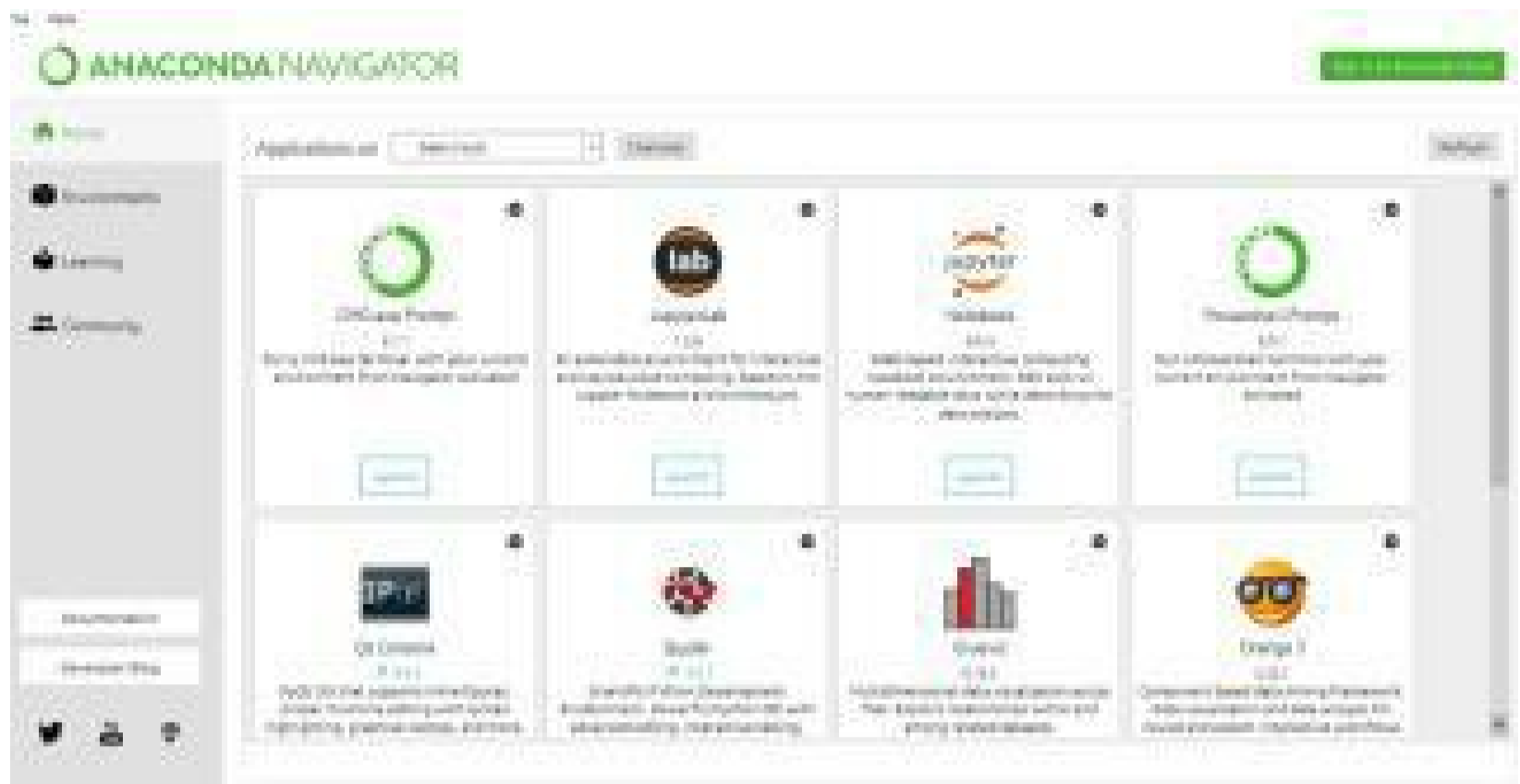
To demonstrate how to implement the KNN algorithm, you will be using a sample data set to predict diabetes.

The goal of the exercise is to predict if a person is at risk for diabetes.

The data set contains a list of 768 people. The list is a mixture of people who did or did not get diagnosed with diabetes.

The CSV file for this data set contains 9 columns of data all pertaining to information that helps the KNN algorithm predict whether or not a person is at risk for diabetes.

To get started, open either the **Python IDE** or **Jupyter Notebook for Anaconda**. For this book the Jupyter Notebook will be used. To access Jupyter Notebook you will need to launch the Anaconda Navigator.



Launch the Jupyter Notebook and allow it to open a browser window in your default browser. For the sake of this book the default browser is Chrome.

When the Jupyter Notebook has loaded in your default browser:

Set Up the Notebook Page

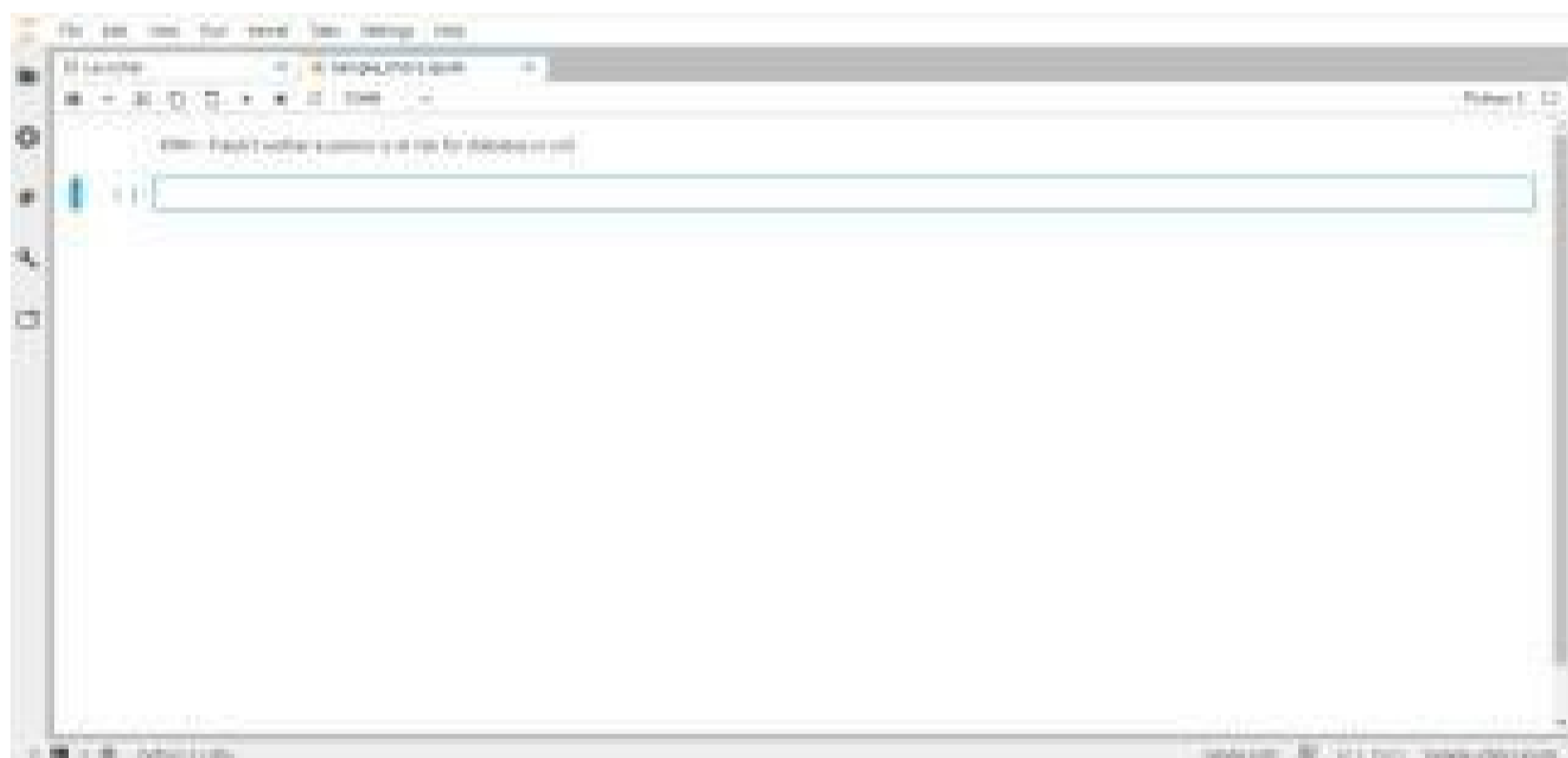
Change the name of the Notebook to “Sample_KNN1”

In the first line of the Notebook type the heading:

KNN - Predict whether a person is at risk for diabetes or not

Change the Cell from a Code cell to a Markdown language cell

Run the code



Import Tools and Set Up the KNN Environment

Create a new cell in which you are going to load or rather import all the libraries that we are going to need for this exercise.

Type the following into the new cell (do not type the # and the information preceding the #, as this is to explain what the command is for):

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

```

from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score

```



Explanation of the above code:

```

import pandas as pd #this will import the Pandas dataframe
import numpy as np #this will import the Numpy numbers array
#This space separates the tools needed to run the model and the setting up of the model.

from sklearn.model_selection import train_test_split #To split the model into training data and then testing the data

from sklearn.preprocessing import StandardScaler #This helps to scale down data that can grow or become too much, causing problems such as noise of KNN number bias.

from sklearn.neighbors import KNeighborsClassifier #This is the K Neighbor classifier tool that you are going to be using

from sklearn.metrics import confusion_matrix #Used for testing the model
from sklearn.metrics import f1_score #Used for testing the model
from sklearn.metrics import accuracy_score #Used for testing the model

```

Run the Code

Once you have finished typing in the commands, you will need to run the code.

Load the Database

The Diabetes.csv information can be obtained from the following website:

<https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/diabetes.csv>

Copy the diabetes.csv file to the \Anaconda3\Scripts directory

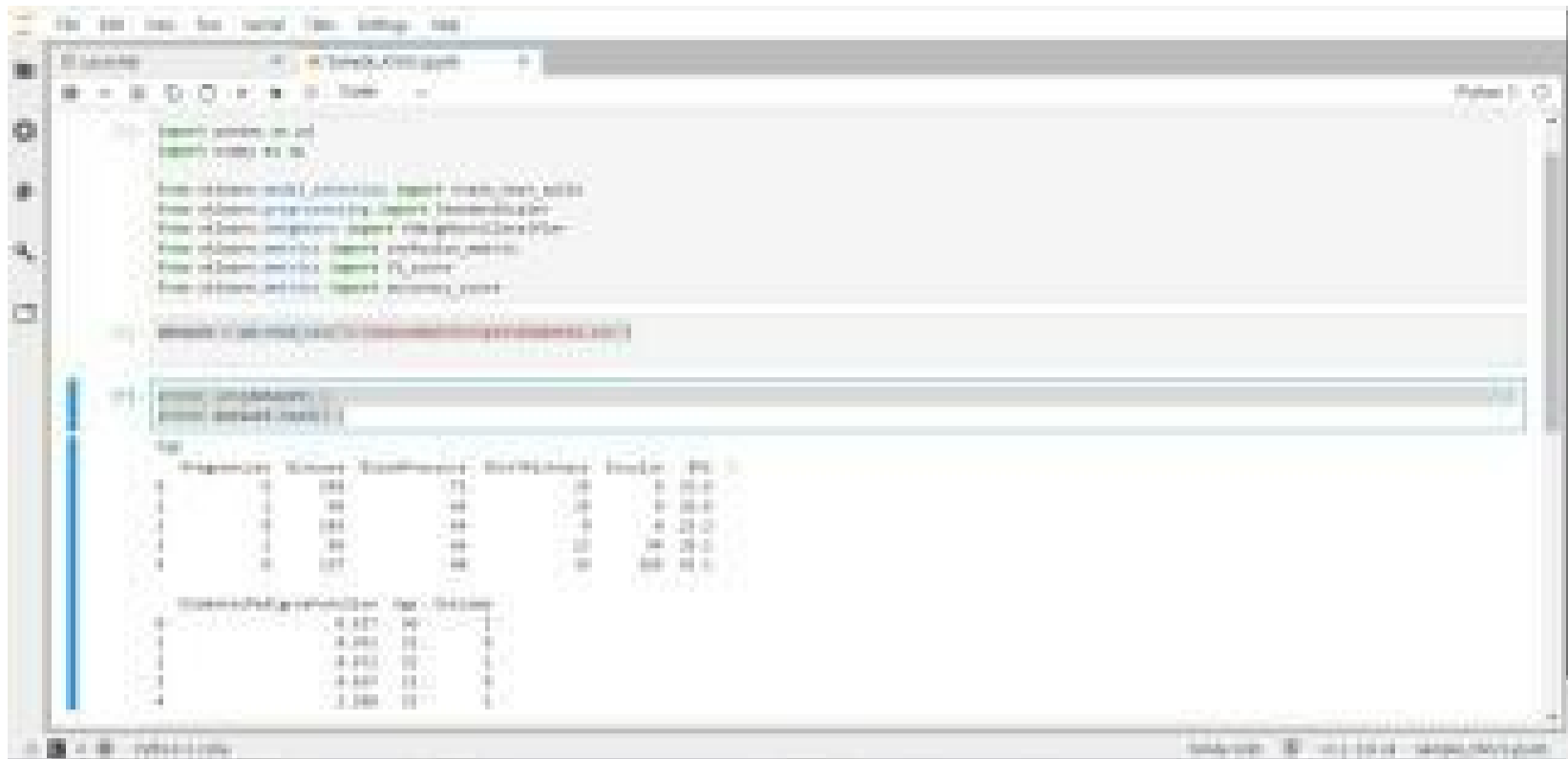
In a new cell in the Jupyter Notebook type the following:

```

dataset = pd.read_csv('C:\Anaconda3\Scripts\diabetes.csv')
print( len(dataset) )
print( dataset.head() )

```

Run the script and it will print out the first few lines of the diabetes data set.



