Experiment No - 8

Aim - Data Visualization: Use Matplotlib and seabom Python libraries for visualization.

Objective:- To understand and apply the Matplotlib and seabom python libraries for visualization using python.

Description-

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 oflabels, titles, legends and pointers passes seamless information the wider audience

Python Libraries
There are a lot of python libraries which could be used to build visualization like matplotlib, vispy, bokeh, seabom, pygal, folium, plotly, cufflinks, and networkx. Of the many, matplotlib and seabom seems to be very widely used for basic to intermediate level of visualizations.

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

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 o n the count of variables , we could have

1. Univariate plot(involves only onevariable)
2. 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.

Scatter Plot

Scatter plots or scatter graphs is abivariate 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, hue=None , style=None, size=None, data=None , palette=None, hue_order=None , hue_norm=None , sizes=None, size_order=None, size_norm=None, markers=True , style_order=None, x_bins=None , y_bins=None , units=None, e stimator=None, ci=95, n_boot= lOOO, alpha= 'auto ', xjitter=None , yjitter=None , legend='brief ', ax=None , **kwargs)

Parameters:

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

data: Dataframe where each column is avariable and each row is an observation. size: Grouping variable that will produce points with different sizes.

style: Grouping variable that will produce points with different markers. palette: Grouping variable that will produce points with different markers.

markers: Object determining how to draw the markers for different levels. alpha: Proportional opacity ofthe points.

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

Histograms:

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.

Countplot

A countplot is a plot between a categorical and a continuous variable. The continuous variable in this case being the number of times the categorical is present or simply the frequency. In a sense, count plot can be said to be closely linked to a histogram or a bar graph.

Syntax : seabom.countplot(x=None , y=None , hue=None , data=None , order=None , hue_order=None , orient=None , color=None, palette=None , saturation=0.75 , dodge=True , ax=None, **kwargs)

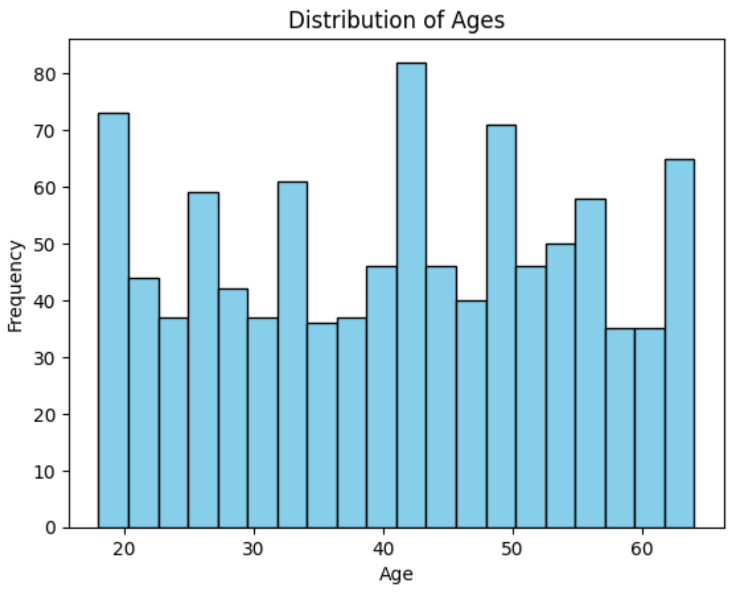
Parameters : This method is accepting the following parameters that are described below:

- x, y: This parameter take names of variables in data or vector data, optional, 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.
- Ifx and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.
- order, hue_order : (optional) This parameter take lists of strings. Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.
- orient : (optional)This parameter take "v" I "h", Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables but can be used to specify when the "categorical " variable is a numeric or when plotting wide-form data.
- color : (optional) This parameter take matplotlib color, Color for all of the elements, or seed for a gradient palette.
- palette : (optional) This parameter take palette name, list, or diet, Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette() , or a dictionary mapping hue levels to matplotlib colors.

