CHAPTER 1

Introduction to Machine Learning

Welcome to *Python Machine Learning*! The fact that you are reading this book is a clear indication of your interest in this very interesting and exciting topic.

This book covers *machine learning*, one of the hottest programming topics in more recent years. *Machine learning (ML)* is a collection of algorithms and techniques used to design systems that learn from data. These systems are then able to perform predictions or deduce patterns from the supplied data.

With computing power increasing exponentially and prices decreasing simultaneously, there is no better time for machine learning. Machine learning tasks that usually require huge processing power are now possible on desktop machines. Nevertheless, machine learning is not for the faint of heart—it requires a good foundation in mathematics, statistics, as well as programming knowledge. The majority of the books in the market on machine learning go into too much detail, which often leaves beginning readers gasping for air. Most of the discussion on machine learning revolves heavily around statistical theories and algorithms, so unless you are a mathematician or a PhD candidate, you will likely find them difficult to digest. For most people, developers in particular, what they want is to have a foundational understanding of how machine learning works, and most importantly, how to apply machine learning in their applications. It is with this motive in mind that I set out to write this book.

This book will take a *gentle* approach to machine learning. I will attempt to do the following:

- Cover the libraries in Python that lay the foundation for machine learning, namely NumPy, Pandas, and matplotlib.
- Discuss machine learning using Python and the Scikit-learn libraries. Where possible, I will manually implement the relevant machine learning algorithm using Python. This will allow you to understand how the various machine learning algorithms work behind the scenes. Once this is done, I will show how to use the Scikit-learn libraries, which make it really easy to integrate machine learning into your own apps.
- Cover the common machine learning algorithms—regressions, clustering, and classifications.

TIP **It is not the intention of this book to go into a deep discussion of machine learning algorithms. Although there are chapters that discuss some of the mathematical concepts behind the algorithms, it is my intention to make the subject easy to understand and hopefully motivate you to learn further.**

Machine learning is indeed a very complex topic. But instead of discussing the complex mathematical theories behind it, I will cover it using easy-to-understand examples and walk you through numerous code samples. This code-intensive book encourages readers to try out the numerous examples in the various chapters, which are designed to be independent, compact, and easy to follow and understand.

What Is Machine Learning?

If you have ever written a program, you will be familiar with the diagram shown in Figure 1.1. You write a program, feed some data into it, and get your output. For example, you might write a program to perform some accounting tasks for your business. In this case, the data collected would include your sales records, your inventory lists, and so on. The program would then take in the data and calculate your profits or loss based on your sales records. You may also perhaps churn out some nice and fanciful charts showing your sales performance. In this case, the output is the profit/loss statement, as well as other charts.

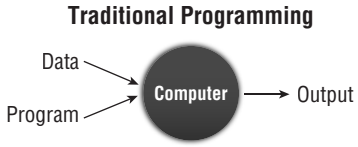

Figure 1.1: In traditional programming, the data and the program produce the output

For many years, traditional desktop and web programming have dominated the landscape, and many algorithms and methodologies have evolved to make programs run more efficiently. In more recent years, however, machine learning has taken over the programming world. Machine learning has transformed the paradigm in Figure 1.1 to a new paradigm, which is shown in Figure 1.2. Instead of feeding the data to the program, you now use the data and the output that you have collected to derive your program (also known as the *model*). Using the same accounting example, with the machine learning paradigm, you would take the detailed sales records (which are collectively both the data and output) and use them to derive a set of rules to make predictions. You may use this model to predict the most popular items that will sell next year, or which items will be less popular going forward.

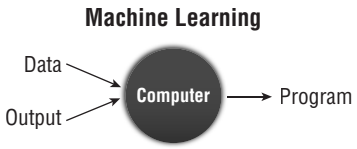

Figure 1.2: In machine learning, the data and the output produce the program

TIP **Machine learning is about finding *patterns* in data.**

What Problems Will Machine Learning Be Solving in This Book?

So, what exactly is machine learning? Machine learning (ML) is a collection of algorithms and techniques used to design systems that learn from data. ML algorithms have a strong mathematical and statistical basis, but they do not take into account domain knowledge. ML consists of the following disciplines:

- Scientific computing
- Mathematics
- Statistics

A good application of machine learning is trying to determine if a particular credit card transaction is fraudulent. Given past transaction records, the data scientist's job is to clean up and transform the data based on domain knowledge so that the right ML algorithm can be applied in order to solve the problem (in this case determine if a transaction is fraudulent). A data scientist needs to know about which method of machine learning will best help in completing this task and how to apply it. The data scientist does not necessarily need to know how that method works, although knowing this will always help in building a more accurate learning model.

In this book, there are three main types of problems that we want to solve using machine learning. These problem types are as follows:

Classification: Is this A or B?

Regression: How much or how many?

Clustering: How is this organized?