Defining Certain Criteria

In this next step you are going to define certain criteria, such as not being able to have zero blood glucose levels. As you can see, the columns listed in the csv file from the previous printing of them are easier to work with for this section.

The first thing you are going to do in this section is define which columns cannot accept a zero. After you have defined that columns can accept a zero, you have to define how the model will replace any zeros. Zeros must be replaced with numpy.NaN which means that there is no data. Then you are going to set up the mean data which takes any non-data out of the data set for computation.

To set this criterion, in a new cell type the following:

```
zero_not_accepted = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI', 'Insulin']
```

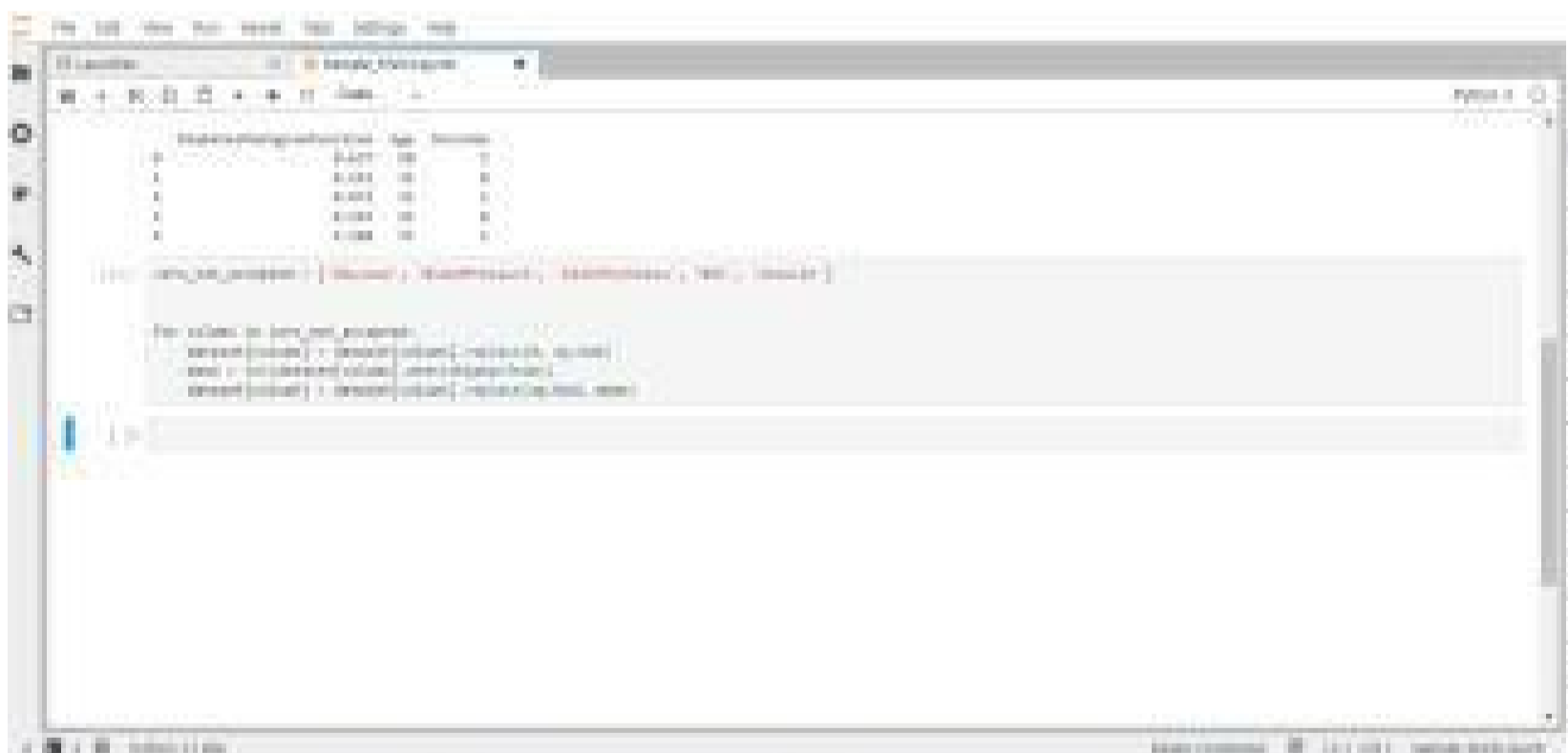
For column in zero_not_accepted:

```
dataset[column] = dataset[column].replace(0, np.NaN)
```

```
mean = int(dataset[column].mean(skipna=True))
```

```
dataset[column] = dataset[column].replace(np.NaN, mean)
```

Run the script. It should return you to a blank cell if there are no errors in the script.



Split the Data Set into Training and Testing

To split the data set you will need to tell the model which columns in the data set are training data, and which are the outcome or part of the answer. For the diabetes data set there are 9 columns, which to the model will be from 0:8 (: is to or if it is on its own : means all).

In the following commands you will see the first line as `x = dataset.iloc[:, 0:8]`, which is the **training data**. What this means is:

- : — means include all the rows
- 0:8 — means all the columns up until but not including column 9. The model starts count at zero.

The next line will be the outcome or part of the answer data. It will be written as `y = dataset.iloc[:, 8]`. What this means is:

- : — means include all the rows
- 8 — tells the model where to find the outcome

The last line of the code is simply setting up the `train_test_split`.

In a blank cell in the Jupyter Notebook type the following:

```
X = dataset.iloc[:, 0:8]
y = dataset.iloc[:, 8]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
```

Run the script and if there are no errors in the code it will return you to a blank cell.



Scaling the Features

When you loaded the sklearn features while setting up the environment to run the diabetes sample data set, you loaded the scaler. What this does is keep the data set within scale. For instance, instead of having data that is of different values in each column, you are going to set the scale between -1 and 1. This eliminates one column having data from 7 to 288, and the next column having data from 1 to 5.

To scale the data you are only going to fit it for the training data but ensure that the testing data still makes sense and processes the data correctly.

In a blank cell in the Jupyter Notebook, type the following:

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Run the script. Once the script has run without error it will return to a blank cell.



Building and Training the Diabetes Test Model

Now that you have set up all the data, it is time to build and then train the model. To fit the train data into the model you are going to have to use **KNeighborsClassifier** to define the model.

The first step is to define the model and initialize the **KNN Classifier**. There are a few things you need to be aware of when setting the classifier variable, and these are:

- `n_neighbors=n` — This is the “k” initializer that establishes the number of nearest neighbors to use. To get this number type the following into a blank cell in the Jupyter Notebook:

```
In [11]: len(y)
Out[11]: 768

In [12]: import math
         math.sqrt(len(y_test))
Out[12]: 27.71247385103399
```

`len(y)`

- You should end up with 768, which is the lines of data in the diabetes data set

`import math`

`math.sqrt(len(y_test))`

- You should end up with a number 12.409673645990857
- This is the number you are going to use as the “k” nearest neighbor number.
- If you round the number up to a whole number you will get 12. As you now should know, using an even number for voters is not ideal.
- We take away 1 in order to use a value of 11 instead to make sure the prediction is more accurate.
- `p=n` — This is the “power parameter” which establishes the metric that is going to be used. For instance, you want to know if a person is at risk for diabetes or not. This is a **Yes** or **No** scenario.
- `metric='Euclidean'` — This defines the metric you are going to use. There are a few metrics you can use for this, but the Euclidean one is the most commonly used.

Run the code and if there are no errors you should get returned to a blank cell.

```
In [13]: classifier = KNeighborsClassifier(n_neighbors=11, p=1, metric='euclidean')
```

You will need to fit the training data to the model. To do this, type the following into a blank cell into Jupyter Notebook:

`classifier.fit(X_train, y_train)`

Run the code and if there are no errors, your screen should now look similar to the example below.

```

In [31]: classifier.fit(X_train, y_train)

Out[31]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                             metric_params=None, n_jobs=None, n_neighbors=12, p=2,
                             weights='uniform')

```

The next step is to set the y_pred. To do this type the following into a blank Jupyter Notebook cell:

```

y_pred = classifier.predict(X_test)
y_pred

```

Run the code and if there are no errors, your screen should look similar to the example below.

```

In [32]: y_pred = classifier.predict(X_test)
y_pred

Out[32]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
               0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
               1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1,
               1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
               1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
               0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
dtype=int64

```

To evaluate the model you will be using the **confusion matrix**. In a blank Jupyter Notebook cell type the following:

```

classifier = KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')
print (cm)

```

Run the code and if there were no errors, your screen should look like the example below.

```

In [33]: cm = confusion_matrix(y_test, y_pred)
print (cm)

Out[33]:
[[94 13]
 [15 32]]

```

The confusion list matrix breaks down as follows:

- The 94 and 32 are predictions that were correct
- The 13 and 15 were predictions that were missed

Next you will print out the f1_score which measures the accuracy of the test for the model. To print the f1_score type the following code into a blank cell in the Jupyter Notebook:

```

print(f1_score(y_test, y_pred))

```

Run the code and if it runs without an error, it will return the following:

```

0.6956521739130436

```

This is to what fraction the test ran the f1 score, which would be 0.69.

Type the following code into a blank Jupyter Notebook cell to get the accuracy score for the test:

```

print(accuracy_score(y_test,y_pred))

```

Run the code and if it runs without an error, it will return the following:

```

0.8181818181818182

```

The test ran with an accuracy of 0.82

In data science, the f1 score is how many false positives there were. The accuracy score is what the model actually got right.

Chapter 6:

Using TensorFlow

To use TensorFlow you have to understand the concept of Deep Learning, which was touched on briefly at the beginning of this book. It is part of Machine Learning that mimics the functions of the human brain.

By using complex algorithms, deep learning models learn from unstructured data to train a neural network. A neural network typically consists of an input layer and an output layer with various hidden layers in between. Deep learning is where artificial intelligence begins. Any neural network with more than 3 hidden layers is known as a deep neural network.

The Layers of a Neural Network and Their Functions

- **Input Layer** — The input layer accepts various inputs such as large volumes of data or pixel layers of an image, for example. It passes the input on to the hidden layer(s) of the neural network.
- **Hidden Layer(s)** — The hidden layer(s) accepts the input data and processes it by performing complex algorithms, operations, and feature extraction. Hidden layers have weights and biases that are continuously updated during the training process. There are different neurons, each of which has multiple weights and one bias (variables).
- **Output Layer** — The output layer outputs the predicted outcome.

Deep Learning Libraries

There are many deep learning libraries that can be used for machine learning models. This book uses TensorFlow, which offers many APIs that can be used to work with TensorFlow.

At the beginning of this book you were introduced to what TensorFlow is. It was created by Google as an open source library that can be used for both traditional mesh learning and deep learning. When TensorFlow was first developed it was used to run numerical computations, mostly large ones.

It soon became apparent that TensorFlow was also good for deep learning. Data is input in multidimensional arrays. These arrays are known as tensors that are extremely good at handling very large amounts of data.

TensorFlow can work with CPUs and GPUs, as it works with Data Flow graphs. These graphs consist of edges and nodes. This makes it easier to use compacted data across a network.

Tensors

In TensorFlow, data is fed into the network as tensors, which are arrays of dissimilar data such as ranks or dimensions that become the input for a neural network. Tensors store data in such a way that it makes it easier for the data to be processed during computation.

The data is stored in a tensor, then fed into the hidden layers to be processed to form a desired output.

Tensor is divided up into:

- **Dimensions** — The Tensor dimension is how many elements there are for the tensor. You can get:
 - A single dimension tensor, which could be 5 rows by 1 column (5 x 1)
 - You can get a tensor that is 5 rows by 4 columns (5 x 4)
 - Multi-Dimensional tensors are 3 x 3 x 3.
- **Rank** — Tensor Ranks are the dimensions ranked based on the dimension of the tensor, for instance:
 - A tensor that only has one element in it has a rank of 0, and is known as a scalar or $s=[200]$
 - A single dimensional tensor which has a row or column of data in it has a rank of 1. It is a vector or $v=[10,11,12]$
 - A two-dimensional tensor which has more than one row or column of data in it has a rank of 2 and is known as a matrix or $m=[2,4,5],[6,7,8]$
 - A three-dimensional tensor has rank 3, $t=[2,4,5],[6,7,8],[9,10,11]$

There can be Ranks 4, 5, and higher depending on the dimension of the tensor(s).

Data Flow Graphs

Once the data, which is stored in tensors, is passed to the next layer, a computation process is run. This computation process is done as data flow graphs. TensorFlow models are created by preparing graphs and nodes to be executed during a session where the data used is taken from the data stored in the tensors.

The first step is to create the graph. During this step no data is actually being executed or used. It is not like traditional coding, as the graph is executed during a session.

For each TensorFlow object a Data Flow Graph is created to represent that object.

Mathematical computations in the Data Flow Graph are known as “**nodes**.”

An “**edge**” in the Data Flow Graph represents multidimensional arrays.

Tensorflow programs work by:

- Building a Data Flow Graph — This is where you write the code to create a computational graph.
- Creating a session — This is the session where you execute the code.
- Executing the Data Flow Graph — The graph is executed during the session.

