Assignment No. 04

Performance	Understanding	Regularity	Total	Dated Sign of Subject Teacher
03	01	01	05	

Date of Performance: ..... **Date of Completion**: .....

Title: Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)

Objectives:

To understand different data visualization techniques for Big Data.

Problem Statement:

Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)

Outcomes:

Students will be able to,

1. Apply the Analytical concept of Big data using Python.

Software and Hardware requirements:

1. Software: Ubuntu OS, Anaconda, Jupyter Notebook
2. Hardware: Processor, Ethernet Connection or WiFi, RAM 1GB, HDD, Sound Card, camera, microphone (depending upon website selection)

Theory:

Data Visualisation in Python using Matplotlib and Seaborn:

Data visualization is an easier way of presenting the data, however complex it is, to analyze trends and relationships amongst variables with the help of pictorial representation.

The following are the advantages of Data Visualization

- Easier representation of compels data
- Highlights good and bad performing areas
- Explores relationship between data points
- Identifies data patterns even for larger data points

While building visualization, it is always a good practice to keep some below mentioned points in mind

- Ensure appropriate usage of shapes, colors, and size while building visualization
- Plots/graphs using a co-ordinate system are more pronounced
- Knowledge of suitable plot with respect to the data types brings more clarity to the information
- Usage of labels, titles, legends and pointers passes seamless information the wider audience

Python Libraries:

1. **Matplotlib:**

   It is an amazing visualization library in Python for 2D plots of arrays, It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. Let's try to understand some of the benefits and features of matplotlib

   - It's fast, efficient as it is based on numpy and also easier to build
   - Has undergone a lot of improvements from the open source community since inception and hence a better library having advanced features as well
   - Well maintained visualization output with high quality graphics draws a lot of users to it
   - Basic as well as advanced charts could be very easily built
   - From the users/developers point of view, since it has a large community support, resolving issues and debugging becomes much easier

2. **Seaborn:**

   Conceptualized and built originally at the Stanford University, this library sits on top of matplotlib. In a sense, it has some flavors of matplotlib while from the visualization point, it is much better than matplotlib and has added features as well. Below are its advantages

   - Built-in themes aid better visualization
   - Statistical functions aiding better data insights
   - Better aesthetics and built-in plots
   - Helpful documentation with effective examples

Nature of Visualization:

Depending on the number of variables used for plotting the visualization and the type of variables, there could be different types of charts which we could use to understand the relationship. Based on the count of variables, we could have

- Univariate plot(involves only one variable)
- Bivariate plot(more than one variable in required)

A Univariate plot could be for a continuous variable to understand the spread and distribution of the variable while for a discrete variable it could tell us the count

Similarly, a Bivariate plot for continuous variable could display essential statistic like correlation, for a continuous versus discrete variable could lead us to very important conclusions like understanding data distribution across different levels of a categorical variable. A bivariate plot between two discrete variables could also be developed.

1. Scatter Plot:

Scatter plots or scatter graphs is a bivariate plot having greater resemblance to line graphs in the way they are built. A line graph uses a line on an X-Y axis to plot a continuous function, while a scatter plot relies on dots to represent individual pieces of data. These plots are very useful to see if two variables are correlated. Scatter plot could be 2 dimensional or 3 dimensional.

Syntax: `seaborn.scatterplot(x=None, y=None)`

Parameters:
x, y: Input data variables that should be numeric.

