Submitted by: Tanmay Guha **Position: Data Science Intern**

TITLE: PYTHON VISUALIZATION LIBRARIES GUIDE: MATPLOTLIB & PLOTLY

Table Of Contents:

---

1. Library Overview:

❖ Matplotlib:

➢ **Introduction:** Matplotlib is Python's foundational plotting library, first released in 2003. It provides a MATLAB-like interface for creating static, interactive, and animated visualizations.

➢ **Key Features**:

- Low-level control over every plot element
- Highly customizable
- Extensive 2D plotting capabilities
- Basic 3D support
- Works well with NumPy and Pandas

- **Typical Use Cases**:
  - Scientific publications
  - Basic to intermediate data visualization
  - When pixel-perfect control is needed
  - Embedding plots in GUI applications

❖ Plotly:

- **Introduction:** Plotly is an interactive, open-source visualization library that creates web-based visualizations that can be displayed in Jupyter notebooks or saved as standalone HTML files.
- **Key Features**:
  - Built-in interactivity (zooming, panning, hovering)
  - Web-based visualizations
  - Support for 3D charts
  - Dash integration for web apps
  - Collaborative features
- **Typical Use Cases**:
  - Interactive dashboards
  - Web applications
  - When hover tooltips are valuable
  - Collaborative data exploration

---

2. Graph Types:

❖ Matplotlib Graphs:

(a) Line Plot:

- **Description**: Shows the relationship between two variables with connected data points.
- **Use Case**: Tracking changes over time (stock prices, temperature trends).
- **Code**:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 10, 100)

y = np.sin(x)

plt.figure(figsize=(8, 4))

plt.plot(x, y, label='sin(x)', color='blue', linestyle='-')

plt.title('Sine Wave')

plt.xlabel('x')

plt.ylabel('sin(x)')

plt.legend()

plt.grid(True)

plt.show()
```

(b) Scatter Plot:

- **Description**: Displays individual data points to show correlation between variables.
- **Use Case**: Identifying relationships or clusters in data.
- **Code**:

```
import matplotlib.pyplot as plt

import numpy as np
```

```
x = np.random.rand(50)

y = np.random.rand(50)

colors = np.random.rand(50)

sizes = 1000 * np.random.rand(50)

plt.figure(figsize=(8, 6))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='viridis')

plt.colorbar()

plt.title('Bubble Chart')

plt.xlabel('X Axis')

plt.ylabel('Y Axis')

plt.show()
```

(c) Bar Chart:

- **Description**: Uses rectangular bars to represent categorical data.
- **Use Case**: Comparing quantities across different categories.
- **Code**:

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']

values = [15, 30, 45, 10]

plt.figure(figsize=(8, 6))

plt.bar(categories, values, color=['red', 'green', 'blue', 'yellow'])

plt.title('Bar Chart Example')

plt.xlabel('Categories')

plt.ylabel('Values')
plt.show()
```

(d) Histogram:

- **Description**: Shows the distribution of numerical data.
- **Use Case**: Understanding data distribution and identifying outliers.
- **Code**:

```
import matplotlib.pyplot as plt

import numpy as np

data = np.random.normal(170, 10, 250)

plt.figure(figsize=(8, 6))

plt.hist(data, bins=30, edgecolor='black', alpha=0.7)

plt.title('Height Distribution')

plt.xlabel('Height (cm)')

plt.ylabel('Frequency')

plt.show()
```

(e) Scatter Plot:

- **Description**: Circular statistical graphic divided into slices.
- **Use Case**: Showing proportions of a whole.
- **Code**:

```
import matplotlib.pyplot as plt

labels = ['A', 'B', 'C', 'D']

sizes = [15, 30, 45, 10]

explode = (0, 0.1, 0, 0)  # "explode" the 2nd slice
```

```
plt.figure(figsize=(8, 6))

plt.pie(sizes,explode=explode,labels=labels,autopct='%1.1f%%',shadow=True
, startangle=90)

plt.axis('equal')  # Equal aspect ratio ensures pie is drawn as a circle

plt.title('Pie Chart Example')

plt.show()
```

