Chapter-4

Importing Data Visualization libraries in python

```
In [1]: import sklearn
        import pandas as pd
        import numpy as np
        import plotly as plot
        import plotly.express as px
        import plotly.graph_objs as go
        import cufflinks as cf
        import matplotlib.pyplot as plt
        import seaborn as sns
        import os
        from sklearn.metrics import accuracy_score
        import plotly.offline as pyo
        from plotly.offline import init_notebook_mode,plot,iplot
```

- import pandas as pd

Pandas is a library that we all have to install in our machine locally during python installation or we have to use **pip install pandas.** Simply import the library the current namespace, but rather than using name 'pandas' it can be instructed as pd for our convenience.

- import numpy as np

NumPy is a general Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object. It is a general purpose array-processing package.

- import matplotlib.pyplot as plt

matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- import sklearn

sklearn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Scikit-learn comes loaded with a lot of features such as Supervised learning algorithms, Cross-validation, Unsupervised learning algorithms, Various toy datasets, Feature extraction

- import plotly as plot

Plotly is known for developing and providing online analytics, statistics and graphing tools for individuals or companies. It also develops/provides scientific graphing libraries for Arduino, Julia, MATLAB, Perl, Python, R and REST.

- import plotly.express as px & import plotly.graph_objs as go
- import plotly.offline as pyo
- from plotly.offline import init_notebook_mode,plot,iplot

Plotly Express is a new high-level Python visualization library: it's a wrapper for Plotly.py that exposes a simple syntax for complex charts. It was specifically designed to have a terse, consistent and easy-to-learn API: with just a single import, you can make richly interactive plots in just a single function call, including faceting, maps, animations, and trendlines. It comes with on-board datasets, color scales and themes, and just like Plotly.py, Plotly Express is totally free: with its permissive open-source MIT license, you can use it however you like. Best of all, Plotly Express is fully compatible with the rest of Plotly ecosystem: use it in your Dash apps, export your figures to almost any file format using Orca, or edit them in a GUI with the Jupyter Lab Chart Editor!

- import cufflinks as cf

Cufflinks is another library that connects the Pandas data frame with Plotly enabling users to create visualizations directly from Pandas. The library binds the power of Plotly with the flexibility of Pandas for easy plotting.

- import seaborn as sns

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. The main idea of Seaborn is that it provides high-level commands to create a variety of plot types useful for statistical data exploration, and even some statistical model fitting.

- import os

Since the os module provides access to system features, the actual options that are available will depend on the underlying operating system. Many of the os options are available for all operating systems, but there are several which are available for UNIX (and most UNIX-based systems, including Linux and the Mac OS), but not for Windows, and some which are available only for certain flavors of UNIX. Since the available commands are system-dependent, when you create a Python program that uses os commands, you will probably want to include a check to find out which os module you're dealing with, using the os.name command

All the procedure will be discussed in next chapter.