Program:

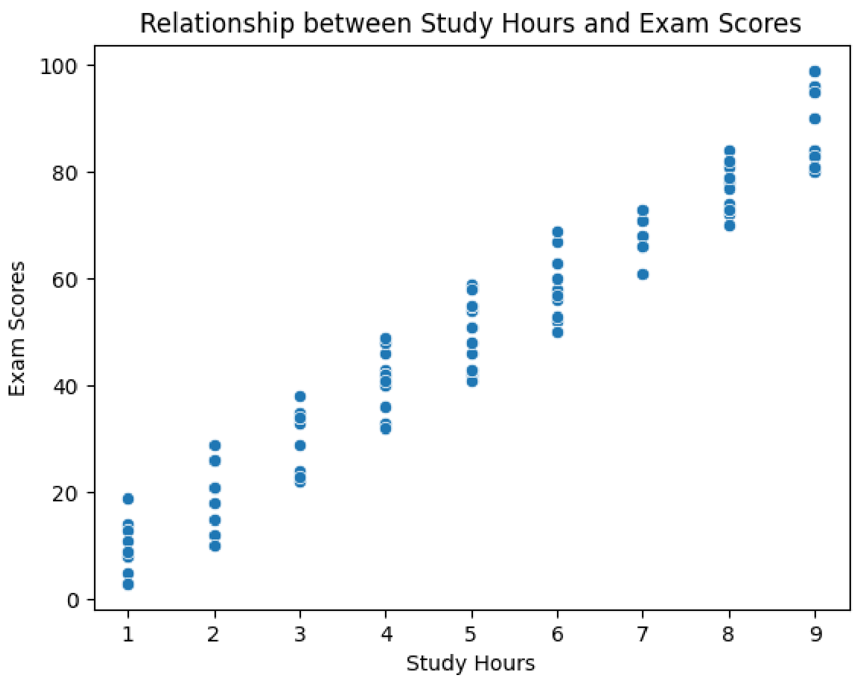
1. **Using Matplotlib**

```
import matplotlib.pyplot as plt

import numpy as np

# Generating random ages for a sample dataset

np.random.seed(42)

ages = np.random.randint(18, 65, size=1000)

# Creating histogram

plt.hist(ages, bins=20, color='skyblue', edgecolor='black')

# Adding labels and title

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.title('Distribution of Ages')

# Display the plot

plt.show()
```



2. **Using Seaborn**

```
import seaborn as sns

import pandas as pd

np.random.seed(0)

study_hours = np.random.randint(1, 10, size=100)

exam_scores = study_hours * 10 + np.random.randint(-10, 10, size=100)

df = pd.DataFrame({'Study Hours': study_hours, 'Exam Scores': exam_scores})

sns.scatterplot(x='Study Hours', y='Exam Scores', data=df)

# Adding labels and title

plt.xlabel('Study Hours')

plt.ylabel('Exam Scores')

plt.title('Relationship between Study Hours and Exam Scores')

# Display the plot

plt.show()
```



Conclusion:

1. **What is the difference between matplotlib and seaborn?**

1) **Matplotlib:**
   a) Provides basic functionality for creating plots and visualizations.
   b) Requires more code to create customized visualizations.
   c) Primarily focuses on 2D plotting.

2) **Seaborn:**
   a) Built on top of Matplotlib, providing a high-level interface for creating attractive and informative statistical graphics.
   b) Offers simpler syntax and more aesthetically pleasing default styles.
   c) Designed for creating complex statistical visualizations with fewer lines of code.

2. **Which library is used to create statistical graphics in Python?**
   Seaborn is the library used to create statistical graphics in Python. It provides high-level functions for creating informative and visually appealing statistical plots.

3. **Which function is used to create a histogram in Seaborn?**
   The seaborn.histplot() function is used to create a histogram in Seaborn. It provides options for customizing the appearance of the histogram, including the number of bins, color, and transparency.