Visualization with Matplotlib and Seaborn

Class 6

25/2/2025

Acknowledgement

The series of the IT & Japanese language course is Supported by AOTS and OEC.

Ministry of Economy, Trade and Industry

Overseas Employment Corporation

What you have Learnt Last Week

We were focused on following points.

- Usage of control and loop flow statement
- Performing Linear Algebra in Numpy
- Concatenation, and Stacking in NumPy
- Import and export data effortlessly between different file formats.
- Inspecting and Understanding Data
- Basics of creating, loading, and exploring DataFrames
- Understanding of 1D, and 2D NumPy arrays
- Array indexing and slicing

What you will Learn Today

We will focus on following points.

- Basics of Matplotlib’s plotting library, setting up figures, and axes
- Customize your plots with various line styles, markers, and colors
- Making data visualization more engaging and informative.
- Visualize relationships and correlations with scatter plots
- Upload code on Github
- Quiz
- Q&A Session

Employee Data Management System

This project simulates an HR system that manages employee details such as Name, Age, Salary, Department, and Work Location

[Steps Involved]

- **Create DataFrames** from different sources (CSV, dictionary, list, NumPy array)
- **Select and filter** employee records based on conditions.
- **Modify DataFrame** (add new columns, delete columns, update rows).
- **Sort employees** by salary and name.
- **Handle missing values** in employee data.
- **Merge employee records** from multiple departments.
- **Save and export** the processed data.

Employee Data Management System

1. Creating a DataFrame from a dictionary

```
import pandas as pd
import numpy as np

# Creating a DataFrame from a dictionary
employees = {
    'Emp_ID': [101, 102, 103, 104, 105],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, 30, 35, np.nan, 29],
    'Salary': [50000, 60000, 75000, 80000, np.nan],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'Location': ['New York', 'San Francisco', 'Chicago', 'Los Angeles', 'New York']
}
df = pd.DataFrame(employees)
```

Employee Data Management System

2. Handling Missing Data and Filter Employees

```
#Handling Missing Data
df['Age'].fillna(df['Age'].mean(), inplace=True)  # Fill missing age with mean
df.dropna(subset=['Salary'], inplace=True)  # Drop rows with missing Salary

#Selecting Data (Filter employees in IT department)
it_employees = df[df['Department'] == 'IT’]
print("\nIT Employees:\n", it_employees)
```

Employee Data Management System

3. Sorting employees by Salary

```
#Adding a new column (Experience Level based on Salary)
df['Experience_Level'] = np.where(df['Salary'] > 70000, 'Senior', 'Junior')

# Sorting employees by Salary in descending order
df_sorted = df.sort_values(by='Salary', ascending=False)
```

Employee Data Management System

4. Creating another DataFrame for merging

```
#Creating another DataFrame for merging
new_hires = pd.DataFrame({
    'Emp_ID': [106, 107],
    'Name': ['Frank', 'Grace'],
    'Age': [27, 26],
    'Salary': [55000, 62000],
    'Department': ['HR', 'Finance'],
    'Location': ['Boston', 'Chicago']
})
```

Employee Data Management System

5. Exporting the final DataFrame to CSV

```
#Merging the new hires into the existing DataFrame
df_final = pd.concat([df, new_hires], ignore_index=True)

# Exporting the final DataFrame to CSV
df_final.to_csv('employee_data.csv', index=False)

print("\nFinal Employee Data:\n", df_final)
```

What is Matplotlib?

Matplotlib is a powerful data visualization library in Python

[Features]

- It provides 2D plotting capabilities and supports basic 3D plotting
- Used for creating static, animated, and interactive visualizations
- Works seamlessly with NumPy, Pandas, and Jupyter Notebook
- Inspired by MATLAB's plotting functions but built for Python

Why Use Matplotlib for Data Visualization?

Matplotlib is a powerful data visualization library in Python

[Why Matplotlib?]

- **Easy to Use** – Simple syntax for quick plotting.
- **Highly Customizable** – Control over colors, labels, grid, styles, and more.
- **Multiple Plot Types** – Line, bar, scatter, histogram, pie charts, etc.
- **Integration** – Works well with SciPy, Pandas, and Seaborn.
- **Interactive & Static Plots** – Export graphs as images or interactive charts.