Classification

In machine learning, *classification* is identifying to which set of categories a new observation belongs based on the set of training data containing in the observed categories. Here are some examples of classification problems:

- Predicting the winner for the U.S. 2020 Presidential Election
- Predicting if a tumor is cancerous
- Classifying the different types of flowers

A classification problem with two classes is known as a *two-class classification* problem. Those with more than two classes are known as *multi-class classification* problems.

The outcome of a classification problem is a discrete value indicating the predicted class in which an observation lies. The outcome of a classification problem can also be a continuous value, indicating the likelihood of an observation belonging to a particular class. For example, candidate A is predicted to win the election with a probability of 0.65 (or 65 percent). Here, 0.65 is the continuous value indicating the confidence of the prediction, and it can be converted to a class value ("win" in this case) by selecting the prediction with the highest probability.

Chapter 7 through Chapter 9 will discuss classifications in more detail.

Regression

Regression helps in forecasting the future by estimating the relationship between variables. Unlike classification (which predicts the class to which an observation belongs), regression returns a continuous output variable. Here are some examples of regression problems:

- Predicting the sales number for a particular item for the next quarter
- Predicting the temperatures for next week
- Predicting the lifespan of a particular model of tire

Chapter 5 and Chapter 6 will discuss regressions in more detail.

Clustering

Clustering helps in grouping similar data points into intuitive groups. Given a set of data, clustering helps you discover how they are organized by grouping them into natural clumps.

Examples of clustering problems are as follows:

- Which viewers like the same genre of movies
- Which models of hard drives fail in the same way

Clustering is very useful when you want to discover a specific pattern in the data. Chapter 10 will discuss clustering in more detail.

Types of Machine Learning Algorithms

Machine learning algorithms fall into two broad categories:

- *Supervised learning algorithms* are trained with labeled data. In other words, data composed of examples of the desired answers. For instance, a model that identifies fraudulent credit card use would be trained from a dataset with labeled data points of known fraudulent and valid charges. Most machine learning is supervised.
- *Unsupervised learning algorithms* are used on data with no labels, and the goal is to find relationships in the data. For instance, you might want to find groupings of customer demographics with similar buying habits.

Supervised Learning

In supervised learning, a labeled dataset is used. A *labeled dataset* means that a group of data has been tagged with a label. This label provides informative meaning to the data. Using the label, unlabeled data can be predicted to obtain a new label. For example, a dataset may contain a series of records containing the following fields, which record the size of the various houses and the prices for which they were sold:

House Size, Price Sold

In this very simple example, *Price Sold* is the label. When plotted on a chart (see Figure 1.3), this dataset can help you predict the price of a house that is yet to be sold. Predicting a price for the house is a *regression* problem.

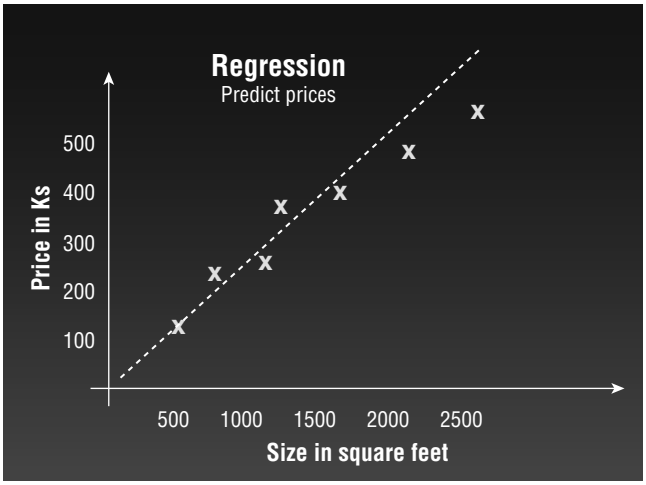

Figure 1.3: Using regression to predict the expected selling price of a house

Using another example, suppose that you have a dataset containing the following:

Tumor Size, Age, Malignant

The *Malignant* field is a label indicating if a tumor is cancerous. When you plot the dataset on a chart (see Figure 1.4), you will be able to classify it into two distinct groups, with one group containing the cancerous tumors and the other containing the benign tumors. Using this grouping, you can now predict if a new tumor is cancerous or not. This type of problem is known as a *classification* problem.

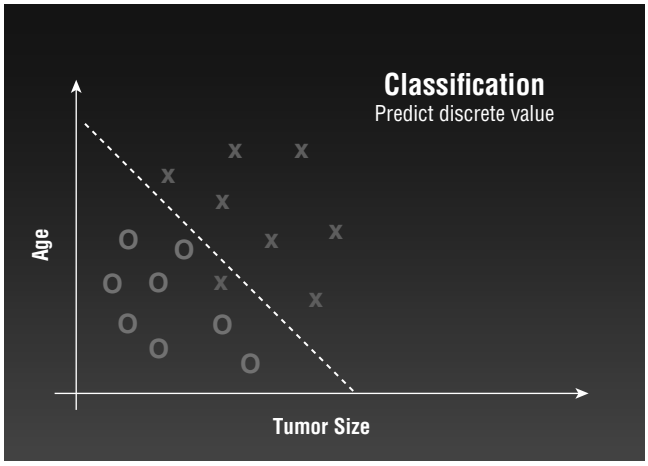

Figure 1.4: Using classification to categorize data into distinct classes

TIP **Chapter 6 through Chapter 9 will discuss supervised learning algorithms in more detail.**

Unsupervised Learning

In unsupervised learning, the dataset used is not labeled. An easy way to visualize unlabeled data is to consider the dataset containing the waist size and leg length of a group of people:

Waist Size, Leg Length

Using unsupervised learning, your job is to try to predict a pattern in the dataset. You may plot the dataset in a chart, as shown in Figure 1.5.