❖ Plotly Graphs:

(a) Interactive Line Plot:

- **Description**: Line plot with hover tooltips and zoom/pan functionality.
- **Use Case**: Interactive time series analysis.
- **Code:**

```
import plotly.express as px

import numpy as np

x = np.linspace(0, 10, 100)

y = np.sin(x)

fig = px.line(x=x, y=y, title='Interactive Sine Wave', labels={'x': 'x', 'y':
'sin(x)'})

fig.update_layout(hovermode='x unified')

fig.show()
```

(b) 3D Scatter Plot:

- **Description**: Three-dimensional scatter plot with rotation capability.
- **Use Case**: Visualizing multivariate relationships.

- **Code:**

```
import plotly.express as px

import numpy as np

np.random.seed(42)

x = np.random.rand(100)

y = np.random.rand(100)

z = np.random.rand(100)

fig = px.scatter_3d(x=x, y=y, z=z, color=z,title='3D Scatter Plot',labels={'x': 'X', 'y': 'Y', 'z': 'Z'})

fig.show()
```

(c) Interactive Bar Chart:

- **Description**: Bar chart with hover details and click events.
- **Use Case**: Interactive category comparisons.
- **Code:**

```
import plotly.express as px

data = {

   'Category': ['A', 'B', 'C', 'D'],

   'Value': [15, 30, 45, 10],

   'Color': ['red', 'green', 'blue', 'yellow']

}

fig = px.bar(data, x='Category', y='Value', color='Color',title='Interactive Bar Chart',hover_data=['Value'])

fig.show()
```

(d) Box Plot:

- **Description**: Displays distribution of data through quartiles.
- **Use Case**: Comparing distributions across categories.
- **Code:**

```
import plotly.express as px
import numpy as np
np.random.seed(42)
data = {
    'Group': np.repeat(['A', 'B', 'C'], 100),
    'Value': np.concatenate([
        np.random.normal(100, 10, 100),
        np.random.normal(90, 20, 100),
        np.random.normal(110, 15, 100)
    ])
}
fig = px.box(data, x='Group', y='Value', color='Group',title='Interactive Box Plot')
fig.show()
```

(e) Choropleth Map:

- **Description**: Thematic map where areas are shaded according to values.
- **Use Case**: Geographic data visualization.
- **Code:**

```
import plotly.express as px
data = px.data.gapminder().query("year == 2007")
```

```
fig=px.choropleth(data,locations="iso_alpha",color="gdpPercap",hover_name="country",color_continuous_scale=px.colors.sequential.Plasma,title='GDP per Capita (2007)')

fig.show()
```

3. Comparison:

Feature	Matplotlib	Plotly
Ease of Use	Steeper learning curve	More intuitive for basic plots
Customization	Extremely flexible	Good, but some limitations
Interactivity	Basic (requires additional code)	Built-in, rich interactivity
Performance	Excellent with large datasets	Can slow with very large datasets
Output Format	Static images (PNG, PDF, etc.)	Interactive HTML/Web
3D Support	Basic	Advanced
Integration	Works everywhere	Best for web/Jupyter environments
Learning Resources	Extensive documentation	Good docs, fewer advanced examples

❖ **When to use Matplotlib**:

- When you need pixel-perfect control
- For publication-quality static images

- When working with very large datasets
- For integration with GUI applications

❖ **When to use Plotly**:

- When interactivity is important
- For web-based dashboards and applications
- When 3D visualization is needed
- For collaborative data exploration

4. Conclusion:

Both **Matplotlib** and **Plotly** are powerful visualization tools.

- Use **Matplotlib** when you need full control over every element and prefer static, publication-quality charts.
- Use **Plotly** when you need rich interactivity, quick exploratory visualizations, or plan to embed plots in web applications.

5. Resources:

- **Matplotlib**: Official Documentation
- **Plotly**: Python Documentation
- **Seaborn**: Tutorial
- **Bokeh**: User Guide
- **Pandas**: User Guide