Install and Import Matplotlib

Using Pyplot Interface (quicker plots)

[Example]

```
import matplotlib.pyplot as plt


# Creating a simple line plot
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]
plt.plot(x, y, marker='o', linestyle='-', color='b', label="Line 1")  # Line plot
plt.xlabel("X-axis Label")   # Label for x-axis
plt.ylabel("Y-axis Label")   # Label for y-axis
plt.title("Simple Line Plot")  # Title of the plot
plt.legend()  # Show legend
plt.show()  # Display the plot
```

Figure and Axes in Matplotlib

Using Object-Oriented Approach (complex plots)

```
import matplotlib.pyplot as plt

# Creating a figure with two subplots (1 row, 2 columns)
fig, ax = plt.subplots(1, 2, figsize=(10, 5))  # 1 row, 2 columns

# First subplot
ax[0].plot([1, 2, 3, 4], [10, 20, 25, 30], marker='o', linestyle='-', color='r')
ax[0].set_title("First Subplot")
ax[0].set_xlabel("X-axis")
ax[0].set_ylabel("Y-axis")

# Second subplot
ax[1].bar(["A", "B", "C", "D"], [5, 7, 3, 8], color='g')  # Bar chart
ax[1].set_title("Second Subplot")

plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

Saving Plot Figures

You can save plots as image files using plt.savefig()

[Visualizing temperature trends using line plots]

```
import pandas as pd
data = pd.read_csv('temperature_data.csv')
plt.plot(data['Month'], data['AvgTemperature'])
plt.xlabel('Month')
plt.ylabel('Temperature (°C)')
plt.title('Monthly Temperature Trends')
plt.show()
# Save as PNG with high resolution
plt.savefig("plot.png", dpi=300, bbox_inches='tight')
```

Task

You can save plots as image files using plt.savefig()

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

# Load the dataset
file_path = "temperature.csv"
data = pd.read_csv(file_path)

# Convert temperature from Fahrenheit to Celsius and
drop missing values
data["AverageTemperatureC"] =
(data["AverageTemperatureFahr"] - 32) * 5/9
data_cleaned =
data.dropna(subset=["AverageTemperatureC"])

# Group by month and calculate the average
temperature
monthly_avg_temp =
data_cleaned.groupby("month")["AverageTemperatureC
"].mean()

# Plot the temperature trend
plt.figure(figsize=(10, 5))
plt.plot(monthly_avg_temp.index,
monthly_avg_temp.values, marker='o', linestyle=
color='b')

# Labels and title
plt.xlabel("Month")
plt.ylabel("Temperature (°C)")
plt.title("Monthly Temperature Trends")

# Show and save the plot
plt.savefig("temperature_plot.png", dpi=300,
bbox_inches='tight')
plt.show()
```

Customizations Your Plots Various Line Styles, Colors

Customization by using various function

1. **Line Customization:** Different colors, line styles, and widths are used
2. **Marker Customization:** Different markers ('o' and 's'), sizes, and colors.
3. **Color Customization:** Uses named colors ('red', 'cyan'), hex ('#00A6D6'), and custom colormaps.
4. **Annotations:** The plt.annotate() function highlights a peak.
5. **Grid and Background:** plt.grid() customizes grid lines, and set_facecolor() changes the background color.

Customizations Your Plots Various Line Styles, Colors

1. Sample the Data

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

# Sample data
x = np.linspace(0, 10, 10)
y1 = np.sin(x)
y2 = np.cos(x)

plt.figure(figsize=(10, 6))
```

Customizations Your Plots Various Line Styles, Colors

2. Sample the Data

```
# Plot with different line styles, colors, and markers
plt.plot(x, y1, color='red', linestyle='-', linewidth=2, marker='o', markersize=8,
         markerfacecolor='yellow', markeredgecolor='black', label='Sine Wave')
plt.plot(x, y2, color='#00A6D6', linestyle='--', linewidth=2, marker='s', markersize=6,
         markerfacecolor='cyan', markeredgecolor='black', label='Cosine Wave')

# Adding labels and title
plt.xlabel("X-axis", fontsize=12)
plt.ylabel("Y-axis", fontsize=12)
plt.title("Customized Plot Example", fontsize=14)
```

Customizations Your Plots Various Line Styles, Colors

3. Adding a legend and annotation

```
# Adding a legend
plt.legend(loc='upper right', fontsize=10)

# Annotating a specific point
plt.annotate('Peak', xy=(1.57, 1), xytext=(3, 1.2),
             arrowprops=dict(facecolor='black', shrink=0.05))
```

Customizations Your Plots Various Line Styles, Colors

4. Adding grid and change background color

```
# Adding a grid with customization
plt.grid(color='gray', linestyle=':', linewidth=0.5)

# Changing the background color
plt.gca().set_facecolor('whitesmoke')

# Show the plot
plt.show()
```

Visualizing Relationships and Correlations with Scatter Plots

A scatter plot is used to visualize the relationship between two numerical variables

[Why Scatter Plot]

1. Show Relationships Between Variables
2. Detect Patterns & Trends
3. Identify Outliers
4. Compare Multiple Datasets
5. Basis for Further Analysis

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

# Sample data
x = np.random.rand(50)
y = np.random.rand(50)

# Create scatter plot
plt.scatter(x, y)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Basic Scatter Plot")
plt.show()
```