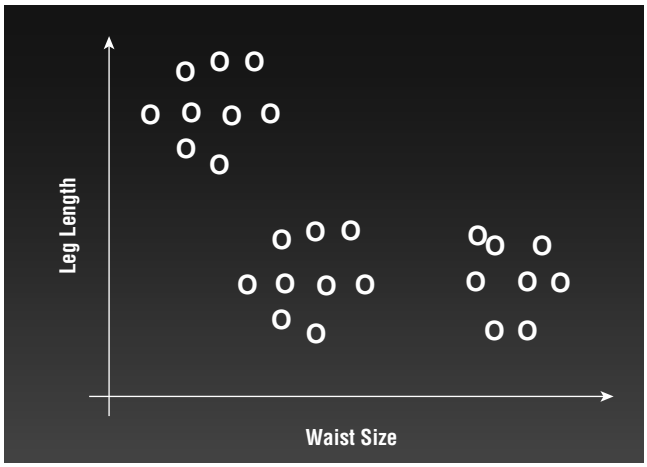

Figure 1.5: Plotting the unlabeled data

You can then use some clustering algorithms to find the patterns in the dataset. The end result might be the discovery of three distinct groups of clusters in the data, as shown in Figure 1.6.

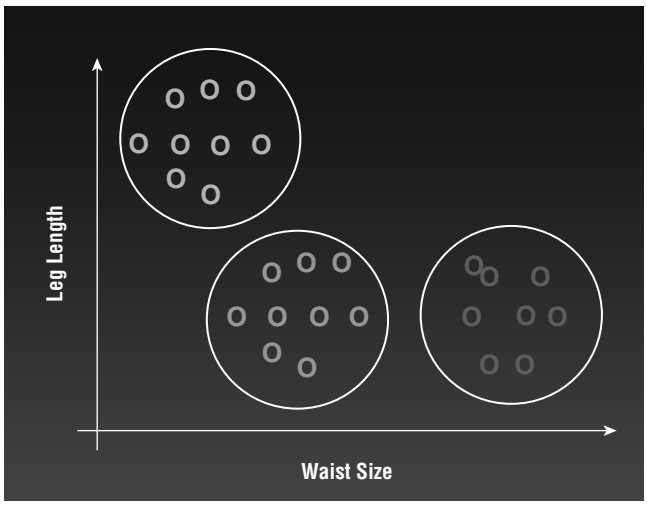

Figure 1.6: Clustering the points into distinct groups

TIP **Chapter 10 will discuss unsupervised learning algorithms in more detail.**

Getting the Tools

For this book, all of the examples are tested using Python 3 and the Scikit-learn library, a Python library that implements the various types of machine learning algorithms, such as classification, regression, clustering, decision tree, and more. Besides Scikit-learn, you will also be using some complementary Python libraries—NumPy, Pandas, and matplotlib.

While you can install the Python interpreter and the other libraries individually on your computer, the trouble-free way to install all of these libraries is to install the Anaconda package. *Anaconda* is a free Python distribution that comes with all of the necessary libraries that you need to create data science and machine learning projects.

Anaconda includes the following:

- The core Python language
- The various Python packages (libraries)
- *conda*, Anaconda's own package manager for updating Anaconda and packages
- Jupyter Notebook (formerly known as *iPython Notebook*), a web-based editor for working with Python projects

With Anaconda, you have the flexibility to install different languages (R, JavaScript, Julia, and so on) to work in Jupyter Notebook.

Obtaining Anaconda

To download Anaconda, go to `https://www.anaconda.com/download/`. You will be able to download Anaconda for these operating systems (see Figure 1.7):

- Windows
- macOS
- Linux

Download the Python 3 for the platform you are using.