TensorFlow Programming Elements

Writing TensorFlow models is not done in the normal way a person would code in Python or another programming language. It is a bit different and as such, there are a few basic programming elements you need to be aware of. These elements include:

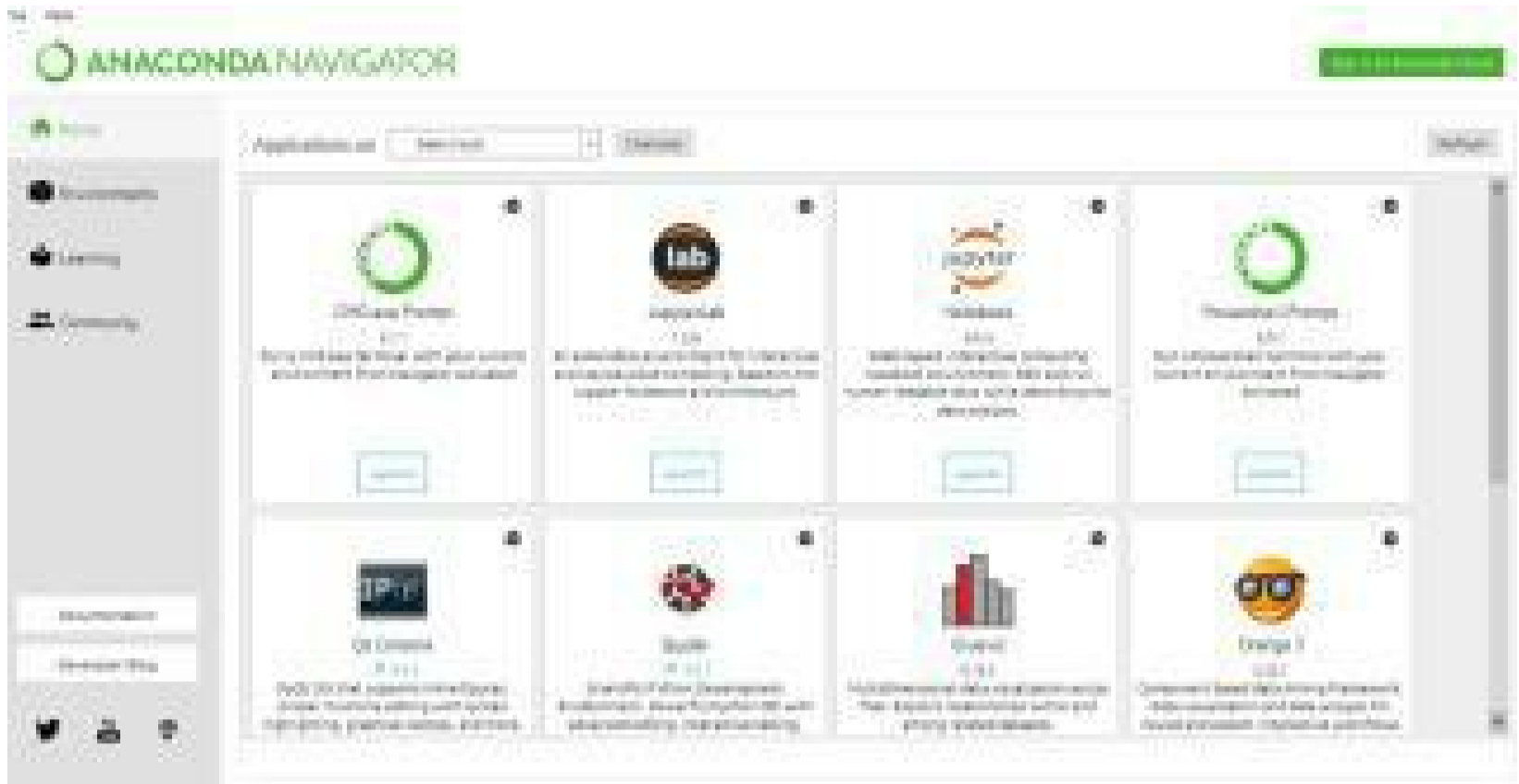
- **Constants** — Constants are similar to variables, but the parameter values of constants do not change. A constant is defined using the following:
 - `tf.constant()`
 - example: `a=tf.constant(3.0)`
- **Variables** — Variables are trainable parameters for a graph. Variables can change as their values are variable. A variable is defined using the following:
 - `tf.variable()`
 - example: `a=tf.Variable([.3])`
- **Placeholders** — Placeholders allow for data to be fed into tensors from an outside source such as a file. Place holders have to be fed by using the `feed_dict` parameter.
 - `tf.placeholder()`
 - example:
 - `a = tf.placeholder(tf.float32)`
 - `b = a*2`
 - `with tf.Session() as sess:`
 - `result = sess.run(b,feed_dict={a:3.0})`
- **Session** — Once the data has been assimilated and a Data Flow Graph has been created, a session is nuanced to evaluate the nodes, and this is known as **TensorFlow runtime**.

Every time you create a constant, variable, or placeholder, you are performing an operation or creating a node. These operations form what is called a node, and each of these nodes will run when you create a session.

Getting Started with TensorFlow

You can use either Python or Anaconda for the following TensorFlow exercises. Anaconda Jupyter Notebook has been used for the sake of this book.

Open the Anaconda Navigator.



Once the Navigator has opened in your default browser, launch the Jupyter Notebook. Depending on your system and default browser, loading the Navigator can take a few to several minutes to load.

Open a “New”, “Python 3” Notebook.

Name the Notebook TensorFlow 1

The first step is to load or rather import the TensorFlow libraries and modules. To do this, type the following into the blank Notebook cell:

```
import tensorflow as tf
```

Run the code and if there are no errors, you should get returned to a blank cell.

Working with TensorFlow Variables, Constants, Strings, Updates, Sessions, Placeholders, and Arrays

Create your first TensorFlow **Variable** by typing in the following in a blank Notebook cell:

```
first=tf.Variable(0)
```

Run the code and if it runs without errors you will be returned to a blank cell.

Create your first TensorFlow **constant** by typing in the following in a blank Notebook cell:

```
second=tf.Variable(1)
```

Run the code and if it runs without errors you will be returned to a blank cell.

Create a new variable by adding the variable and constant. You can do this by typing in the following in a blank Notebook cell:

```
new_value=tf.add(first,second)
```

Run the code and if it runs without errors you will be returned to a blank cell.

Update the first variable by assigning the new_value to it. You can do this with the “update” parameter. Do this by typing in the following in a blank Notebook cell:

```
update=tf.assign(first,new_value)
```

Run the code and if it runs without errors, you will be returned to a blank cell.

In the Graph Data if you have **Variables** they need to be **initialized** before you can start a session, or the session will terminate with an error. To initialize the variables in a graph, type the following into in a blank Notebook cell:

```
init_vbs=tf.global_variables_initializer()
```

Run the code and if it runs without errors, you will be returned to a blank cell.

Create your first **session** for Graph Data Flow. You do this by typing the following into in a blank Notebook cell:

```
sess=tf.Session()
sess.run(init_vbs)
print(sess.run(first))
```

Run the code and if it runs without errors the code will return a “0” before returning you to a blank cell. When you create a session it will stay open until you run “sess close” to close the current session.

To **print** out the “**update**” variable value, create a session by typing the following into a blank Notebook cell:

```
for _ in range(5):
    sess.run(update)
    print(sess.run(first))
```

Run the code and if it runs without errors, the code will return a list of numbers from 1 to 5 before returning you to a blank cell. This update operation will run the “for” operation five times, starting with the zero variable and adding one to it 5 times.



Working with **strings** is much the same as working with constants and variables in TensorFlow. If you wanted to string together the words “tooth” and “paste” to make the word “toothpaste” you could so by typing the following into a blank Notebook cell.

```
init_vbs=tf.global_variables_initializer()
```

Run the code and if it runs without errors you will be returned to a blank cell.



Working with **placeholders** can be a little confusing if you have not used them before. For the sake of this example you will be assigning a variable called “**a**” which will not have a value. If the variable does not have a value it is like having an empty cell in Excel that will be used later but still impacts another calculation somewhere in the spreadsheet. Thus, the variable “**a**” needs to be assigned a “**placeholder**.” These placeholders are usually defined as “**floating points**.” The next variable you will create for this exercise will be “**b**” which you want to be twice the value of “**a**”, whatever it may be. Then you will “**feed**” the placeholder for “**a**” some data and finally run the session to see the final outcome.

In a blank Notebook cell type the following:

```
a=tf.placeholder(tf.float32)
b=a*2
result=sess.run(b,feed_dict={a:3})
print(result)
```

Run the code and if it runs without errors it will return the number “**6.0**” before returning you to a blank cell.

```
In [40]: a=tf.placeholder(tf.float32)
        b=a*2
        result=sess.run(b,feed_dict={a:[3,5,7]})
        print(result)

6. 10. 14
```

To feed the dictionary (feed_dict) with a vector of rank, type the following in a blank Notebook cell:

```
a=tf.placeholder(tf.float32)
b=a*2
result=sess.run(b,feed_dict={a:[3,5,7]})
print(result)
```

Run the code and if it runs without errors it will return the numbers “[6. 10. 14.]” before returning you to a blank cell. This is an example of a single dimensional array in TensorFlow.

```
In [41]: result=sess.run(b,feed_dict={a:[3,5,7]})
        print(result)

[ 6. 10. 14.]
```

To feed the dictionary (feed_dict) with a multidimensional 3 x 3 x 3 array, type the following in a blank Notebook cell:

```
mda={a:[[[2,4,6],[8,10,12],[14,16,18]],[[1,3,5],[7,9,11],[13,15,17]],[[18,19,20],[21,22,23],
[24,25,26]]]}}
b=a*2
result=sess.run(b,feed_dict=mda)
print(result)
```

Run the code and if it runs without errors it will return with an array of numbers that consist of 3 blocks of numbers in 3 columns and 3 rows (3 x 3 x 3 array) before returning you to a blank cell. This is an example of a multidimensional array in TensorFlow.

```
In [42]: mda={a:[[[2,4,6],[8,10,12],[14,16,18]],[[1,3,5],[7,9,11],[13,15,17]],[[18,19,20],[21,22,23],
[24,25,26]]]}}
        b=a*2
        result=sess.run(b,feed_dict=mda)
        print(result)

[[ 4.  6. 12.]
 [16. 20. 24.]
 [28. 32. 36.]]

[[ 2.  6. 10.]
 [14. 18. 22.]
 [26. 30. 34.]]

[[18. 20. 22.]
 [42. 44. 46.]
 [48. 50. 52.]]]
```

To close a session in TensorFlow type the following into a blank Notebook cell:

```
sess.close()
```

Run the code and if it runs without errors it will return you to a blank Notebook cell.

Working TensorFlow Data Flow Graph and Programming Structure

Open a “New” “Python 3” Notebook Page in Jupyter Notebook.

Name the new page “TensorFlow 2.”

Type the following into a blank Notebook cell:

```
import tensorflow as tf
```

Run the code and if it runs without errors it will return you to a blank Notebook cell.

Creating the Default Graph

TensorFlow programs start off with defining the default graph. To do this type the following into a blank Notebook cell:

```
graph=tf.get_default_graph()
```

Run the code and if it runs without errors it will return you to a blank Notebook cell.

There will be no operations as yet for the graph as you have not written any. To find out what or if any operations have been performed for the current default graph, you can type the following into a blank Notebook cell:

```
graph.get_operations()
```

Run the code and if it runs without errors it will return a “[]” before returning you to a blank Notebook cell.

To see how the `graph.get_operations()` works, type the following into a blank Notebook cell:

```
d1=tf.constant(5,name='d1')
operations=graph.get_operations()
operations
```

Run the code and if it runs without errors, before it returns you to a blank Notebook cell it will list the following:

```
[<tf.Operation 'd1' type=Const>]
```

The above are the operations or nodes that have been performed in the default graph for this exercise. Every time you create an operation in the current graph when you run the `.get_operations` it will list all the current operations you have created in the current graph.

Create two more operations before you do the session execution by typing in the following in a blank Notebook cell:

```
d2=tf.constant(15,name='d2')
d3=tf.add(d1,d2,name='d3')
d4=tf.multiply(d2,d3,name='d4')
```

Run the code and if it runs without errors, it will return you to a blank Notebook cell

When you have created all the operations you want to run for the current graph, you will need to create and run a session to execute the graph. To do this, type the following into a blank Notebook cell:

Check the graph operations by typing in the following and running the code:

```
operations=graph.get_operations()
operations
```

This should run clear and return all the operations you have created which should be 4 (d1, d2, d3, d4).



In a blank Notebook cell type the following to create the session to execute the current default graph:

```
sess=tf.Session()
for op in graph.get_operations(): print(op.name)
```

Run the code and if it is error free it will run clear and return you to a blank Notebook cell.

End the current session by typing in:

```
sess.close()
```

Chapter 7:

Machine Learning and Neural Networks

TensorFlow is used for Deep Learning Neural networks models. Neural Networks were first developed in the late 1950s. They were modeled on the understanding of how the human brain works. Neural networks are a subset of Machine Learning and AI. They have been designed to mimic certain aspects of human neurons. Neural Networks pass data through interconnected nodes. The data is analyzed, classified, and then passed on to the next node, for further classification and categorization. Most Neural Networks usually contain two to three hidden layers which makes them Deep Learning Neural Networks. There are Deep Learning Neural Networks that contain hundreds of layers.