Customizing Scatter Plots

Changing Colors Based on Categories

```
# Assigning categories randomly
categories = np.random.randint(0, 3, 50)  # Three categories: 0, 1, 2
colors = ['red', 'blue', 'green']

plt.scatter(x, y, c=[colors[i] for i in categories])
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Categorical Colors")
plt.show()
```

Customizing Scatter Plots

Using Different Marker Styles

```
markers = ['o', 's', 'D']  # Circle, Square, Diamond
for i, marker in enumerate(markers):
    plt.scatter(x[categories == i], y[categories == i], marker=marker, label=f'Category {i}')

plt.legend()
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Different Marker Styles")
plt.show()
```

Customizing Scatter Plots

Adjusting Transparency (alpha)

```
plt.scatter(x, y, alpha=0.5, color='purple')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Transparency")
plt.show()
```

Adding Size Variations

Using Data-Driven Marker Sizes

```
sizes = np.random.rand(50) * 500  # Scale up marker sizes

plt.scatter(x, y, s=sizes, alpha=0.5, color='orange')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Varying Marker Sizes")
plt.show()
```

Adding Size Variations

Scaling Markers Dynamically

```
z = np.random.rand(50) * 100  # Another variable for scaling

plt.scatter(x, y, s=z, alpha=0.5, c=z, cmap='coolwarm')
plt.colorbar(label="Size Scaling Variable")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Dynamic Scaling and Colormap")
plt.show()
```

Colormap (cmap=coolwarm)

- **Low z_{values} → Dark blue.**
- **High z_{values} → Dark red.**

Colormap and Colorbar

A colormap is used in scatter plots to represent an additional dimension of data (e.g., intensity, magnitude)

```
plt.scatter(x, y, c=z, cmap='viridis', s=100, alpha=0.7)
plt.colorbar(label="Color Intensity Based on Z")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Colormap and Colorbar")
plt.show()
```

Colormap (cmap='viridis)

- **Low z_{values} → Dark purple/blue.**
- **Mid z_{values} → Green.**
- **High z_{values} → Yellow.**

Displaying Regression Trends

used for polynomial regression, which means fitting a curve to data points

```
# Generate a linear relationship
x = np.linspace(0, 10, 50)
y = 3*x + np.random.randn(50) * 5  # y = 3x + noise

plt.scatter(x, y, color='blue', label="Data")

# Fit a linear trend line
m, b = np.polyfit(x, y, 1)  # First-degree polynomial fit
plt.plot(x, m*x + b, color='red', label="Trend Line")

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.title("Scatter Plot with Regression Trend Line")
plt.show()
```

Why Seaborn?

Using Seaborn for creating beautiful, informative, and statistical visualizations

[Why Seaborn?]

1. High-Level Interface for Statistical Graphics
2. Built-in Themes for Better Aesthetics
3. Simplifies Complex Visualizations
4. Integrates Well with Pandas and NumPy
5. Supports Multiple Plot Types
6. Handles Large Datasets Efficiently
7. Enhances Matplotlib with Additional Features

Displaying Regression Trends

Using Seaborn for creating beautiful, informative, and statistical visualizations

```
import seaborn as sns

sns.regplot(x=x, y=y, scatter_kws={'color':'blue'}, line_kws={'color':'red'})
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot with Seaborn Regression")
plt.show()
```

3D Scatter Plots

Using mpl_toolkits.mplot3d

```
from mpl_toolkits.mplot3d import Axes3D

# Generate 3D data
x = np.random.rand(50)
y = np.random.rand(50)
z = np.random.rand(50)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c=z, cmap='plasma', s=100)

ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
ax.set_title("3D Scatter Plot")

plt.show()
```

Colormap (cmap=plasma)

- **Low z_{values} → Dark purple.**
- **High z_{values} → Dark orange.**

Quiz Section

Quiz

Everyone student should click on submit button before time ends otherwise MCQs will not be submitted

[Guidelines of MCQs]

1. There are 20 MCQs
2. Time duration will be 10 minutes
3. This link will be share on 6:10pm (Pakistan time)
4. MCQs will start from 6:15pm (Pakistan time)
5. This is exact time and this will not change
6. Everyone student should click on submit button otherwise MCQs will not be submitted after time will finish
7. Every student should submit Github profile and LinkedIn post link for every class. It include in your performance

Assignment

Assignment should be submit before the next class

[Assignments Requirements]

1. Create a post of today's lecture and post on LinkedIn.
2. Make sure to tag @Plus W @Pak-Japan Centre and instructors LinkedIn profile
3. Upload your code of assignment and lecture on GitHub and share your GitHub profile in respective your region group WhatsApp group
4. If you have any query regarding assignment, please share on your region WhatsApp group.
5. Students who already done assignment, please support other students

Q&A Session

ありがとうございます。
Thank you.
شکریا

For the World with Diverse Individualities