NOTE **At the time of this writing, Python is in version 3.7.**

TIP **For this book, we will be using Python 3. So be sure to download the correct version of Anaconda containing Python 3.**

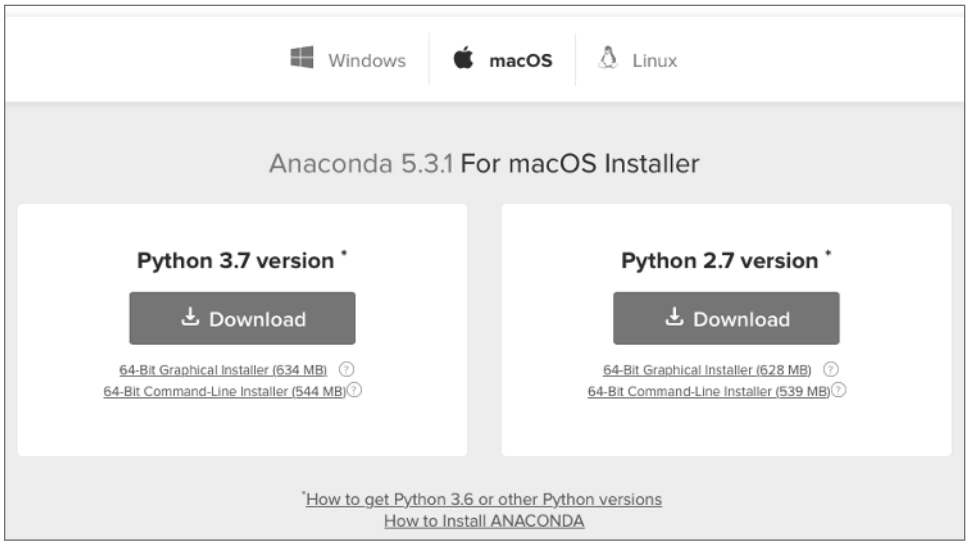

Figure 1.7: Downloading Anaconda for Python 3

Installing Anaconda

Installing Anaconda is mostly a non-event process. Double-click the file that you have downloaded, and follow the instructions displayed on the screen. In particular, Anaconda for Windows has the option to be installed only for the local user. This option does not require administrator rights, and hence it is very useful for users who are installing Anaconda on company-issued computers, which are usually locked down with limited user privileges.

Once Anaconda is installed, you will want to launch Jupyter Notebook. Jupyter Notebook is an open source web application, which allows you to create and share documents that contain documentation, code, and more.

Running Jupyter Notebook for Mac

To launch Jupyter from macOS, launch *Terminal* and type the following command:

```
$ jupyter notebook
```

You will see the following:

```
$ jupyter notebook
[I 18:57:03.642 NotebookApp] JupyterLab extension loaded from
/Users/weimenglee/anaconda3/lib/python3.7/site-packages/jupyterlab
[I 18:57:03.643 NotebookApp] JupyterLab application directory is
/Users/weimenglee/anaconda3/share/jupyter/lab
[I 18:57:03.648 NotebookApp] Serving notebooks from local directory:
/Users/weimenglee/Python Machine Learning
[I 18:57:03.648 NotebookApp] The Jupyter Notebook is running at:
```

```
[I 18:57:03.648 NotebookApp]
http://localhost:8888/?token=3700cfe13b65982612c0e1975ce3a68107399b07f89
b85fa
[I 18:57:03.648 NotebookApp] Use Control-C to stop this server and shut
down all kernels (twice to skip confirmation).
[C 18:57:03.649 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first
time,
    to login with a token:
        http://localhost:8888/?token=3700cfe13b65982612c0e1975ce3a681073
99b07f89b85fa
[I 18:57:04.133 NotebookApp] Accepting one-time-token-authenticated
connection from ::1
```

Essentially, Jupyter Notebook starts a web server listening at port 8888. After a while, a web browser will launch (see Figure 1.8).

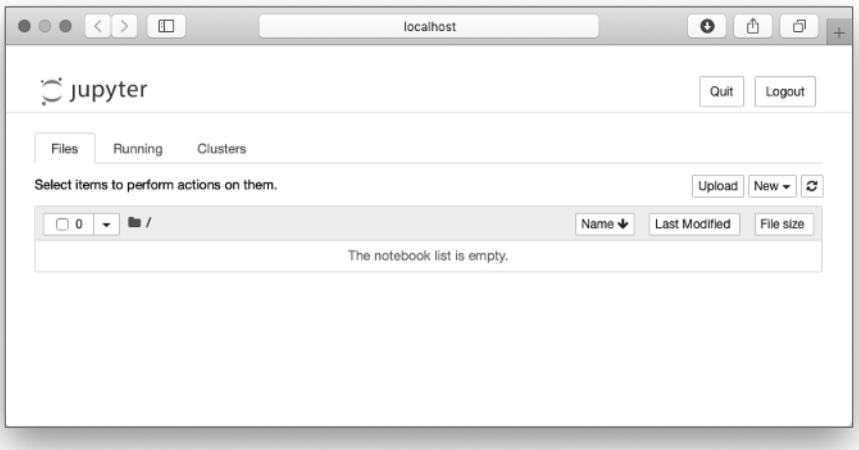

Figure 1.8: The Jupyter Notebook Home page

TIP **The Home page of Jupyter Notebook shows the content of the directory from where it is launched. Hence, it is always useful to change to the directory that contains your source code first, prior to launching Jupyter Notebook.**

Running Jupyter Notebook for Windows

The best way to launch Jupyter Notebook in Windows is to launch it from the *Anaconda Prompt*. The Anaconda Prompt automatically runs the batch file located at `C:\Anaconda3\Scripts\activate.bat` with the following argument:

```
C:\Anaconda3\Scripts\activate.bat C:\Anaconda3
```

TIP **Note that the exact location of the Anaconda3 folder can vary. For example, by default Windows 10 will install Anaconda in** `C:\Users\<username>\AppData\Local\Continuum\anaconda3` **instead of** `C:\Anaconda3`**.**

This sets up the necessary paths for accessing Anaconda and its libraries.