Neural Networks

There are many types of Neural Networks and these include the following:

Feedforward Neural Networks

Feedforward Neural Networks are the simplest form of Neural Networks. They work by way of data that flows through a feedforward network from the input, through to the node layers containing the neurons, and exits through the output layer. The data flow in one direction and unlike the more modern neural networks, feedforward networks do not cycle or iterate over the data. They perform a single operation on the input data and provide the solution in an output stream.

Single-Layer Perceptron

Single-Layer Perceptron is a form of a feedforward neural network with only one layer of nodes. It is the simplest form of a feedforward neural network. The input is sent to each node already weighted. The neuron either fires or does not fire based on the input, weight, and a set minimal threshold. The value is set as either activated or deactivated depending on if the criteria is met or not.

Multi-Layer Perceptron

Multi-layer perceptrons consist of two or more layers, with the output of the upper layer becoming the input of the lower layer. Because there are many layers, this form of neural network often takes advantage of backpropagation, where the produced output is compared with the expected output. The degree of error is fed back through the network to adjust the weights on the appropriate nodes, all with the intention of producing an output closer to the desired output state. It is these multi-layer algorithm networks with backpropagation that have become some of the most successful and powerful Machine Learning models.

Recurrent Neural Networks

Recurrent Neural Networks propagate data in both directions. Data flows forward like feedforward networks but also backward from later to earlier processing stages. Recurrent neural networks' main strength is the ability to remember previous states. Before recurrent neural networks, a neural network would forget previous tasks once it began a new one. Recurrent neural networks allow information to persist. This means they can take input from subsequent events and "remember" their experience in previous ones. Recurrent neural networks are a series of input and output networks with the output of the first becoming the input of the second. It follows that the output of the second series becomes the input of the third, and so on. This cycling allows Recurrent Neural Networks to develop closer and closer approximations to the desired output.

Chapter 8:

Machine Learning and Big Data

Big Data is the practice of dealing with large volumes of data. Although computer scientists have been dealing with Big Data for decades, the term Big Data comes from the 1990s. What sets Big Data apart from standard data sets is the sheer volume of the data that is collected, processed, learned from, and continues to grow exponentially.

Today the term Big Data brings with it a series of assumptions and practices that have made it a field all its own. Big Data is data containing more variety arriving in increasing volumes and with increasing velocity.

The 5 V's of Big Data

Volume

The term Big Data was coined in the early 2000s, when the amount of data available for analysis was too overwhelming for conventional systems to handle. Volume refers to the sheer amount of data produced these days with no signs of slowing down. Most systems analysts and infrastructure managers base their systems on data growth doubling every two years.

Velocity

Along with the rise in the amount of data being created, there has been an increase in the speed at which it is produced. Things like smartphones, RFID chips, and real-time facial recognition produce enormous amounts of data that is produced in real time. Real-time data is data that has to be dealt with as soon as it is created. The increasing speed of this data being produced puts a heavy strain on the capacity of a system's bandwidth, processing power, and storage space.

Variety

Data does not get produced in a single format. It is stored numerically in detailed databases, produced in structureless text and email documents, or stored digitally in streaming audio and video. There is stock market data, financial transactions, and so on, all of it uniquely structured. So not only must large amounts of data be handled very quickly, it is produced in many formats that require distinct methods of handling for each type.

Value

Data can be a valuable commodity if it can be extracted and used in a structured, useful way. The less structure a data set has, the more processing it needs to produce useful results for it. Conventional data processing tools are not able to process extremely large amounts of data, and this is where the value of Big Data comes in.

Veracity

Not all data that is captured is of value or can be used, especially when dealing with assumptions and predictions parsed out of large data sets. When dealing with large data sets, knowing the veracity of the data being used plays an important part in the output generated. Data veracity can be limited by factors such as data bias, errors introduced due to software bugs, data set omissions, questionable data sources, and human error. By limiting the amount of human interaction using machine learning and AI as much as possible allows for less error due to things such as wrong keystrokes, not reading data correctly, etc.

Big Data Uses

The point of dealing with all this data is to identify useful detail out of all the noise. Noise in data refers to data that is deemed repetitive or unnecessary. Valuable data allows businesses to find ways to reduce costs, increase speed and efficiency, design new products and brands, and make better, more informed decisions.

Some examples for current uses of Big Data include:

Product Development

Big Data can be used to predict customer demand. Using current and past products and services to classify key attributes, they can then model these attributes' relationships and their success in the market.

Predictive Maintenance

Buried in structured data are indices that can predict the mechanical failure of machine parts and systems. Year of manufacture make and model, and so on, provide a way to predict future breakdowns. This data, when correctly analyzed, can predict problems before they happen so maintenance can be deployed preemptively, reducing both cost and system downtime.

Customer Experience

Using Big Data, businesses can get a much clearer view of the customer experience by examining social media, website visit metrics, call logs, and any other recorded customer interaction to modify and improve the customer

experience. Big Data allows for better customer relations due to the amount and detail of the data companies can extract. By using Big Data to identify problematic issues, businesses can handle them quickly and effectively, thereby reducing the risk of negative customer experiences or press.

Fraud & Compliance

Big Data helps identify data patterns suggesting fraud or tampering. Aggregation of these large data sets makes regulatory reporting much faster.

Operation Efficiency

Big Data is currently providing the most value and return in the area of operation efficiency. Big Data helps companies analyze and assess production systems, examine customer feedback and product returns, as well as a host of other useful business factors. This in turn can help companies reduce outages, waste, and can help them to anticipate future demand and trends. Big Data can also be useful in assessing current decision-making processes and how well they function in meeting demand.

Innovation

Big Data's strength is in creating better relations between meaningful labels. For a large business, this can mean helping businesses, people, institutions, and other entities to predict market trends. Being able to understand the customer gives companies an edge in the marketplace to lead the way in innovation. Knowing if a product is valuable but could be more valuable with certain modifications due to customer feedback can help organizations get ahead of their competition. Innovation driven by Big Data is really only limited by the ingenuity and creativity of the people curating it.

Chapter 9:

Machine Learning Classification and Regression

Regression and classification are two types of machine learning outputs that can be produced. This is called predictive modelling, which is when a model is designed to make a new data prediction based on historical data.

Classification Problems

A classification problem involves an input that requires an output in the form of a label or category. Regression problems, on the other hand, involve an input that requires an output value as a quantity.

Classification problems are expected to produce an output that is a label or category. For instance, a function is created from an examination of the input data that produces a discrete output. A familiar example of a classification problem is whether a given email is spam or not spam.

Classification can involve probability, providing a probability estimate along with its classification. 0.7 spam, for example, suggests a 70% chance an existing email is spam. If this percentage meets or exceeds the acceptable level for a spam label, then the email is classified as spam and therefore the email needs to go into the spam folder. Otherwise, it is classified as not spam and will be delivered to the inbox.

One common method to determine a classification problem's algorithm probability of accuracy is to compare the results of its predictive model against the actual classification of the data set it has examined. On a data set of 5 emails, for example, where the algorithm has successfully classified 4 out of 5 of the emails in the set, the algorithm could be said to have an accuracy of 80%.

Regression Problems

In a regression problem, the expected output is in the form of an unlimited numerical range. The price of a used car is a good example. The input might be the year, color, mileage, condition, etc., and the expected output is a dollar value, such as \$4,500 - \$6,500. The skill set of a regression algorithm can be determined using various mathematical techniques. A common skill set measure for a regression algorithm is to calculate the root mean squared error, RMSE.

Although it is possible to modify each of the above methods to produce each others' result, such as turning a classification algorithm into a regression algorithm and vice versa, the output requirements of the two define each algorithm quite clearly:

- Classification algorithms produce a discrete category result which can be evaluated for accuracy, while regression algorithms cannot.
- Regression algorithms produce a ranging result and can be evaluated using root mean squared error, while classification algorithms cannot.

While machine learning employs both types of methods for problem-solving, both classification and regression, what method is employed for any particular problem depends on the nature of the problem and how the solution needs to be presented.

Chapter 10: Machine Learning and the Cloud



Cloud computing is a complex, and at times expensive, undertaking. It involves the integration of hundreds or even thousands of computer servers stored in large data centers that can be housed anywhere in the world. An example of a cloud computing system is Amazon Web Services (AWS) which has a data center that can host as many as 80,000 servers. This “raw iron” serves as the backbone of the cloud service. The servers operate virtual machine managers called hypervisors. Hypervisors can be either software or hardware.

The advantage of cloud computing goes to the consumer subscribing to the supplier of the cloud computing services. Companies like Amazon who offer cloud computing services absorb the costs of these large data centers so their customers don’t have to. The consumer gets to reap the benefit of a virtual networking environment that they can access from anywhere in the world instead of being limited to an office.

Cloud computing plays a valuable role in machine learning in that it can cut out the exorbitant costs of equipment. The machine learning process is an expensive process and gets even more so when you consider the equipment required to run it. Large corporations that have billion dollar yearly turnover struggle to keep up with the system demands ML can put on them. For institutes such as colleges or universities, this can be even more daunting. Cloud computing and its on the fly management of systems resources can drastically cut the cost of machine learning.

Companies using cloud computing to implement machine learning do not need an IT department capable of creating and managing the AI infrastructure either. This is taken care of by the companies offering the cloud services.

Just as cloud-based services offer SaaS (software as a service) solutions, machine learning cloud services offer SDKs (software developer kits) and APIs (application programming interfaces) to embed machine learning functions into applications. These connections offer support for most programming languages. This allows for developers to harness the power of machine learning processes directly in their applications, which is where the most value lies.

Benefits of Cloud-Based Machine Learning

Cloud-based machine learning projects make it easy for a company or organization to use limited levels of resources during experimentation and training of the model. It provides on the fly systems scaling that can increase or decrease processing power or storage capacity as and when needed. Although a lot of ML programming resources are open source, the scale of the processing and storage can be limiting cost-wise. Cloud-based machine learning systems make systems resources more obtainable and cost effective for both business and individuals.

Amazon AWS, Google Cloud Platform, and Microsoft Azure have many machine learning services that do not require knowledge of Artificial Intelligence, theories of machine learning, or even a data science team. Each of these platforms offer their own benefits, and which one you chose depends entirely on what your ML needs are.

Chapter 11: Machine Learning and the Internet of Things (IoT)



In the early 1980s a coke machine at Carnegie Mellon University was one of the first examples of the Internet of Things. By accessing the coke machine over the Internet, it could be determined if there was stock in the machine and what the temperature of the drink was.

The Internet of Things (IoT) has substantially evolved since the early 1980s and now includes devices such as smartphones, wireless sensors for security devices, GPS systems, and so on. IoT is a term that applies to any device that is connected to or through the Internet. These can be devices ranging from a jet engine to a robotic arm in a factory or a sensor in a thermostat.

Uses for the Internet of Things

The IoT can be classified as anything with an on or off switch that either works through or can work with the Internet. In today's fast past and ever evolving high-tech world, IoT plays a huge part in anyone who has the Internet, or is connected to, the Internet's life.

Uses for the IoT include:

Consumer Applications

Internet of Things devices created for consumer use range from smartphones to connected vehicles, health monitoring devices, and machine ordering devices. A lot of these devices have their software origins in programming methods, like those devised in Python.

Smart Homes

Home automation is where your smart house manages the house's resources for the occupant(s). These resources include lighting, fridge, air conditioning, security, media, and so on. In a lot of homes certain appliances such as refrigerators can organize part of the grocery list, security systems can alert owners of perimeter breaches to their smartphones and can log workers in and out of the property.

Elder and Disabled Care

Smart homes are able to take care of the elderly or disabled, making it easier to monitor them. It also allows the person in need of care a little more independence and 24/7 help and monitoring.

Healthcare Industry

Smart Healthcare is a fast-emerging trend where computers, the Internet, and artificial intelligence are merging to improve our quality of life. The Internet of Health Things (IoHT) is a specific branch of the Internet of Things. IoHT is designed for health and medically related purposes and is what is driving the digitization of the healthcare

system. The digitization of the healthcare system provides connectivity between properly equipped healthcare services and medical resources. IoHT also allows for the remote monitoring and in some cases operation of healthcare systems and notification systems.

Combined with machine learning, the health care IoHT ecosphere can improve a person's quality of life, guard against drug errors, encourage health and wellness, and respond to or even predict responses by emergency personnel.

Transportation Industry

The IoT assists various transportation systems in the integration of control, communications, and information processing. It can be applied throughout the entire transportation system — drivers, users, vehicles, and infrastructure. This integration allows for inter and even intra-vehicle communication, logistics and fleet management vehicle control, smart traffic control, electronic toll collection systems, smart parking, as well as safety and road assistance. By using various smart sensors, IoT applied to transportation can help companies that operate vehicle fleets avoid road accidents by assessing various anomalies in both the driver and vehicle.

Building and Home Automation

IoT devices can be used in any kind of building capable of monitoring and controlling the electrical, mechanical, and electronic systems within the building. Building and home automation are beneficial for better and more efficient management of energy, air control, and fire prevention.

Manufacturing Industry

Manufacturing equipment can be fitted with IoT devices for:

- Sensing
- Identification
- Communication
- Actuation monitoring
- Processing
- Networking

IoT in manufacturing allows for the seamless integration between manufacturing and control systems. It has changed the face of manufacturing in that companies are now able to:

- Manufacture new products a lot faster
- Reduce human error
- Increase productivity
- Reduce waste

The IoT can be applied to employ digital control systems for automating process controls, manage service information systems in the optimization of plant safety and security, as well as maintain and control operating tools.

Machine learning, when integrated into these networked systems, allows asset management and surveillance to maximize reliability by employing predictive maintenance and statistical evaluation. Machine learning allows IoT to maximize reliability through asset management using predictive maintenance, statistical evaluation, and measurements.

