**CHAPTER 7**

**LIBRARIES USED AND THEIR INSTALLATION**

**7.1 NumPy**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, [16] various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much

more.

**7.1.1 Description**

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python.

**7.1.2 Function**

NumPy fully supports an object-oriented approach, starting, once again, with ndarray. For example, the array is a class, possessing numerous methods and attributes. Many of its methods are mirrored by functions in the outermost NumPy namespace, allowing the programmer to code in whichever paradigm they prefer. This flexibility has allowed the NumPy array dialect and NumPy array class to become the de-facto language of seventy multi-dimensional data interchange used in Python.

**7.1.3 Installation**

1. If you use conda, you can install NumPy from the defaults or conda-forge channels:

conda create -n my-env

conda activate my-env

conda config –env –add channels conda-forge

conda install numpy

1. If you use pip, you can install NumPy with:

pip install numpy

**7.1.4 Use in the Project:**

At the core of the NumPy package, is the ndarray object.

1. Vectorization describes the absence of any explicit looping, indexing, etc., in the code- NumPy uses Vectorization hence is very fast.
2. NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
3. A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

In the project, the images are stored in the NumPy array.

**7.2 Pandas**

Pandas is mainly used for data analysis. Pandas allow importing data from various file formats such as comma-separated-values, JSON, SQL, and Microsoft Excel. Pandas allow various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

**7.2.1 Description**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three clause BSD license.

Pandas stand for “Python Data Analysis Library.” According to the Wikipedia

page on Pandas, “the name is derived from the term “panel data,” an econometrics term for

multidimensional structured data sets.”

**7.2.2 Function**

1. Fast and efficient Data Frame object with default and customized indexing.
2. Tools for loading data into in-memory data objects from different file formats.
3. Data alignment and integrated handling of missing data.
4. Reshaping and pivoting of date sets.
5. Label-based slicing, indexing and subsetting of large data sets.
6. Columns from a data structure can be deleted or inserted.
7. Group by data for aggregation and transformations.
8. High performance merging and joining of data.
9. Time Series functionality.
   * 1. **Installation**
10. If you use conda, you can install Pandas from the defaults or conda-forge channels:

conda create -n my-env

conda activate my-env

conda config –env –add channels conda-forge

conda install pandas

1. If you use pip, you can install Pandas with:

pip install pandas

* + 1. **Use in the project**

Data I/O:

We used a .csv file, to load the data of the file we have to use the Dataframe object of pandas.

Data preview:

**7.3 Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt.

**7.3.1 Description**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

**7.3.2 Function**

1. Develop publication quality plots with just a few lines of code.
2. Use interactive figures that can zoom, pan, update etc.
3. Take full control of line styles, font properties, axes properties.
4. Export and embed to a number of file formats and interactive environments.

**7.3.3 Installation**

1. If you use conda, you can install Matplotlib from the defaults or conda-forge channels:

conda create -n my-env

conda activate my-env

conda config –env –add channels conda-forge

conda install matplotlib

1. If you use pip, you can install Matplotlib with:

pip install matplotlib

**7.3.4 Use in the Project**

In the project Matplotlib is used for drawing insights of the given features and the diseases. For e.g.: Effect of the different thorax diseases on Group of people (by age).

**7.4 Util**

Python Utils is a collection of small Python functions and classes which make common patterns shorter and easier.

**7.4.1 Description**

Utility Class, also known as Helper class, is a class, which contains just static

methods, it is stateless and cannot be instantiated. It contains a bunch of related methods, so

they can be reused across the application.

**7.4.2 Function**

This module makes it easy to execute common tasks in Python scripts such as converting text to numbers and making sure a string is in unicode or bytes format.

**7.4.3 Installation**

The package can be installed through pip (this is the recommended method):

pip install python-utils

**7.4.4 Use in the Project**

We have used utils to extract numbers from strings from the age field.

**7.5 Seaborn**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**7.5.1 Description**

.

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas’ data structures. Behind the scenes, seaborn uses matplotlib to draw its plots.