To launch the Anaconda Prompt, type **`Anaconda Prompt`** in the Windows Run textbox. To launch Jupyter Notebook from the Anaconda Prompt, type the following:

```
(base) C:\Users\Wei-Meng Lee\Python Machine Learning>jupyter notebook
```

You will then see this:

```
[I 21:30:48.048 NotebookApp] JupyterLab beta preview extension loaded from
C:\Anaconda3\lib\site-packages\jupyterlab
[I 21:30:48.048 NotebookApp] JupyterLab application directory is
C:\Anaconda3\share\jupyter\lab
[I 21:30:49.315 NotebookApp] Serving notebooks from local directory:
C:\Users\Wei-Meng Lee\Python Machine Learning
[I 21:30:49.315 NotebookApp] 0 active kernels
[I 21:30:49.322 NotebookApp] The Jupyter Notebook is running at:
[I 21:30:49.323 NotebookApp]
http://localhost:8888/?token=482bfe023bd77731dc132b5340f335b9e450ce5e1c4
d7b2f
[I 21:30:49.324 NotebookApp] Use Control-C to stop this server and shut
down all kernels (twice to skip confirmation).
[C 21:30:49.336 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first
time,
    to login with a token:
        http://localhost:8888/?token=482bfe023bd77731dc132b5340f335b9e45
0ce5e1c4d7b2f
[I 21:30:49.470 NotebookApp] Accepting one-time-token-authenticated
connection from ::1
```

Essentially, Jupyter Notebook starts a web server listening at port 8888. It then launches your web browser showing you the page in Figure 1.9.

Creating a New Notebook

To create a new notebook, locate the New button on the right side of the screen and click it. You should be able to see Python 3 in the dropdown (see Figure 1.10). Click this option.

Your new notebook will now appear (see Figure 1.11).

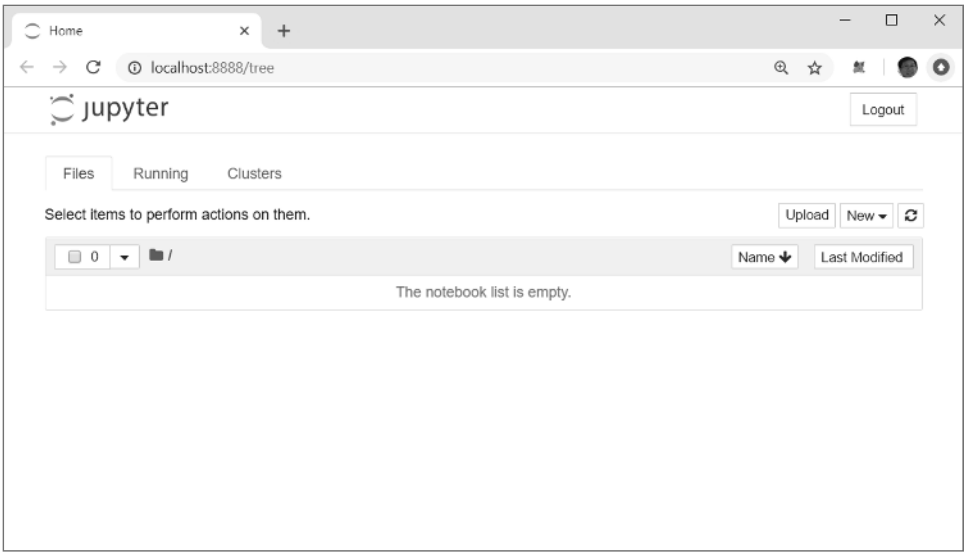

Figure 1.9: Jupyter Notebook showing the Home page

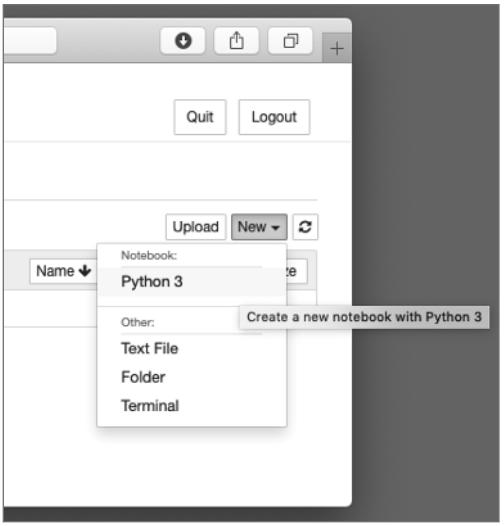

Figure 1.10: Creating a new Python 3 notebook

Naming the Notebook

By default, your notebook will be named "Untitled". To give it a suitable name, click "Untitled" and type in a new name. Your notebook will be saved in the directory from which you have launched Jupyter Notebook. The notebook will be saved with a filename that you have given it, together with the `.ipynb` extension.