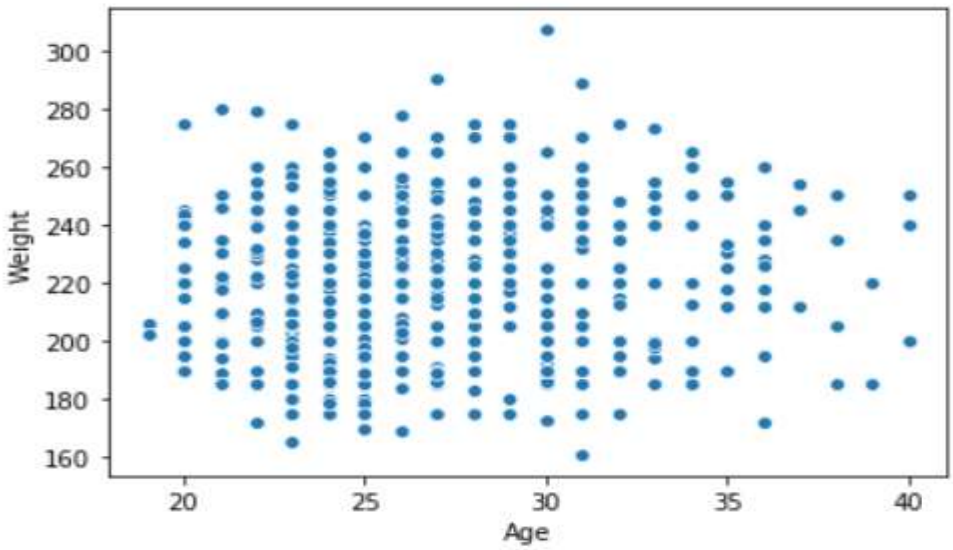
Returns: This method returns the Axes object with the plot drawn onto it.

Example:

```
# import module
import seaborn
import pandas

# load csv
data = pandas.read_csv("nba.csv")

# plotting
seaborn.scatterplot(data['Age'],data['Weight'])
```



2. Histogram:

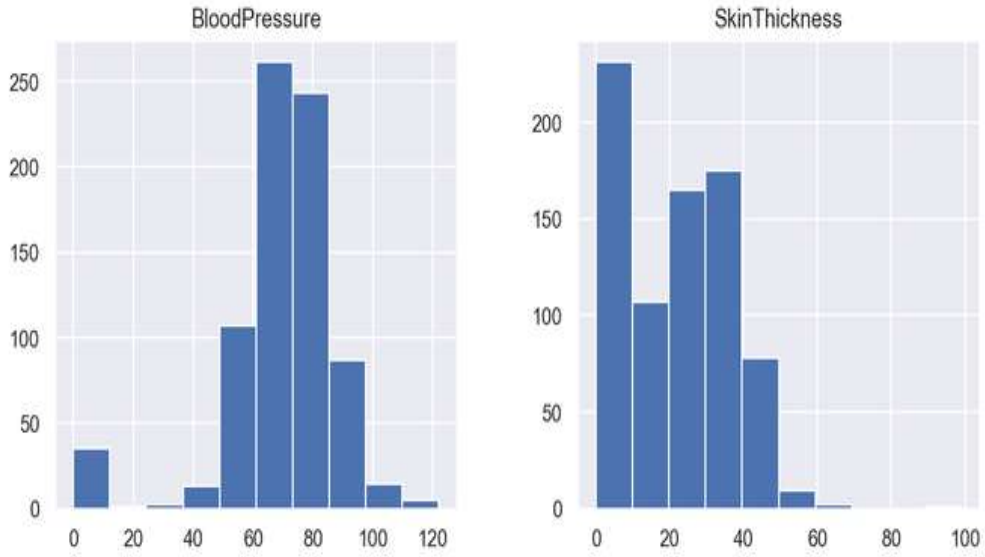
Histograms display counts of data and are hence similar to a bar chart. A histogram plot can also tell us how close a data distribution is to a normal curve. While working out statistical method, it is very important that we have a data which is normally or close to a normal distribution. However, histograms are *univariate* in nature and bar charts *bivariate*.

A bar graph charts actual counts against categories e.g. height of the bar indicates the number of items in that category whereas a histogram displays the same categorical variables in *bins*.

Bins are integral part while building a histogram they control the data points which are within a range. As a widely accepted choice we usually limit bin to a size of 5-20, however this is totally governed by the data points which is present.