**7.5.2 Function**

Here is some of the functionality that seaborn offers:

1. A dataset-oriented API for examining relationships between multiple variables
2. Specialized support for using categorical variables to show observations or aggregate statistics.
3. Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data.
4. Automatic estimation and plotting of linear regression models for different kinds
5. dependent variables
6. Convenient views onto the overall structure of complex datasets
7. High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
8. Concise control over matplotlib figure styling with several built-in themes
9. Tools for choosing color palettes that faithfully reveal patterns in your data

**7.5.3 Installation**

1. Official releases of seaborn can be installed from PyPI:

pip install seaborn

1. The library is also included as part of the Anaconda distribution:

conda install seaborn

**7.5.4 Use in the Project**

We have used the seaborn library to plot a boxplot. This boxplot checks for outliers in our dataset.

**7.7 Keras**

Keras is an open-source deep learning library written in Python. The project was started in 2015 by Francois Chollet. It quickly became a popular framework for developers, becoming one of, if not the most, popular deep learning libraries.

**7.7.1 Description**

Keras is popular because the API was clean and simple, allowing standard deep learning models to be defined, fit, and evaluated in just a few lines of code. A secondary reason Keras took-off was because it allowed you to use any one among the range of popular deep learning mathematical libraries as the backend (e.g., used to perform the computation), such as TensorFlow, Theano, and later, CNTK. This allowed the power of these libraries to be harnessed (e.g., GPUs) with a very clean and simple interface.

**7.7.2 Function**

The Keras functional API provides a more flexible way for defining models.

1. It specifically allows you to define multiple input or output models as well as models that share layers. More than that, it allows you to define ad hoc acyclic network graphs.
2. Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a Model that specifies the layers to act as the input and output to the model.

**7.7.3 Installation**

Virtualenv is used to manage Python packages for different projects. This will be helpful to avoid breaking the packages installed in the other environments. So, it is always recommended to use a virtual environment while developing Python applications.

* **Linux/Mac OS**

Linux or mac OS users, go to your project root directory and type the below command to create virtual environment,

python3 -m venv kerasenv

After executing the above command, “kerasenv” directory is created with bin, lib and include folders in your installation location.

* **Windows**

py -m venv keras

Now, activate the environment..

* **Linux/Mac OS**

Now we have created a virtual environment named “kerasvenv”.

Move to the folder and type the below command,

$ cd kerasvenv kerasvenv

* **Windows**

Windows users move inside the “kerasenv” folder and type the below command,

$ source bin/activate.\env\Scripts\activate

Keras depends on the following python libraries:

1. NumPy
2. Pandas
3. Scikit-learn
4. Matplotlib
5. SciPy
6. Seaborn

**7.7.4 Use in the Project**

To train our model we need Dense, Flatten and conv2d layers which all fall under the library keras. We have also used keras to save the deep learning model. We have also used keras to load the saved model which is later used during deployment of our model in tkinter.

**7.7 Warnings in Python**

Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn’t warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.

Python programmers issue warnings by calling the warn() function defined in this module. (C programmers use PyErr\_WarnEx(); see Exception Handling for details).

Warning messages are normally written to sys.stderr, but their disposition can be changed flexibly, from ignoring all warnings to turning them into exceptions. The disposition of warnings can vary based on the warning category, the text of the warning message, and the source location where it is issued. Repetitions of a particular warning for the same source location are typically suppressed.

There are two stages in warning control: first, each time a warning is issued, a determination is made whether a message should be issued or not; next, if a message is to be issued, it is formatted and printed using a user-settable hook.

The determination whether to issue a warning message is controlled by the warning filter, which is a sequence of matching rules and actions. Rules can be added to the filter by calling filterwarnings() and reset to its default state by calling resetwarnings().