The manufacturing industry has its own version of the Internet of Things, the industrial Internet of Things (IIoT). Since its inception, it has changed the face of the manufacturing industry and had a huge positive impact on the manufacturing industry's bottom line.

Agriculture

The IoT has also proved useful in the agricultural industry, allowing farmers to collect useful data about humidity, wind speed and direction, temperature, soil composition, and pest infestations. The collection of this data when combined with machine learning algorithms allows farmers to automate farm techniques. These techniques include improving crop quality and quantity, minimizing risk, reducing crop maintenance, and minimizing waste. Farmers can monitor soil temperature and moisture, then use the data to determine more accurate times to water or fertilize fields.

Energy Management

There are not many electrical devices in today's world that cannot attach to the Internet. Smart Houses already have energy management functions, but it will not be long before most electrical devices that connect to the Internet will be able to communicate with a power source. This will allow for the power source and device to balance energy

generation with load usage and to optimize the energy consumption thereof. The Internet of Things allows for the remote control and scheduling of systems such as internal lighting, HVAC, ovens, and so on.

In order to successfully implement energy management on this scale, the system needs the power of machine learning to provide the predictive models. ML is necessary in energy management to anticipate and balance loads, aided by the constant flow of information from the Internet of Things devices located throughout the power grid. The IoT monitors for energy management systems to also provide service level, usage, peak hours, and other valuable information back to the energy supply companies. The information provided to the energy supply companies is also what allows for artificial intelligence systems to track and identify when components are reaching the end of life and need repair or replacement. This is critical in helping to eliminate surprise power outages.

Environmental Monitoring and Management

IoT can be used for environmental monitoring for the much-needed protection and improvement of the environment. With the power of ML behind strategically placed sensors, information such as air and water quality, soil and atmospheric conditions, wildlife through their habitats, seismic anomalies, and so on can be collected. IoT implemented across wide geographic areas allows for early warning systems for natural disasters like tsunamis, earthquakes, storms, and tornadoes. These early-warning systems are critical in aiding emergency evacuations and alerting the public. They are also crucial for helping emergency response services to be able to respond and offer more localized and effective aid. Machine learning, when interfaced with the Internet of Things, has enormous geographic scalability.

Smart Cities

The Internet of Things can be used for control and monitoring of both urban and rural infrastructure. IoT can monitor and control bridges, railways, and off and on-shore wind farms. It can be employed to monitor changes and events in structural conditions that might threaten safety or increase risks to the public.

IoT Security Concerns

Security is one of the main concerns for users of the conventional Internet, and security of the Internet of Things is a much-discussed topic. Many people are concerned that the industry is developing too rapidly and without an appropriate discussion about the security issues involved in these devices and their networks. The Internet of Things, in addition to the standard security concerns found on the Internet, has unique challenges. These challenges include security controls in industry, Internet of Things business processes, hybrid systems, and end nodes, as well as invasion of privacy issues as data can be easily traced, collected, and stored to be used for various means.

Security is likely the main concern over adopting Internet of Things tech. Cyber-attacks on components of this industry are likely to increase as the scale of IoT adoption increases. There is the real possibility of cyber threats becoming physical ones, as opposed to merely a virtual threat.

Current Internet of Things systems have many security vulnerabilities including a lack of encrypted communication between devices, weak authentication, especially where some devices are allowed to run in production environments with the default credentials implemented. The lack of verification or encryption of software updates, and even SQL injection, provide bad actors the ability to easily steal user credentials, intercept data, and collect Personally Identifiable Information that can be used in malevolent ways. It is also the easiest route to hackers injecting malware into updated firmware.

There are a lot of conspiracy theorists who keep on about “big brother spying on the world.” The thing is, some Internet-connected devices do spy on people in their own homes, and these devices are devices such as kitchen appliances, thermostats, security and computer cameras, and televisions. Many components of modern cars are susceptible to manipulation should a bad actor gain access to the vehicle’s onboard systems. Cars can have a bad actor manipulate components that include dashboard displays, the horn, heating or cooling systems, the hood and trunk releases, the engine, the door locks, and even the braking system. Vehicles with wireless connectivity are vulnerable to wireless remote attacks which means that they can be tampered with while the driver is using the vehicle.

Demonstration attacks on Internet-connected devices have been made to many devices including medical ones such as insulin pumps, implantable cardioverter defibrillators, and pacemakers. Because some of these devices have severe limitations on their size and processing power, they are usually unable to make use of standard security measures. These devices are incapable of using strong encryption methods for communication or even employing firewalls due to the nature of them.

Privacy concerns over the Internet of Things have two main thrusts:

Legitimate Uses

Governments and large corporations may set up massive IoT services which by their nature collect enormous amounts of data. To a private entity, this data can be monetized in many different ways, with little or no recourse

for the people whose lives and activities are swept up in the data collection.

For governments, massive data collection from the Internet of Things networks provide the data necessary to provide services and infrastructure, to save resources and reduce emission, and so on. At the same time, these systems will collect enormous amounts of data on citizens, including their locations, activities, shopping habits, travel, and so on.

To some, this is the realization of a true surveillance state. Without a legal framework in place to prevent governments from simply scooping up endless amounts of data to do with as they wish, it is difficult to refute this argument.

Illegitimate Uses

Illegitimate uses of the massive Internet of Things networks include everything from DDOS (distributed denial of service) attacks through malware attacks on one or more of the IoT devices on the network.

Security vulnerabilities in even one device on an Internet of Things network can, by the sheer nature of the devices full IoT communication capability, mean an infected device may not only provide its illegitimate host with the data it provides, but metadata of other devices in the network including the main system that houses all the information and processes.

In 2016, a DDOS attack powered by an Internet of Things device compromised by a malware infection led to over 300,000 device infections and brought down both a DNS provider and several major websites. This Mirai Botnet was able to single out for attack devices that consisted mostly of IP cameras, DVRs, printers, and routers.

While there are several initiatives being made to increase security in the Internet of Things marketplace, there are still the hurdles of government regulation and intergovernmental cooperation that need to be clearly defined and understood on a global scale in order to ensure public safety and privacy.

Chapter 12: Machine Learning and Robotics



Robots can be defined as machines, often programmable by a computer, that are able to carry out a series of complex actions without intervention. A robot can have its control systems embedded or be controlled by an external device. Unlike the popular movie stereotype, robots do not look like humans. Instead, they are usually designed to perform a task, and that requirement determines how they appear.

George Devol invented the first digitally operated and programmable called Unimate in 1954 (Unimate, n.d.). Unimate was bought by General Motors in 1961 for the purpose of lifting hot metal die castings, and it became the first mass produced robot. And like computers, robots have changed our society. Their strength, agility, and ability to continue to execute the same repetitive tasks perfectly are an enormous benefit to industry and society alike. And while they did cause some serious disruption to the manufacturing industries, putting many people out of work, their ascension in our societies has provided far more employment opportunities than they have taken.

There are several categories of robots which include:

Industrial or Service Robots

Industrial or service robots are probably the most popular or familiar types of robots. They appear in most automated factories and industries. They usually consist of an “arm” with one or more joints, which ends with a gripper or manipulating device. They first appeared in automobile factories such as General Motors (Unimate). They are fixed in one location and are unable to move about. Industrial robots are more commonly found in manufacturing and industrial locations. Service robots are basically the same in design as industrial robots but are found outside of manufacturing industries. Service robots can be found in shopping malls, banks, entertainment centers, or where there is a need for customer interaction.

Educational Robots

Educational robots are used as teacher aids or for educational purposes. As early as the 1980s, robots were introduced to children with the turtle robot controlled by LEGO. When combined with LEGO, educational robots moved beyond the robotic turtles and onto robot kits which became known as LEGO Mindstorm. These kits are used in classrooms all around the world for elementary and middle school students. Students can even compete in the LEGO League where they get to design, build, and program LEGO robots. These LEGO robots need to be able to perform certain functions in order to complete various challenges.

Robotics plays a vital role in the education of children for the twenty-first century and beyond. Kids that are incapable of communicating with the world around them are able to do so through robots. As the world is becoming more and more reliant on technology, robotics is a way to ensure children are able to understand the world they are living in.

Modular Robots

Modular robots consist of several independent units that work together. They can be identical or have one or more variations in design. Modular robots are able to attach together to form shapes that allow them to perform tasks. The programming of modular robotic systems is more complex than a single robot, but ongoing research in many universities and corporate settings is proving that this design approach is superior to single large robots for many types of applications. When combined with Swarm Intelligence (SI), modular robots are proving quite adept at creative problem-solving.

Collaborative Robots

Collaborative robots are designed to work with human beings. They are mostly industrial robots that include safety features to ensure they do not harm anyone as they go about their assigned tasks. An excellent example of this kind of collaborative robot is Baxter (Baxter, n.d.). Introduced in 2012, Baxter is an industrial robot designed to be programmed to accomplish simple tasks but is able to sense when it comes into contact with a human being and stops moving.

Robotic Learning

When robots are coupled with machine learning, researchers use the term “Robotic Learning.” This field has a contemporary impact in at least four important areas:

Vision

Machine learning has allowed robots to sense their environment visually and to make sense of what they are seeing. New items can be understood and classified without the need to program the robot ahead of time to recognize its environment.

Grasping

Coupled with vision, machine learning allows robots to manipulate items in their environment. This includes new items that the system may not have known before. Industrial or service robots would need to be reprogrammed in order to interact or recognize something new. With the introduction of machine learning, the robot comes equipped with the ability to navigate new item shapes and sizes automatically with no programming necessary.

Motion Control

With the aid of machine learning, robots are able to move about their environments and avoid obstacles in order to continue their assigned tasks.

Data

Robots are now able to understand patterns in data, both physical and logistical, and act accordingly.

Examples of Industrial Robots and Machine Learning

One example of the benefit of applying machine learning to robots is of an industrial robot which receives boxes of frozen food along a conveyor. Because it is frozen, these boxes often have frost, sometimes quite a lot of frost. This actually changes the shape of the box randomly. Thus, a traditionally-trained robot with very little tolerance for these shape changes would fail to grasp the boxes correctly. With machine learning algorithms, the robot is now able to adapt to different shapes, random as they are and in real time, and successfully grasp the boxes.

Another industrial example includes a factory with over 90,000 different parts. It would not be possible to teach a robot how to manipulate these many items. With machine learning, the robot is able to be fed images of new parts it will be dealing with, and it can determine its own method to manipulate them.

As more and more robots are combined with machine learning algorithms, industry becomes more reliant on them. There are millions of robots being used globally every day in nearly every sector of society.

Neural Networks with Scikit-learn

Neural networks are a machine learning framework that tries to mimic the way the natural biological neural networks operate. Humans have the capacity to identify patterns with a very high degree of accuracy. Any time you see a cow, you can immediately recognize that it is a cow. This also applies to when you see a goat. The reason is that you have learned over a period of time what a cow or a goat looks like and what differentiates the two.

Artificial neural networks refer to computation systems that try to imitate the capabilities of human learning. This is done through complex architecture that resembles the nervous system of a human being.

Chapter 13:

Machine Learning Models

There are many models of machine learning. These theoretical models describe the heuristics used to accomplish the ideal, allowing the machines to learn on their own.

Decision Tree

The decision tree technique is one of the most commonly used ML techniques. Either formally or informally, we decide on a single course of action from many possibilities based on previous experience. The possibilities look like branches, and we take one of them and reject the others.

The decision tree model gets its name from the shape created when its decision processes are drawn out graphically. A decision tree offers a great deal of flexibility in terms of what input values it can receive. The tree's outputs can take the form of a category, binary, or numerical data. The strength of decision trees is in how the degree of influence of different input variables can be determined by the level of decision nodes in which they are considered.

A big weakness of decision trees is the fact that every decision boundary is a forced binary split. There is no nuance. Each decision is either yes or no, one or zero. Moreover, the decision criteria can consider only a single variable at a time. There cannot be a combination of more than one input variable.

Decision trees cannot be updated incrementally, which means a trained training set cannot be used for new data. Instead, a new one has to be created to work with new training data.

Ensemble methods address many tree limitations. In essence, the ensemble method uses more than one tree to increase output accuracy. There are two main ensemble methods — bagging and boosting.

The bagging ensemble method, also known as Bootstrap Aggregation, is meant to reduce decision tree variance. The training data is broken up randomly into subsets, and each subset is used to train a decision tree. The results from all trees are averaged, providing a more robust predictive accuracy than any single tree on its own.

The boosting ensemble method resembles a multi-stage rocket. The main booster of a rocket supplies the vehicle with a large amount of inertia. When its fuel is spent, it detaches, and the second stage combines its acceleration to the inertia already imparted to the rocket and so on.

For decision trees, the first tree operates on the training data and produces its output. The next tree uses the earlier tree's output as its input. When the input is in error, the weighting it is given makes it more likely the next tree will identify and at least partially mitigate this error. The end result of the run is a strong learner emerging from a series of weaker learners.

Linear Regression

The premise of linear regression methods rests on the assumption that the output, a numeric value, may be expressed as a combination of the input variable set, which will also be a numeric value.

A simple example might look like this:

$$x = a_1y_1 + a_2y_2 + a_3y_3$$

Where x is the output, a_1 , a_2 , a_3 , and so on, these are the weights that are accorded to each input.

Y 's values would be the inputs.

A weakness of the linear regression model is the fact that it assumes linear input features, which might not be the case. Inputs must be tested mathematically for linearity.