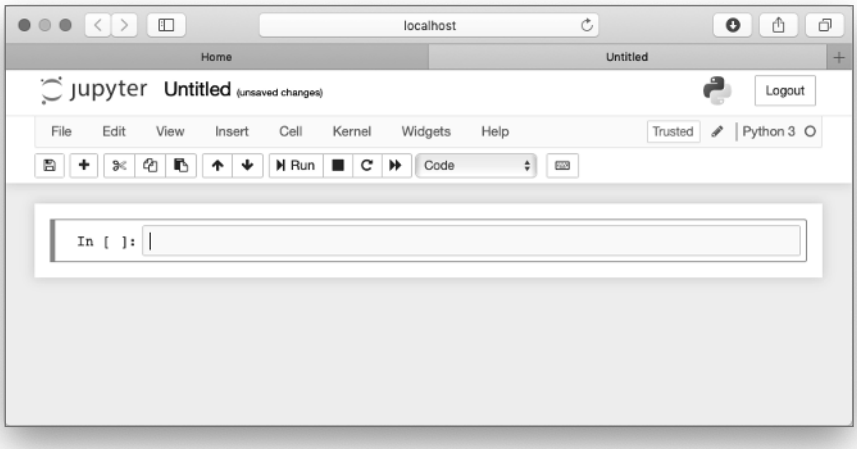

Figure 1.11: The Python 3 notebook created in Jupyter Notebook

TIP **Jupyter Notebook was previously known as *iPython Notebook*; hence the `.ipynb` extension.**

Adding and Removing Cells

A notebook contains one or more cells. You can type Python statements in each cell. Using Jupyter Notebook, you can divide your code into multiple snippets and put them into cells so that they can be run individually.

To add more cells to your notebook, click the + button. You can also use the Insert menu item and select the option Insert Cell Above to add a new cell above the current cell, or select the Insert Cell Below option to add a new cell below the current cell.

Figure 1.12 shows the notebook containing two cells.

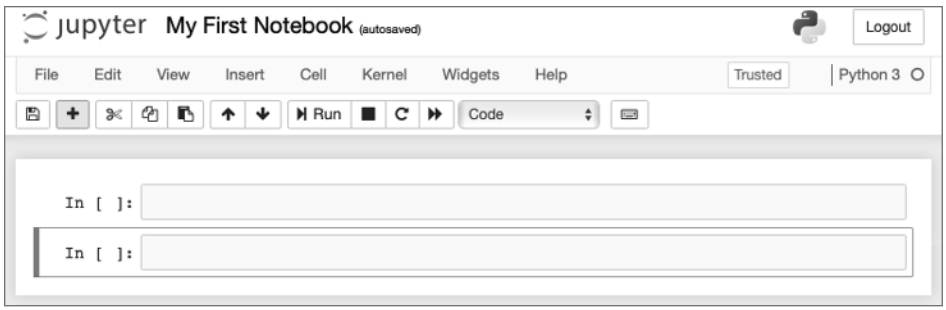

Figure 1.12: The notebook with two cells

Running a Cell

Each cell in a Jupyter Notebook can be run independently. To execute (run) the code in a cell, press Ctrl+Enter, or click the arrow icon displayed to the left of the cell when you hover your mouse over it (see Figure 1.13).

Figure 1.13: Running (executing) the code in the cell

When cells are run, the order in which they were executed is displayed as a running number. Figure 1.14 shows two cells executed in the order shown. The number 1 in the first cell indicates that this cell was executed first, followed by number 2 in the second cell. The output of the cell is displayed immediately after the cell. If you go back to the first cell and run it, the number will then change to 3.

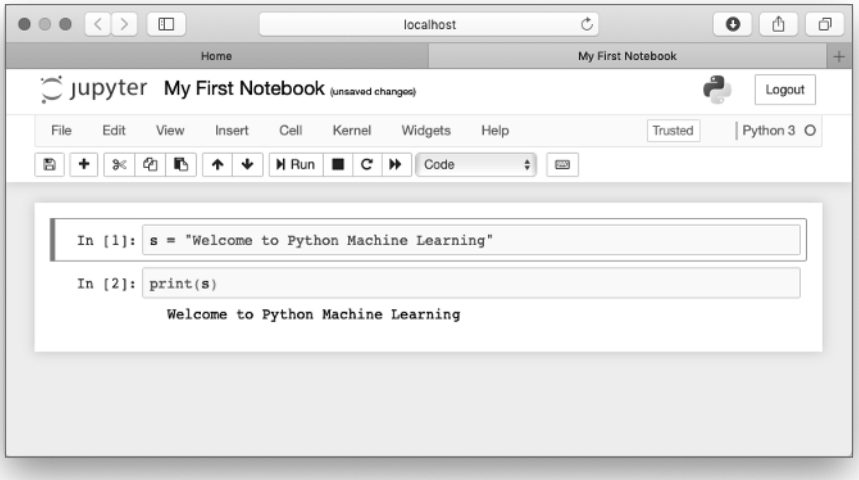

Figure 1.14: The number displayed next to the cell indicates the order in which it was run