The printing of warning messages is done by calling showwarning(), which may be overridden; the default implementation of this function formats the message by calling formatwarning(), which is also available for use by custom implementations

**7.7.1 Warning Categories**

In Python there are a variety of built-in exceptions which reflect categories of warning, some of them are:

1. **Warning Class:** It is the super class of all warning category classes and a subclass of the Exception class.
2. **UserWarning Class:** warn() function default category.\
3. **DeprecationWarning Class:** Base category for alerts regarding obsolete features when those warnings are for other developers (triggered by code in \_\_main\_\_ unless ignored).
4. **SyntaxWarning Class:** Base class for warnings of suspicious syntactic attributes.
5. **RuntimeWarning Class:** Base class for warnings of suspicious run time attributes.
6. **FutureWarning Class:** Base class for warnings on obsolete features when certain warnings are meant for end-users of Python-written programs.
7. **PendingDeprecationWarning Class:** Base class for warnings of an outdated attribute.
8. **ImportWarning Class:** Base class for warnings caused during a module importation process.
9. **UnicodeWarning Class:** Base class for Unicode based warnings.
10. **BytesWarning Class:** Base class for bytes and bytearray based warnings.
11. **ResourceWarning Class:** Base class for resource-related warnings.

**7.8 Statsmodels**

Statsmodel is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at statsmodels.org.

The models module of scipy.stats was originally written by Jonathan Taylor. For some time it was part of scipy but was later removed. During the Google Summer of Code 2009, statsmodels was corrected, tested, improved and released as a new package. Since then, the statsmodels development team has continued to add new models, plotting tools, and statistical methods.

Most results have been verified with at least one other statistical package: R, Stata or SAS. The guiding principle for the initial rewrite and for continued development is that all numbers have to be verified. Some statistical methods are tested with Monte Carlo studies. While we strive to follow this test-driven approach, there is no guarantee that the code is bug-free and always works. Some auxiliary function are still insufficiently tested, some edge cases might not be correctly taken into account, and the possibility of numerical problems is inherent to many of the statistical models. We especially appreciate any help and reports for these kind of problems so we can keep improving the existing models.

The existing models are mostly settled in their user interface and we do not expect many large changes going forward. For the existing code, although there is no guarantee yet on API stability, we have long deprecation periods in all but very special cases, and we try to keep changes that require adjustments by existing users to a minimal level. For newer models we might adjust the user interface as we gain more experience and obtain feedback. These changes will always be noted in our release notes available in the documentation.

If you encounter a bug or an unexpected behaviour, please report it on the issue tracker. Use the show versions command to list the installed versions of statsmodels and its dependencies.

**7.8.1 Features**

Statsmodels is a popular library in Python that enables us to estimate and analyze various statistical models. It is built on numeric and scientific libraries like NumPy and SciPy. Some of the essential features of this package are-

1. It includes various models of linear regression like ordinary least squares, generalized least squares, weighted least squares, etc.
2. It provides some efficient functions for time series analysis.
3. It also has some datasets for examples and testing.
4. Models based on survival analysis are also available.
5. All the statistical tests that we can imagine for data on a large scale are present.

**7.9 Scikit-Learn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Scikit-learn, first developed as a Google Summer of Code project in 2007, is the now widely considered to be the most popular Python library for machine learning.

There are a number of reasons why this library is seen as one of the best choices for machine learning projects, especially in production systems. These include, but aren’t limited to the following:

1. It has a high level of support and strict governance for the development of the library which means that it is an incredibly robust tool.
2. There is a clear, consistent code style which ensures that your machine learning code is easy to understand and reproducible, and also vastly lowers the barrier to entry for coding machine learning models.
3. It is widely supported by third-party tools so it is possible to enrich the functionality to suit a range of use cases.

If you are learning machine learning then Scikit-learn is probably the best library to start with. Its simplicity means that it is fairly easy to pick up and by learning how to use it you will also gain a good grasp of the key steps in a typical machine learning workflow.

**7.9.1 Features**

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows:

1. **Supervised Learning algorithms:** Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
2. **Unsupervised Learning algorithms:** On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
3. **Clustering:** This model is used for grouping unlabeled data.
4. **Cross Validation:** It is used to check the accuracy of supervised models on unseen data.
5. **Dimensionality Reduction:** It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
6. **Ensemble methods:** As name suggest, it is used for combining the predictions of multiple supervised models.
7. **Feature extraction:** It is used to extract the features from data to define the attributes in image and text data.
8. **Feature selection:** It is used to identify useful attributes to create supervised models.
9. **Open Source:** It is open source library and also commercially usable under BSD license.