K-Means Clustering Algorithm

K-means is an unsupervised machine learning algorithm for cluster analysis. It is an iterative, non-deterministic method. The algorithm operates on data sets using predefined clusters. Clusters are like categories; for example, if you were searching for shoes with the search term "Nike," the search would return all the pages containing Nike shoes. But the word "Nike" has more than one classification as it also pertains to clothes, and other items branded under the trade name. The K-Means clustering algorithm is used to group results with similar concepts. Thus, the algorithm will group all results that pertain to Nike shoes into one cluster, all results that pertain to Nike shirts in another, and so on.

K-Means Clustering Applications

K-Means Clustering algorithms are used by most web search engines to cluster web pages by similarity and to identify the relevance of search results. K-Mean clustering is a valuable tool in any application where unstructured data needs to be divided into meaningful categories.

Neural Network

As already covered in this book, the strength of neural networks is their ability to learn non-linear relationships between inputs and outputs.

Bayesian Network

Bayesian networks produce probabilistic relationships between outputs and inputs. This type of network requires all data to be binary. The strengths of the Bayesian network include high scalability and support for incremental learning. Bayesian Machine Learning models are particularly good at classification tasks such as detecting if an email is or is not spam.

Support Vector Machine

Support Vector Machine algorithms are supervised machine learning algorithms that can be used for classification and regression analysis. It is most often used for classification problems which the algorithm does by dividing categories using a hyperplane, which is a flat 3-dimensional sheet. It splits the data by category into separate dimensions and will continue to add dimension until there is no data overlap.

It is an incredibly powerful method for classifying data that is not without its issues. One of these issues is that looking at data once it has been mapped into higher dimensions isn't possible. The more it is split, the bigger chance there is of the data becoming gibberish. The process of SVM is not suited for very large data sets that are more complex and need more time to train. SVM is more suited to smaller, less complex data sets that do not take much training.

Machine Learning and Swarm Intelligence

Swarm Intelligence (SI) is defined as collaborative behavior, natural or artificial, of decentralized, self-organized systems. An ant colony or a “swarm” of autonomous mini-drones in a lab fall into the category of swarm intelligence.

In artificial intelligence, a swarm is typically a collection of agents that interact with each other and their environment. The inspiration for Swarm Intelligence comes from nature, from the collaboration of bees, or the flocking of birds, the gathering of herd animals, or any group of creatures with no centralized decision-making process but rather a collaborative one. These creatures appear to act intelligently even when no single individual one exhibits any signs of exceptional intelligence.

Swarm Behavior

One of the central tenets gleaned from swarm research has been the notion of emergent behavior. When a number of individuals are given simple rules for complex behavior, some behaviors seem to arise despite there being no rule or instruction to create them.

An artificial life program, called Boids, was created by Craig Reynolds in 1986, which simulated birds flocking together. In this program, each individual bird was given a simple set of rules to follow. When he let the birds free, his experimental birds behaved like a real bird flock. He soon discovered that he could add more rules to make more complex flocking behavior in the likes of goal seeking or obstacle avoidance.

Applications of Swarm Intelligence

Swarm Intelligence can be applied in many areas. Military applications include research into techniques for unmanned vehicle control. For NASA, swarm tech has been considered for the mapping of planets. In a 1992 paper, George Bekey and M. Anthony Lewis discussed using swarm intelligence in nanobots inserted into the human body to attack and destroy cancer tumors.

Ant-based Routing

In the telecommunication industry, the use of Swarm Intelligence has been researched using a routing table where small control packets, called ants, are rewarded for successfully traversing a route. Variations on this research include forwards, backward, and bi-directional rewards. Such systems are not repeatable because they behave randomly, so commercial uses have thus far proved elusive.

One promising application for Swarm Intelligence is wireless communication networks. In this case, the network relies on a limited number of locations that are expected to provide adequate coverage for users. Stochastic diffusion search (SDS) was modeled to address the problematic application areas of various ant-based swarm intelligence, with great success in some cases.

Airlines have experimented with ant-based Swarm Intelligence. Southwest Airlines uses software that employs swarm theory to manage its airlines on the ground. Each pilot acts like an “ant” in the swarm, discovering through experience what gate is best for him or her. This behavior turns out to be the best for the airline as well. The pilot colony uses the gates each one is familiar with to create more efficient arrival and departure times. This also offers the added bonus of data feedback to ensure there are limited to no gate backups, thus cutting down on delays.

Crowd Simulation

The movies are using Swarm Intelligence simulations for depicting animal and human crowds instead of trying to recreate real life ones. In *Batman Returns*, Swarm Intelligence was employed to create realistic bat simulations in the scenes where there were swarms of bats. For the *Lord of the Rings* movies, Swarm Intelligence simulations were employed to depict the massive battle scenes.

Human Swarms

When combined with mediating software, a network of distributed people can be organized into swarms by implementing closed-loop, real-time control systems. These systems, acting out in real-time, allow human actors to act in a unified manner, a collective intelligence that operates like a single mind, making predictions or answering questions. Testing in academic settings suggests these human swarms can out-perform individuals in a wide variety of real-world situations.

Swarm Intelligence and Machine Intelligence are both forms of artificial intelligence. There is much debate about SI being a subset of machine intelligence as it is a different approach toward the goal of smart machines. SI models the behavior of a particular kind of animal or animals, to achieve desired results. Thus, Swarm Intelligence and machine intelligence, even if one is not classified as a subset of the other, can complement each other. In an attempt to determine emotions from subtext text, a Swarm Intelligence approach will differ from a monolithic approach. Instead of one machine learning algorithm to detect emotion in text, a swarm approach might be to create many simple machine learning algorithms, each designed to detect a single emotion. These heuristics can be layered in hierarchies to avoid any emotion-detector fouling up the end result.

For example, take a machine learning swarm meant to detect emotion in written text while examining this sentence: “When I pulled back the covers of my bed this morning, a giant spider ran over my leg, and I immediately ran out of the bedroom screaming.”

This is a complex sentence and as such it is difficult for a natural language machine learning algorithm to parse for emotion. However, a swarm of simple machine learning algorithms dedicated to detecting one kind of emotion would likely have the terror algorithm scoring high, while fun and happiness scored low.

Another more difficult example would be the following sentence:

“I watched the game yesterday and saw us kick butt in the first period, but by the third, I was terrified we would lose.”

Human beings understand hyperbole. The writer is not terrified, but anxious the team will lose the game. Our swarm machine intelligence algorithms could have the fear or terror algorithm scoring high, but this would be inaccurate. Because swarm models can be hierarchical, one model’s output could be the input of another. In this case, a master model that detects emotion could filter through the outputs of each individual emotion algorithm, noting that “excitement” had been triggered by “kick butt,” parse that the subject of the sentence is a sport, and determine that anxiety is a better fit than terror.

If you take the above into account, then it seems only fair to define Swarm Intelligence as an application of machine learning with an extremely narrow focus.

Chapter 14:

Applications of Machine Learning

Most people in today's world are using real-world applications of machine learning in their everyday lives. Some of these applications are easily recognized, while others are hidden and operating in the background to make people's lives easier, more convenient, or safer. Though you might not recognize it, there are many technological advancements in your home that work through programming and training to become more accessible. These devices have the ability to learn and provide the basis for modern technology upon which mankind is very much reliant in the modern day.

Applications of machine learning in the everyday modern world include:

Virtual Personal Assistants

The application of virtual personal assistants (VPA) include Google Assistant, Apple's Siri, Amazon's Alexa, and Microsoft's Cortana. But these are only a few of the more widely used and well-known virtual assistants people interact with almost every day. They are also the most popular and well-known examples of applied machine learning systems in use in the modern world.

However, most personal assistants rely on more than one kind of machine learning technique in order to operate at their full potential. Most VPA's are either voice, motions, or need text entered to be activated. Thus, a person needs to first establish communication with the VPA using either a common writing script or voice in their language of choice. This establishes the connection or rapport with the machine which will answer back in the same language. Speech recognition is one of the machine learning skills these assistants use to understand you. Once the VPA has recognized the command and understood it, it will set in motion another machine learning technique to search for the desired answers being asked for. The last step in the process will be to respond with those answers and then track the response which the VPA will compare to previous responses. These answers are stored and used to assimilate more accurate ones in order to be more precise and efficient for any future requests.

Computing Predictions

Anyone that has used either Google or Apple maps has used a form of machine learning to help them arrive at a given destination. These apps in turn store locations, directions, and speeds in a central location to lead you to your destination, providing turn details and so on before you actually reach the turning point. At the same time, they are aggregating the data from all of the users nearby who are using their service to make more accurate predictions with regard to road travel information. This information includes tracking traffic congestion in order to offer a modified route to your destination in real-time to avoid delays to the journey. When there are not enough people using the app at any given time, the system relies on predictive machine learning from previous traffic data accumulated from a previous time the journey to the same location was taken. This way the ML can anticipate the traffic if it is unable to collect the data in real-time.

Rideshare companies such as Uber and Lyft are prime examples of companies that make use of machine learning applications. These applications are used to determine the cost of a ride, based on the user's current location and their destination based on various trends, previous customers, and data gathered about the route. These services would become technically impossible without machine learning crunching massive amounts of data, both historical and in real-time.

Online Fraud Detection

Securing cyberspace is one of the many goals of machine learning. Tracking online fraud is an important tool for achieving this goal. PayPal employs machine learning in its efforts to prevent money laundering. Their software can track the millions of transactions taking place and separate legitimate from illegitimate transactions going on between buyers and sellers.

Data Security

Another form of security in cyberspace is combating malware. This is a massive problem that continues to escalate and become more sophisticated. Machine learning is able to quickly search and find anomalies that point to malware in order to combat the issue before it becomes a problem. Machine learning can detect malware with a high degree of accuracy, and most new malware strains can be caught almost as quickly as they can be created and dealt with just as efficiently.

Machine learning is also very good at monitoring data access and finding anomalies, which might predict possible security breaches.

Personal Security

Everyone getting on a plane or entering a large event is screened to ensure the safety of the flight or event. Machine learning is starting to aid the manual checking of people, spotting anomalies human screenings might miss, and

helping to prevent false alarms. This help promises to speed up security screening in a substantial way, while at the same time ensuring safer events through the more powerful screening processes machine learning can provide.

What this means in real-life is the elimination of long slow queues in places like airports, large events, and so on.

Video Surveillance

The problem with video surveillance is that for it to really be 24hr and 365 days a year, it has to be constantly surveyed. Humans tend to get distracted, need bathrooms, coffee, and food breaks. They also doze off when they become bored or can miss things if something becomes too monotonous.

The benefit is machine learning never gets tired, or needs a break, or has its attention wander, so nothing is ever missed. Machine learning can focus on anomalous behavior like standing for a long time without moving, sleeping on benches, or stumbling, meaning human beings can be freed up to do what they do much better: deal with these people as required.

Social Media Services

Social media platforms utilize many forms of machine learning. Platforms such as Facebook, for example, constantly offer friend suggestions of “people you may know.” This is a simple concept of learning through experience. Facebook’s technology watches who your friends are as well as friends of those friends. It also logs what profiles you visit, how often, what articles you follow, and pages you visit. By aggregating this information, machine learning is able to predict people you are more likely to enjoy interacting with and so recommends them to you as friend suggestions.

The same sort of process is used when uploading a picture to Facebook of yourself and some friends. Facebook uses machine learning to identify those people in your photo by comparing them to images your friends have uploaded of themselves. These photos are scanned for the poses people are making, as well as any unique features in the background, geo-locating the image if it can.

Online Customer Support

Many websites offer a service to text chat with customer service while you browse their pages. While some companies do employ live chat support staff, there are those that use chatbots instead or a mixture of the two. Many companies cannot afford to pay a person to provide this service, so instead a chatbot will answer questions relying on details from the website to provide its answers. Machine learning means these chatbots get better over time, learning how to and how not to answer questions, and to seem more human in their interactions.

Search Engine Results, Refined

Search engine providers rely on machine learning to refine their responses to search terms. They monitor your activity after you’ve been given some results to examine. Do you click on the first link and stay on this page for a long time? Or do you navigate to page 2 or 3 or further down the results they have provided, clicking some links and immediately returning? This means the results returned did not satisfy your query as well as they could. Machine learning uses this feedback to improve the search algorithms used.

Product Recommendations and Market Personalization

Browsing online products is another way ML learns a person's online habits. You may have noticed that once you have browsed a product the Google adverts on a site may change to a similar one, or Facebook will suddenly insert random pages about the product while you are browsing the app.

As the search engine ML stores the products you browse, you will find similar products popping up on Twitter and even as spam or promotions in your email. Using the kind of information gathered from shopping or browsing habits, machine learning is used to tailor everything from specific email campaigns aimed at you personally, to similar product advertisements. It will start to offer up various coupons or deals by the retailers offering the product you’ve been looking at. As the amount of data for these machine learning algorithms grows, a person can expect this personal approach to advertising to become even more focused and accurate.

Financial Trading

Being able to “predict the market” is something of a holy grail in financial circles. The truth is humans have never been much better at this than chance. However, machine learning is bringing the holy grail of marketing prediction closer to a reality. Using the power of massive supercomputers crunching enormous quantities of data, large financial firms with proprietary predictive systems make a large volume of high-speed transactions.

With high speed and enough volume, even low-probability trades can end up making these firms enormous amounts of money.

Healthcare

Machine learning is revolutionizing the healthcare system. With its ability to ingest enormous amounts of data and

use that to spot patterns humans simply cannot see, machine learning is able to diagnose some diseases up to a year before a human diagnosis. At the same time, by crunching large data sets of populations, machine learning is able to identify groups or individuals who are more likely to need hospitalization due to diseases like diabetes. The predictive power of machine learning is likely the most fruitful avenue in healthcare since much of disease-fighting rests on early diagnosis.