As you can see, code that was executed previously in another cell retains its value in memory when you execute the current cell. However, you need to be careful when you are executing cells in various orders. Consider the example in Figure 1.15. Here, we have three cells. In the first cell, we initialize the value of `s` to a string and print its value in the second cell. In the third cell, we change the value of `s` to another string.

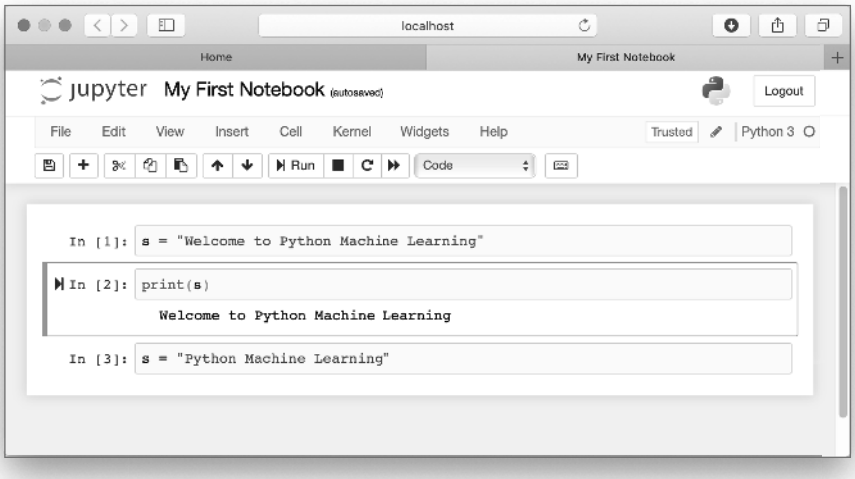

Figure 1.15: The notebook with three cells

Very often, in the midst of testing your code, it is very common that you may make modifications in one cell and go back to an earlier cell to retest the code. In this example, suppose that you go back and rerun the second cell. In this case, you would now print out the new value of `s` (see Figure 1.16). At first glance, you may be expecting to see the string "Welcome to Python Machine Learning," but since the second cell was rerun after the third cell, the value of `s` will take on the "Python Machine Learning" string.

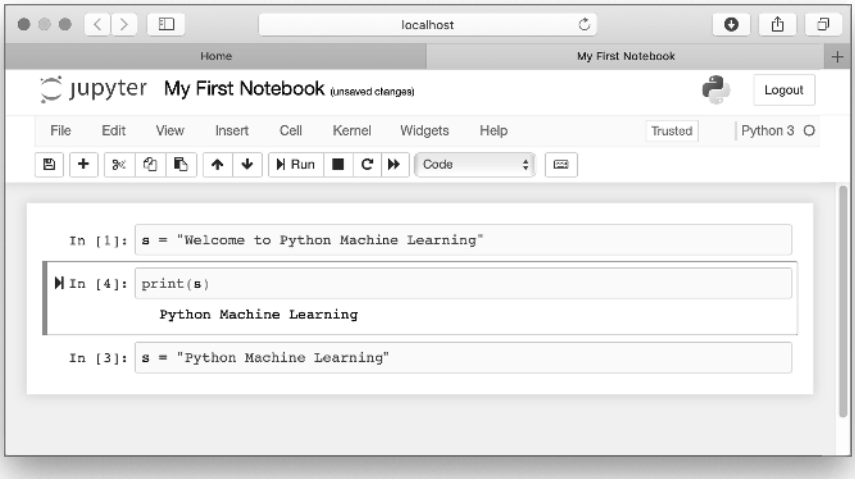

Figure 1.16: Executing the cells in non-linear order

To restart the execution from the first cell, you need to restart the kernel, or select Cell ⇨ Run All.

Restarting the Kernel

As you can run any cell in your notebook in any order, after a while things may get a little messy. You may want to restart the execution and start all over again. This is where restarting the kernel is necessary (see Figure 1.17).