Syntax:

```
features = ['BloodPressure', 'SkinThickness']
diabetes[features].hist(figsize=(10, 4))
```



3. Pie Chart:

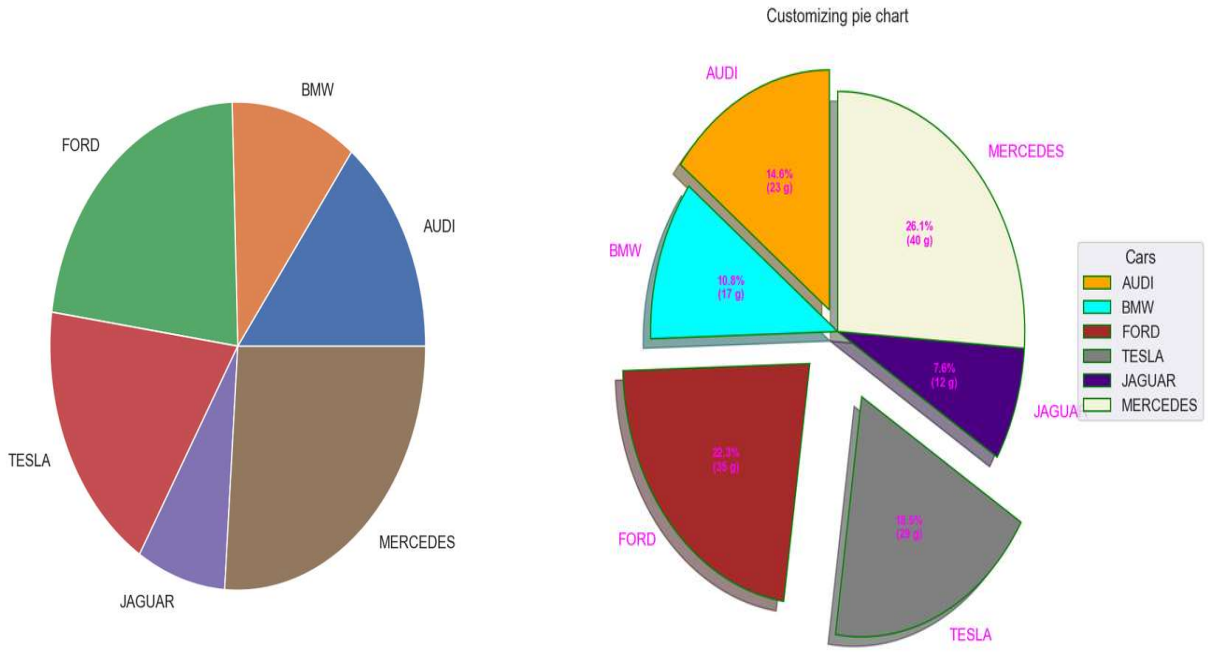
Pie chart is a *univariate* analysis and is typically used to show percentage or proportional data. The percentage distribution of each class in a variable is provided next to the corresponding slice of the pie. The python libraries which could be used to build a pie chart is *matplotlib* and *seaborn.*

Syntax:

```
matplotlib.pyplot.pie(data, explode=None, labels=None, colors=None,
autopct=None, shadow=False)
```

Parameters:

- **data** represents the array of data values to be plotted, the fractional area of each slice is represented by **data/sum(data)**. If sum(data)<1, then the data values returns the fractional area directly, thus resulting pie will have empty wedge of size 1-sum(data).
- **labels** is a list of sequence of strings which sets the label of each wedge.
- **color** attribute is used to provide color to the wedges.
- **autopct** is a string used to label the wedge with their numerical value.
- **shadow** is used to create shadow of wedge.



4. Line Plot:

Lineplot Is the most popular plot to draw a relationship between x and y with the possibility of several semantic groupings.

Syntax : *sns.lineplot(x=None, y=None)*

Parameters:

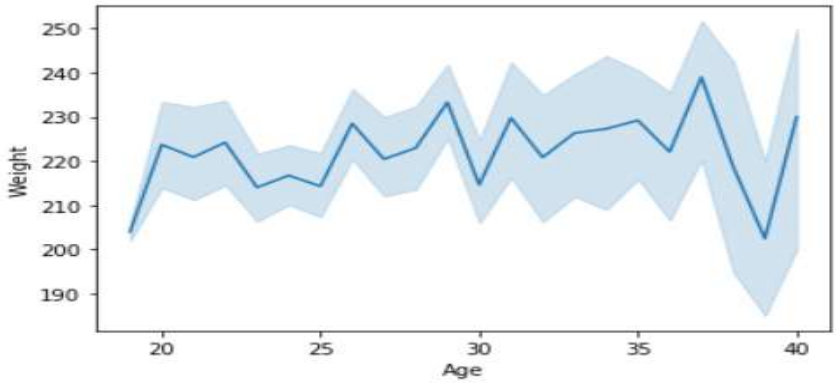
x, y: *Input data variables; must be numeric. Can pass data directly or reference columns in data.*

Example:

```
# import module
import seaborn as sns
import pandas

# loading csv
data = pandas.read_csv("nba.csv")

# plotting lineplot
sns.lineplot( data['Age'], data['Weight'])
```



5. Bar Plot:

Barplot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.

