Matplotlib

Notes by Mannan Ul Haq (BDS-3C)

Matplotlib is a popular Python library for creating 2D and 3D plots and visualizations. It provides a versatile and flexible platform for creating a wide range of static, animated, or interactive plots and charts. Matplotlib is widely used in data analysis, scientific research, and data visualization tasks.

Creating Histograms with Matplotlib

1. Import Necessary Libraries:

- Import `matplotlib.pyplot` as `plt` for creating visualizations.
- Import other required libraries like `pandas` and `numpy` for data manipulation.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2. Load Data:

- Use `pd.read_csv()` to load your dataset into a Pandas DataFrame.
- Extract the specific data column you want to visualize, such as **"SepalLengthCm"**.

```
File_Data = pd.read_csv("Iris.csv")
Data = File_Data["SepalLengthCm"]
```

3. Create the Histogram:

- Use `plt.hist()` to create a histogram. Specify the data, the number of bins, and optional styling parameters.
- `bins` parameter determines the number of bins or intervals to divide your data into. In this case, `int(np.sqrt(len(Data)))` calculates a suitable number of bins based on the data length.

```
plt.hist(Data, bins=int(np.sqrt(len(Data))), color="lightblue", edgecolor="black")
```

4. Customize the Plot:

- Add a title to your histogram using `plt.title("Distribution of Sepal Length")`.
- Label the x and y axes with `plt.xlabel("Sepal Length (cm)")` and `plt.ylabel("Frequency")`.
- You can customize the colors and appearance as needed.

5. Display the Plot:

- Use `plt.show()` to display the histogram.

6. Interpretation:

- The histogram visually represents the **distribution** of the **"Sepal Length (cm)"** data.
- It shows how the data is distributed across different intervals (bins).
- You can observe the **frequency** (number of occurrences) of data points within each interval.

Creating Box Plots with Matplotlib

Box plots are effective tools for visualizing the spread and distribution of data, as well as identifying potential outliers. This guide will walk you through the process of creating box plots using Matplotlib, a popular Python data visualization library.

1. Import Necessary Libraries:

Before you start, make sure you have the required libraries imported:

```
import pandas as pd
import matplotlib.pyplot as plt
```

- `pandas` for data manipulation.
- `matplotlib.pyplot` as `plt` for creating visualizations.

2. Load Data:

Load your dataset into a Pandas DataFrame. In this example, we'll use the "Iris.csv" dataset and focus on a specific data column, such as **"SepalLengthCm"**.

```
# Load the dataset (replace 'Iris.csv' with your dataset)
data = pd.read_csv("Iris.csv")

# Extract the data column you want to visualize
sepal_length = data["SepalLengthCm"]
```

3. Create the Box Plot:

Use `plt.boxplot()` to create a box plot of your data. Specify the data you want to visualize, and you can add optional styling parameters if needed.

```
plt.boxplot(sepal_length, vert=False, patch_artist=True)
```

- `vert=False` makes the box plot horizontal (optional, use `vert=True` for vertical).
- `patch_artist=True` fills the boxes with colors (optional, you can remove this parameter or set it to `False` if you prefer no filling).

4. Customize the Plot:

Enhance your box plot by customizing its appearance:

- Add a title to the plot for clarity using `plt.title("Box Plot of Sepal Length")`.
- Label the x-axis with `plt.xlabel("Sepal Length (cm)")`.
- You can further customize colors, styles, and other parameters as per your preferences.

5. Display the Plot:

Finally, display your customized box plot:

```
plt.show()
```

6. Interpretation:

- The box plot provides a visual representation of the **spread** and **distribution** of the **"Sepal Length (cm)"** data.
- It shows the **quartiles (box)**, **median (line inside the box)**, and **potential outliers (individual data points beyond the whiskers)**.
- Use the box plot to gain insights into the **central tendency**, **variability**, and **presence of outliers in your data**.

Creating Scatter Plots with Matplotlib

1. Import Necessary Libraries:

- Import `matplotlib.pyplot` as `plt` for creating visualizations.
- Import other required libraries like `pandas` and `numpy` for data manipulation.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

2. Load Data:

- Use `pd.read_csv()` to load your dataset into a Pandas DataFrame.
- Extract the specific data columns you want to visualize, such as **"Feature1"** and **"Feature2"**.

```
File_Data = pd.read_csv("YourData.csv")
X = File_Data["Feature1"]
Y = File_Data["Feature2"]
```

3. Create the Scatter Plot:

- Use `plt.scatter()` to create a scatter plot. Specify the x and y data points and customize the style as needed.
- You can use optional parameters like `color`, `marker`, and `alpha` for styling.

```
plt.scatter(X, Y, color="blue", marker="o", alpha=0.5)
```

4. Customize the Plot:

- Add a title to your scatter plot using `plt.title("Scatter Plot of Feature1 vs. Feature2")`.
- Label the x and y axes with `plt.xlabel("Feature1")` and `plt.ylabel("Feature2")`.
- You can further customize the appearance, such as changing the marker style, size, or color.

5. Display the Plot:

- Use `plt.show()` to display the scatter plot.

6. Interpretation:

- The scatter plot visually represents the relationship between **"Feature1"** and **"Feature2"**.
- Each point on the plot represents a data point, with the x-coordinate corresponding to **"Feature1"** and the y-coordinate corresponding to **"Feature2"**.
- You can observe patterns, clusters, or trends in the data points, helping to understand the relationship between the two variables.