Another inroad to healthcare by machine learning is robotic assistants in the operating room. While many of these robots are simply tools used by doctors, some are semi-autonomous and aid in the surgery themselves. In the not too distant future, machine learning will allow robot surgeons to perform operations with complete autonomy. This will free up surgeons to perform the far more complex and experimental procedures that machine learning is not prepared to perform. There has already been great advancement in VR and AR when it comes to surgical and medical procedures.

Natural Language Processing

Being able to understand natural human speech has been anticipated for a long time. With machine learning, being able to talk to robots or AI has become a reality, allowing people to be able to talk to their AI and smartphone devices. There are even “customer support” robots with speech recognition capabilities that can help with a problem or request when asked in a language of choice. Some, if not most, of these types of natural learning ML algorithms have the capability of understanding and being able to decipher more than one language. Natural language processing is even able to take complex legal contract language and translate it into plain language for non-lawyers to understand.

Smart Cars

The autonomous car is already being tested for companies such as food delivery services. These smart cars will interconnect with other cars and other Internet-enabled devices, as well as be able to learn about its owner and passengers so they can make the driver more comfortable. The ability of these vehicles to learn these patterns will help them to make necessary adjustments such as temperature control, audio control, seat setting, and be able to predict the fastest safest route and so on. Because autonomous cars will communicate with each other as they become more numerous on the road, they will become safer. Accidents will drop to next to zero when people are no longer behind the wheel. This is the power of machine learning really flexing its muscles, and only one of the many, many ways ML is set to change the future of humans.

Chapter 15:

Limitation of Machine Learning

There are many problems and limitations with machine learning. Concerns about machine learning limitations have been summarized in a simple phrase, which outlines the main objections to machine learning. It suggests machine learning is greedy, brittle, opaque, and shallow.

Problematic Limitation of Machine Learning

If you examine each of those terms in detail, it clearly outlines that although machine learning has come along in leaps and bounds since its inception, it is far from perfect. But being aware of these limitations when one is learning about ML may help to deal with these limitations and bridge the gap between remarkable to outstanding performance.

Greedy

By calling it greedy, critics of machine learning point to the need for massive amounts of training data to be available in order to successfully train machine learning systems to acceptable levels of error. Because machine learning systems are trained and not programmed, their usefulness will be directly proportional to the amount as well as the quality of the data sets available that are used to train them.

Related to the size of training data required is the fact that, for supervised and semi-supervised machine learning training, the raw data used must first be labeled. In these types of ML systems that data needs to be labeled so that it can be meaningfully employed by the software to train.

In essence, the task of labeling training data means to clean up the raw content and prepare it for the software to ingest. But labeling data can itself be a very complex task, as well as often a laborious and tedious one. Improperly labeled data fed into a supervised machine learning system will produce nothing of value. It will just be a waste of time.

Brittle

To say machine learning is brittle is to highlight a very real and difficult problem in Artificial Intelligence. Machine learning systems are trained to provide extremely accurate and valuable predictions based on the data it has been trained to deal with. The problem, however, comes in when asking that trained system to examine a data set even slightly different from the type it was trained on. This will often cause a complete failure of the system to produce any predictive value. Hence, machine learning systems are unable to contextualize what they have learned and apply it to even extremely similar circumstances to those on which they have been trained. At the same time, attempting to train an already trained machine learning algorithm with a different data set will cause the system to “forget” its previous learning. This means all the previous training, data evaluation, and collection was a waste of time.

Bias in machine learning systems is another example of how machine learning systems can be brittle. There are several different kinds of bias that threaten Artificial Intelligence which include:

- **Bias in Data** —Machine learning is, for the foreseeable future at least, at the mercy of the data used to train it. If this data is biased in any way, whether deliberately or by accident, those biases hidden within it may be passed onto the machine learning system itself. If not caught during the training stage, this bias can taint the work of the system when it is out in the real world doing what it was designed to do.
- **Acquired Bias** —While interacting with people in the real world, machine learning systems can acquire the biases of the people they are exposed to. A real-world example was Microsoft’s Tay, a chatbot designed to interact with people on Twitter using natural language. Within 24 hours, Microsoft was forced to pull the plug on their offensive chatbot that did not start out as the racist chatbot it became in its short 24-hour lifespan. The truly disturbing thing about Tay was that the ML actually started to generate its own comments on the controversial topic, leaving machine learning critics and skeptics to become even more critical and skeptical in the face of a company with as much tech clout as Microsoft not being able to properly train or control their chatbot.
- **Emergent Bias** —An echo chamber is a group or collection of people who all believe the same things. This could be a political meeting in a basement or a chat room on the Internet. Echo chambers are not tolerant of outside ideas, especially those that disagree with or attempt to refute the group’s core beliefs. Facebook has become, in many ways, the preeminent echo chamber producer on Earth. But while the echo chamber phrase is meant to describe a group of people with similar ideas, Facebook’s artificial intelligence has taken this idea one step further. It has emerged as an echo chamber of one. What this does on social media platforms such as Facebook is only let what agrees with the information you have been collecting reach you, while information that may not agree with you gets

discarded or not let through. You may have noticed that when you started out on the social media platform you got all sorts of news feeds being fed through. As you browsed the Internet, opened pages, and read various articles, what appeared on your feed pertained more and more to these searches, articles, and views. Emergent bias is like having a group of people protecting you from the outside world by deflecting bad news, bad influences, and so on. Emergent bias does not allow for a person to challenge their beliefs. How we know we are correct about a particular issue is by testing our beliefs against those who believe otherwise. Do our arguments hold up to their criticism or might we be wrong? In a Facebook echo chamber, that kind of learning and growth becomes less and less possible. Some studies suggest that spending time with people who agree with you tends to polarize groups, making the divisions between them worse. Machine learning, unless otherwise trained, tends to develop tunnel vision by not allowing opposing ideals.

- **Goals Conflict Bias** —Machine learning can often support and reinforce biases that exist in the real world, because doing so increases their reward system. For ML systems it is rewarded when it achieves its goal. For example, say you run a college and you want to increase enrollment. The contract you have with the advertising company is that you will pay a certain amount for each click you get, meaning someone has seen your advertisement for the college and was interested enough to at least click on the link. In cases like these, simple generic adverts like “Go to College!” wouldn’t work very well, especially when competition these days is so steep. In order to attract the attention required you would have to run several simultaneous ad campaigns highlighting the college’s courses, strength, and benefits. It is in the advertiser’s best interest to get as many clicks to the college’s landing pages as possible. To do this the advertiser employs a machine learning algorithm to track who clicks on what advertisement. In turn, the ML begins to target those groups with the specific ads to gain more clicks. Although this at first seems like a win-win situation for the college and the advertiser the ML is developing a conflict bias. Targeting groups of people is a lot similar to profiling in that it may favor men for engineering, women for nursing, and so on, which means eventually only women will see the adverts for courses such as nursing and men for engineering. This aligns with an unfortunate cultural stereotype that still exists in western culture, and your college may be seen to be unwittingly perpetuating it. In its desire to be rewarded, the machine learning assigned to maximizing clicks for your advertising campaign found an existing social bias and exploited it to increase revenue for its owner. These two goals, increasing your enrollment and reducing gender bias in employment, have come into conflict and machine learning sacrificed one to achieve the other.

Opaque

One of the main criticisms of machine learning, and in particular, against Neural Networks, is that they are unable to explain to their creators why they arrive at the decisions they do.

This is a problem for two reasons:

- More and more countries are adopting Internet laws that include a right to an explanation. The most influential of these laws is the GDPR (The EU General Data Protection Regulation). This law guarantees EU citizens the right to an explanation why an algorithm that deals with an important part of their lives made the decision given to them. For example, an EU citizen turned down for a loan by a machine learning algorithm has a right to demand why this happened. Because some artificial intelligence tools like neural networks are often not capable of providing any explanation for their decision, and the fact this decision is hidden in layers of math not readily available for human examination, such an explanation may not be possible. This has, in fact, slowed down the adoption of artificial intelligence in some areas.
- The importance of an artificial intelligence to be able to explain its decisions to its creators by verifying that the underlying process is in fact meeting expectations in the real world. For machine learning, the decision-making process is mathematical and probabilistic. In the real world, decisions often need to be confirmed by the reasoning used to achieve them. Take, for example, a self-driving car involved in an accident. Assuming the hardware of the car is not completely destroyed, experts will want to know why the car took the actions it did. Perhaps there is a court case, and the decision of liability rests on how and why the car took the actions it did. Without transparency around the decision-making process of the software, justice might not be served in the courts, software engineers may not be able to find and correct flaws, and more people might be in danger from the same software running in different cars.

Philosophical Objections and Limitation to Machine Learning

Along with real world problems of machine learning, there are also the philosophical ones. People do not like

change, especially change that may take away a piece of their world in the form of their job or replace them in any way. This becomes a real issue if you consider not only the rate at which the technology industry is growing but the direction it is growing in.

From the invention of the calculator through to the computer, mankind has been fed on movies about machines one day taking over the world. Those who have seen the first *Star Trek* movies would have seen technology that was then a thing of awe, and now not only have become a reality but one that is fast becoming obsolete. Maybe not the space ships, but talking computers and things like tablets.

The invention of the first robot in the automobile industry caused a huge uproar as the robots could not only perform the function a lot more accurately, they did not need down time. This meant that production could be increased and the need for human operators decreased. Hence the reason why movies like the *Terminator* ring through a person's mind when they think of machine learning and AI.

If you stop to think just how much machines are starting to worm their way into a person's every day life, you realize that maybe humans do have a bit of a cause to be at least wary if not skeptical. Already the new generations are so attached to their devices that there are actual clinics that deal with "Internet" and "technology" addictions. When was the last time you saw a young person actually reading a paper book? Or looking up when they were walking or taking public transportation?

Although ML and AI are amazing concepts that are designed to improve the quality of human lives, the fact of the matter is they still raise ethical and philosophical questions along with some strong objections.

A few of these objections include:

Jobs

One of the main concerns surrounding artificial intelligence is the way these systems are encroaching upon human employment. Automation is not a new problem, and vast numbers of jobs have already been lost in manufacturing and other such industries to robots and automated systems.

Because of this, some argue that this concern that machines and artificial intelligence will take over so many jobs there will be no more economy, is not really a threat. There have been a few such takeovers in the past. The economy proved to be able to shift, adapt, and people found other jobs. Some could even argue that the net number of jobs has increased since the loss of so many jobs to automation.

So what is the real problem around jobs loss due to automation?

The problem comes in during the next round of job losses due to AI which will be unprecedented. Artificial intelligence will not only replace drudgery and danger. It will keep going. There are around 3.5 million AI currently in existence, but how long until the number has increased ten-fold and then ten-fold again as AI becomes more advanced? Think about the trucking industry. How long until all trucking is handled by machine learning systems running self-driving trucks? And what about ride-sharing services like Uber and Lyft? What becomes of the human element that used to offer these services?

The question to ask is not what jobs will be replaced, but what jobs can't be replaced. In the machine learning round of employment disruption, white collar jobs are just as much in jeopardy as blue. What about accountants, financial advisers, copywriters, and advertisers?

The pro-artificial intelligence camp argues that this disruption will open up new kinds of jobs, things we can't even imagine. Each major disruption of the economy in the past that displaced many forms of employment with automation often caused the creation of new employment unforeseen before the disruption. The difference with artificial intelligence is there is no reason to believe these new forms of employment won't quickly be taken over by machine learning systems as well.

And all the above assumes the use of the current type of very specific machine learning. What happens if researchers are able to generalize learning in artificial intelligence? What if these systems become able to generalize what they learn and apply what they know in new and different contexts? Such a generalized artificial intelligence system could very quickly learn to do just about anything. So that begs the question of just how real a *Terminator* type future for the human race could actually be. If all jobs become AI, where does that leave humans?

Evil Systems

A very serious danger from artificial intelligence is the fact that anyone with resources can acquire and use it. Western governments are already toying with the idea of autonomous weapons platforms using artificial intelligence to engage and defeat an enemy, with little or no human oversight. As frightening as this might be, in these countries at least, there are checks and balances on the development and deployment of such devices, and in the end, the population can vote for or against such things.

But even if these platforms are developed and only used appropriately, what is to stop an enemy from capturing one, reverse engineering it, and then developing their own? What's to stop a rival power from investing in the

infrastructure to create their own?

The threat of machine learning used by bad actors is very real. How do we control who gains access to this powerful technology? The genie is out of the bottle. It can't be put back in. So, how do we keep it from falling into the wrong hands? We need only keep in mind a simple chatbot that became a racist hellion in under 24 hours to know this is a real issue.

Taking Over the World

So plots like the *Terminator* are probably far-fetched, but if you think about it, you can understand why this could be of great concern to humans. Already most people date online, companies are developing various robotic dolls for companions, and the new generation would rather text the person sitting right next to them than actually talk.

Humans are becoming more and more reliant on their technology, so it is only normal for one to wonder when this is going to end, especially when you have some of the leading minds of the world such as Elon Musk and Steven Hawking voicing their strong opinions on the subject. Even the father of the World Wide Web has his doubts about the way in which his creation is being used. It is not hard to see why the conspiracy theorist, anti-technology, and human rights protestors may be against creations such as ML and AI.

Chapter 16:

Machine Learning and the Future

For all the negative press and doubts, the future of machine learning and AI are very real but also not as bad as one may think. There is enormous potential for this field. There are systems to make human life better, safer, and more comfortable.