Syntax: `seaborn.barplot(x=None, y=None, hue=None, data=None)`

Parameters:

- **x, y :** This parameter take names of variables in data or vector data, Inputs for plotting long-form data.
- **hue :** (optional) This parameter take column name for colour encoding.
- **data :** (optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.

Returns : Returns the Axes object with the plot drawn onto it.

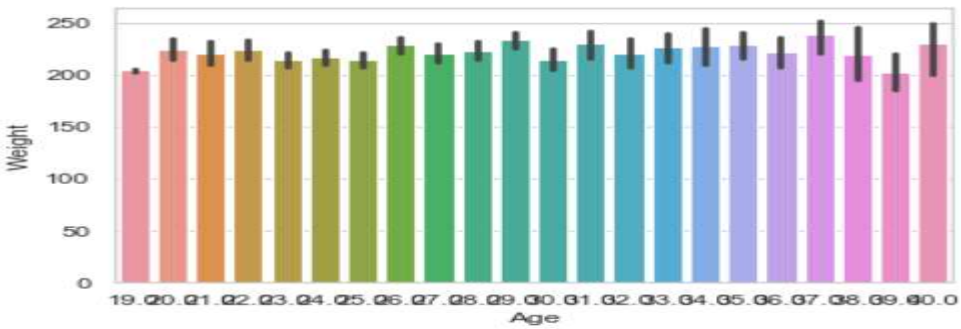
Example:

```
# import module

import seaborn

seaborn.set(style = 'whitegrid')

# read csv and plot
data = pandas.read_csv("nba.csv")
seaborn.barplot(x ="Age", y ="Weight", data = data)
```



6. Box Plot:

A box plot (or box-and-whisker plot) s is the visual representation of the depicting groups of numerical data through their quartiles against continuous/categorical data.

A box plot consists of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

Syntax: `seaborn.boxplot(x=None, y=None, hue=None, data=None)`

Parameters:

- x, y, hue: Inputs for plotting long-form data.
- data: Dataset for plotting. If x and y are absent, this is interpreted as wide-form.

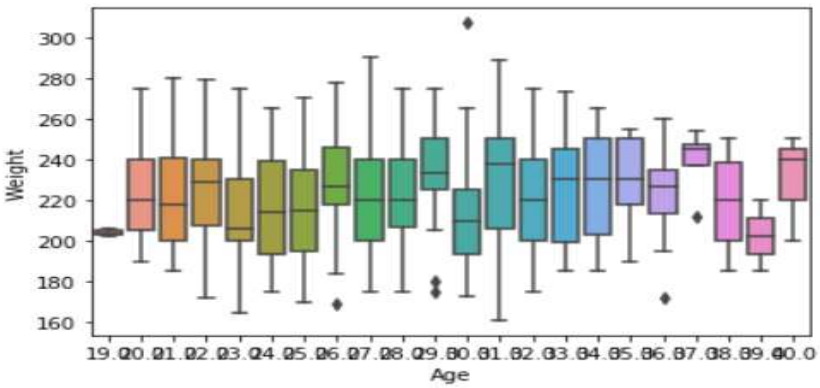
Returns: It returns the Axes object with the plot drawn onto it.

Example:

```
# import module

import seaborn as sns
import pandas

# read csv and plotting
data = pandas.read_csv( "nba.csv" )
sns.boxplot( data['Age'], data['Weight'])
```



Conclusion: Thus we have learnt Visualize the data using R/Python by plotting the graphs.

FAQ:

1) How to create a Histogram?
2) How to create a Bar Chart?
3) How to create a Stacked Bar Chart?
4) How to create a Box Plot?
5) How to plot the entire data in a single command?
6) How to create an Area Chart?