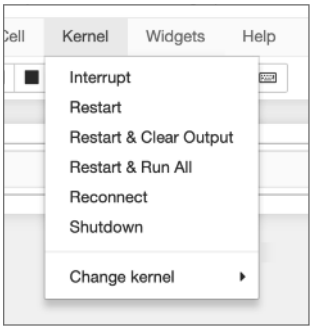

Figure 1.17: Restarting the kernel

TIP **When your code goes into an infinite loop, you need to restart the kernel. There are two common scenarios for restarting the kernel:**

Restart & Clear Output **Restart the kernel and clear all of the outputs. You can now run any of the cells in any order you like.**

Restart & Run All **Restart the kernel and run all of the cells from the first to the last. This is very useful if you are satisfied with your code and want to test it in its entirety.**

Exporting Your Notebook

Once you are done with your testing in Jupyter Notebook, you can now export code from your notebook to a Python file. To do so, select File ⇨ Download as ⇨ python (.py). (See Figure 1.18.)

A file with the same name as your notebook, but now with the `.py` extension, will be downloaded to your computer.

TIP **Make sure that you select the python (.py) option and not the Python (.py) option. The latter option saves the file with an `.html` extension.**

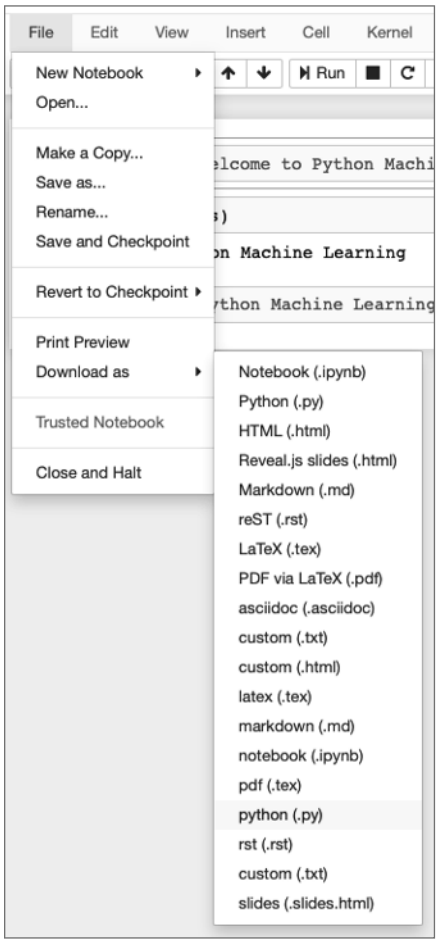

Figure 1.18: Exporting your notebook to a Python file

Getting Help

You can get help in Jupyter Notebook quite easily. To get help on a function in Python, position your cursor on the function name and press Shift+Tab. This will display a pop-up known as the *tooltip* (see Figure 1.19).

To expand the tooltip (see Figure 1.20), click the + button on the upper-right corner of the tooltip. You can also get the expanded version of the tooltip when you press Shift+Tab+Tab.

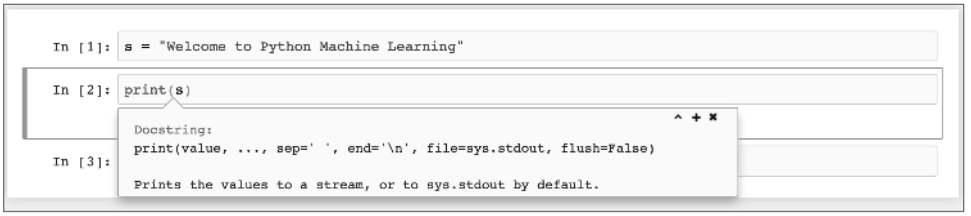

Figure 1.19: The tooltip displays help information

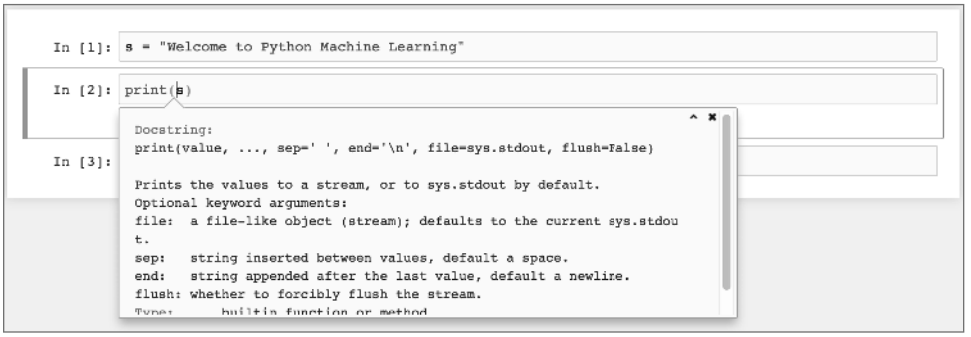

Figure 1.20: Expanding the tooltip to show more detail

Summary

In this chapter, you learned about machine learning and the types of problems that it can solve. You also studied the main difference between supervised and unsupervised learning. For developers who are new to Python programming, I strongly advise you to install Anaconda, which will provide all of the libraries and packages you'll need to follow the examples in this book. I know that you are all eager to start learning, so let's move onward to Chapter 2!