These future machine learning systems include the following:

Security

Facial recognition and aberrant behavior detection, are the toolkits from machine learning that are available today. They will become ubiquitous in the future, protecting the public from criminal behavior and getting people in trouble the help they need.

In the cyber world, machine learning will grow and increase its influence in identifying cyber-attacks, malicious software code, and unexpected communication attempts. At the same time, dark hat software crackers are also working on machine learning tools to aid them in breaking into networks, accessing private data, and causing service disruptions. The future of the cyber world will be an ongoing war between white and black hat machine learning tools, much like the war of good and evil anywhere in the world really.

Another sweeping change to security would be autonomous drones controlled by machine learning algorithms. Drones can maintain constant aerial surveillance over entire cities at very little cost. With advancements in image recognition, motion capture, video recognition, and natural language processing, they will be able to communicate with people on the street, respond to natural disasters, and automobile accidents. They will also be able to ferry medications needed in emergency situations where traditional service vehicles cannot gain access. They will be able to find and rescue lost hikers by leading them to safety, delivering them needed supplies, and alerting authorities of their GPS location.

Markets

The rise of machine learning will generate completely new artificial intelligence-based products and services. Entire industries will be created to service this new software, as well as new products to be added to the Internet of Things. This may include a whole new generation of robots complete with learning algorithms and the ability to see, hear, and communicate by using natural language.

Retail

In the retail sector, service ML creates a more personalized and all-around better customer service. Machines do not have moods or are influenced by their surroundings as easily as humans are. If they are able to learn they can quickly adjust and adapt to the customers' needs with a few short questions. Instead of merely showing a person what they want, machine learning will be dedicated to showing them what they need.

A smart algorithm looking out for a person's well-being, for instance, would not throw more and more fast food ads at an at-risk consumer. Instead, reminders about their health, coupons for gym memberships, or recipes for their favorite salads might become part of the artificial intelligence toolkit for notifications. Machine learning will help to balance people's lives in many ways by using knowledge about general health, and even their own medical records, to provide information about not only what they want, but also what they might need.

Healthcare

Machine learning will know almost everything about a person through records on the Internet and records that have been captured from a visit to the doctor. These records would not only be for the patient's benefit but for the doctor's or healthcare provider's. A person not wearing the medic-alert badge could be saved by their healthcare records for instance.

On the diagnostic side, machine learning will do the busy work it is best at: examining our x-rays, blood-work, mammograms, and so on, looking for patterns human beings cannot see. This will afford pre-emptive diagnostic information to doctors so they can head off serious illness at early stages or prevent them altogether.

Many if not most surgeries will be performed by artificial intelligence-enabled robots, either assisting human surgeons, being supervised by them, or even, eventually, working fully autonomously.

Machine learning will be able to more effectively monitor things such as blood gasses under anesthesia, heart rate, and other health measures during operations. They are able to react in milliseconds should something appear to be wrong. Iatrogenic disease will decrease dramatically, if not disappear completely.

The Environment and Sustainability

Machine learning will be able to study the movement of people in cities, what they use and don't use, their patterns

of use, how they travel, and where. Deep learning from this data will allow city planners to employ machine learning algorithms to design and construct both more efficient and pleasant cities. It will allow massive increases in density without sacrificing quality of life. These efficiencies will reduce or even possibly eliminate net carbon emissions from cities.

Augmented Reality

When wearing Google, Microsoft, Apple, or Facebook glasses in the future, the embedded processes, video capture, audio capture, and microphones on these devices will do much more than give directions to find a location. Machine learning will be able to see what we see and provide explanations along with predictive guidance throughout our day. Imagine having your day “painted” with relevant information on interior walls and doors, and outside on buildings with signs, guiding you through your schedule topped off with the information you need when you need it.

Information Technology

Machine learning will become a tool people and businesses can apply as needed like SasS. This MLaaS will allow software to be aware of its surroundings, to see people, to hear people, and speak to people in a natural language. Connected to the Internet, every device will become smart, and generate an ecosphere around people that attends to their needs and concerns, often before they even realize they have them. These “Cognitive Services” will provide APIs and SDKs, leading to rapid “smart” software development and deployment.

Specialized hardware will increase the speed of machine learning training, as well as increase its speed in servicing the public. Dedicated AI chips will bring about a huge change in artificial intelligence speed and ubiquity.

Microcomputers will come equipped with machine learning capabilities so that even the smallest device will be able to understand its surroundings. Where there is no convenient power supply, these devices will run on dime-sized batteries, lasting for months or years of service before needing replacement.

Trust Barriers

Natural speech means we will be able to talk to our devices and be understood by them. The current trust barriers between some people, business sectors, and governments will slowly break down as machine learning demonstrates its reliability and effectiveness. Improved unsupervised learning will reduce the time required to develop new machine learning software with required specifications.

Conclusion

The impact of machine learning on our world is already ubiquitous. Our cars, our phones, our houses, and so much more are already being controlled and maintained through rudimentary machine learning systems. But in the future, machine learning will radically change the world. Some of those changes are easy to predict. In the next decade or two, people will no longer drive cars. Instead, automated cars will drive people. But in many other ways, the effect of machine learning on our world is difficult to predict.

ML and AI raise many questions such as:

- Will machine learning algorithms replace so many jobs, from trucking to accounting to many other disciplines?
- If so, what will there be left for humans to do?
- In 100 years, will there be work for anyone at all?

For all our advancements on the topics, the fact is we don’t know the answer to questions like these. So far there is no limit to what machine learning can accomplish, given time and data and the will to use it to achieve a particular task. If machines do advance to such limits as anticipated, things will still be done, such as food will still be grown, picked, and delivered. The difference will be in how the food is being grown, picked, and delivered.

The only real certainty about artificial intelligence and machine learning is that it is increasing in both speed of deployment and in areas of influence. It promises many benefits and many radical changes in our society. In order to not get left behind, it stands to reason that learning and growing with ML and AI gives a person a greater edge and foothold on the future.

Machine Learning Recap

In conclusion, machine learning is a branch of artificial intelligence that involves the design and development of systems capable of showing an improvement in performance based on their previous experiences. This means that, when reacting to the same situation, a machine should show improvement from time to time.

With machine learning, software systems are able to predict accurately without having to be programmed explicitly.

The goal of machine learning is to build algorithms which can receive input data then use statistical analysis so as to predict the output value in an acceptable range.

Machine learning originated from pattern recognition and the theory that computers are able to learn without the need for programming them to perform tasks. Researchers in the field of artificial intelligence want to determine whether computers are able to learn from data. Machine learning is an iterative approach, and this is why models are able to adapt as they are being exposed to new data. Models learn from their previous computations so as to give repeatable, reliable results and decisions

References

- Alan Turing: Creator of modern computing.* (n.d.). BBC Teach. <https://www.bbc.co.uk/teach/alan-turing-creator-of-modern-computing/zhwp7nb>
- Analytical Engine.* (n.d.). Encyclopedia Britannica. <https://www.britannica.com/technology/Analytical-Engine>
- Baxter.*(n.d.). Robots. <https://robots.ieee.org/robots/baxter/>
- Blaise Pascal (1623~166).* (n.d.). Educalc. <https://www.educalc.net/196488.page>
- Boids.* (n.d.). Wikipedia. <https://en.wikipedia.org/wiki/Boids#:~:text=Boids%20is%20an%20artificial%20life,the%20flocking%20>
- Brownlee, J. (2019, May 22). *Difference Between Classification and Regression in Machine Learning.* Machine Learning Mastery. <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/#:~:text=Fundamentally%2C%20classification%20is%20about%20predicting,is%20about%20p>
- Cillania, M. (2015, October 13). *Ada Lovelace: The First Computer Programmer.* Mental Floss. <https://www.mentalfloss.com/article/53131/ada-lovelace-first-computer-programmer>
- Dacombe, J. (2017, October 23). *An introduction to Artificial Neural Networks (with example).* Medium. <https://medium.com/@jamesdacombe/an-introduction-to-artificial-neural-networks-with-example-ad459bb6941b>
- Deep Blue.* (n.d.). IBM. <https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>
- Foote, K. (2016, August 16). *A Brief History of the Internet of Things.* Dataversity. <https://www.dataversity.net/brief-history-internet-things/#:~:text=One%20of%20the%20first%20examples,at%20the%20Carnegie%20Melon%20Universit>
- George Boole.* (n.d.). Encyclopedia Britannica. <https://www.britannica.com/biography/George-Boole>
- Gonzalez, O. (2018, January 07). *GOTTFRIED WILHELM LEIBNIZ: HOW HIS BINARY SYSTEMS SHAPED THE DIGITAL AGE.* Inverse. <https://www.inverse.com/article/46587-gottfried-wilhelm-leibniz-binary-system>
- History of Computing.* (n.d.). Meeting Tomorrow. <https://meetingtomorrow.com/blog/history-of-computers-and-computing/>
- History of Machine Learning.* (n.d.). Imperial College London. <https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html>
- IFR forecast: 1.7 million new robots to transform the world's factories by 2020.* (n.d.). IFR. <https://ifr.org/news/ifr-forecast-1.7-million-new-robots-to-transform-the-worlds-factories-by-20/>
- Internet of things.* (n.d.). Wikipedia. https://en.wikipedia.org/wiki/Internet_of_things
- Jain, K. (2015, January 05). *Scikit-learn(sklearn) in Python – the most important Machine Learning tool I learnt last year!* Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/>
- Lewis, M., & Bekey, G. (1992). *The Behavioral Self-Organization of Nanorobots Using Local Rules.* ResearchGate. https://www.researchgate.net/publication/3690783_The_Behavioral_Self-organization_Of_Nanorobots_Using_Local_Rules
- Loiseau, J. (2019, March 11). *Rosenblatt's perceptron, the first modern neural network.* Medium. <https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a>
- Marsalli, M. (n.d.). *McCulloch-Pitts Neurons.* The Mind Project. <http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=212>
- Mittal, V. (2017, October 03). *Top 15 Deep Learning applications that will rule the world in 2018 and beyond.* Medium. <https://medium.com/breathe-publication/top-15-deep-learning-applications-that-will-rule-the-world-in-2018-and-beyond-7c6130c43b01>
- Osinski, B. & Budek, K. (2018, July 05). *What is reinforcement learning? The complete guide.* Deepsens.ai. <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/>
- Pythagoras' Theorem [PDF File].* (2009). Mathcentre. <http://www.mathcentre.ac.uk/resources/uploaded/mc-ty->

[pythagoras-2009-1.pdf](#)

Schwartz, O. (2019, November 25). *In 2016, Microsoft's Racist Chatbot Revealed the Dangers of Online Conversation*. IEEE Spectrum. <https://spectrum.ieee.org/tech-talk/artificial-intelligence/machine-learning/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation>

Study: Attack on KrebsOnSecurity Cost IoT Device Owners \$323K. (2018, March 07). KrebsOnSecurity. <https://krebsonsecurity.com/2018/05/study-attack-on-krebsonsecurity-cost-iot-device-owners-323k/>

This little-known pioneering educator put coding in the classroom. (2016, August 25). The Conversation. <https://theconversation.com/this-little-known-pioneering-educator-put-coding-in-the-classroom-63971>

Top Five Use Cases of Tensorflow: Deep Learning. (2017, February 03). Exastax. <https://www.exastax.com/deep-learning/top-five-use-cases-of-tensorflow/>

Unimate. (n.d.). Robotics. <https://www.robotics.org/joseph-engelberger/unimate.cfm>

Vaseekaran, G. (2018, September 28). *Machine Learning: Supervised Learning vs Unsupervised Learning*. Medium. <https://medium.com/@gowthamy/machine-learning-supervised-learning-vs-unsupervised-learning-f1658e12a780>

What's the difference between artificial intelligence (AI), machine learning (ML) and natural language processing (NLP)? (n.d.). Sonix. <https://sonix.ai/articles/difference-between-artificial-intelligence-machine-learning-and-natural-language-processing>

Wilhelm Schickard invented a calculating machine. (n.d.). Centre for Computing History. <http://www.computinghistory.org.uk/det/5921/Wilhelm-Schickard-invented-a-calculating-machine/>

Why Python is the preferred language for Machine Learning? (2017, August 21). Ivy Professional School. <http://ivyproschoool.com/blog/2017/08/21/why-python-is-the-preferred-language-for-machine-learning/>

Zajechowski, M. (n.d.). *Automatic Speech Recognition (ASR) Software – An Introduction*. UsabilityGeek. <https://usabilitygeek.com/automatic-speech-recognition-asr-software-an-introduction/>

Images

Anaitit. (n.d.). *Kytin PC Tieto Kone (Switch PC Information Machine)* [Photograph]. Pixabay. <https://pixabay.com/fi/photos/kytkin-pc-tietokone-piiri-5015530/>

CCO Dominio Pubblico. (n.d.). *Cardboard Robots* [Photograph]. PXHere. <https://pxhere.com/it/photo/1559681>

Creative Commons Zero - CC0. (n.d.). *Black and red laptop* [Photograph]. PeakPX. <https://www.peakpx.com/513079/black-and-red-laptop>

Creative Commons - CC0. (n.d.). *Men's white prosthetic right arm* [Photograph]. PeakPX. <https://www.peakpx.com/549666/men-s-white-prosthetic-right-arm>

Turner, R. (2020, May 28). *All Technical Screenshots* [Screenshot].

Turner, R. (2020, May 28). *All Script Screenshots* [Screenshot].

Turner, R. (2020, May 28). *All Analysis Screenshots* [Screenshot].