**Minor in AI**

**Batch 04 – Notes**

28 Feb 2025

Title: Mastering Data Visualization with Matplotlib: From Basics to Stunning Plots!

Syllabus:

* Understand the Basics of Matplotlib
* Import Matplotlib and set up a basic plotting environment.
* Create Basic Plots
* Generate line plots using plt.plot().

Matplotlib is a powerful Python library for data visualization, widely used for creating static, animated, and interactive plots. It provides a variety of chart types, including line graphs, bar charts, histograms, and scatter plots, making it ideal for exploratory data analysis. Matplotlib integrates well with libraries like NumPy and pandas, allowing seamless plotting of structured data. It offers extensive customization options, enabling users to modify colors, labels, and styles. The library supports multiple backends, making it versatile for different environments. With its ease of use and detailed control over graphical representations, Matplotlib is a fundamental tool for data scientists and analysts.

Note: Please refer to class video for more insights

Sample Program to generate a line plot:

Go to a weather site and collect the data of temperatures for a week or hour wise for a day and then plot the graph.

import matplotlib.pyplot as plt

# Sample weather data (replace with your actual data)

days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

temperatures = [25, 28, 25, 26, 27, 29, 30]

# Create the line plot

plt.plot(days, temperatures)

# Add labels and title

plt.xlabel("Day of the week")

plt.ylabel("Temperature (°C)")

plt.title("Weekly Temperature Trend")

# Display the plot

plt.show()

Read data from spread sheet and load to pandas and print

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_excel('sh-data.xlsx', sheet\_name='Sheet1')

plt.bar(df['Student Name'], df['Height'])

plt.xlabel('Student Name')

plt.ylabel('Height')

plt.title('Student Heights')

plt.show()

**Comparison Graph:**

import matplotlib.pyplot as plt

# Data: Days of the week

days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

# Data: Code written (in lines) & Bugs found

code\_written = [50, 120, 200, 180, 300, 50, 20]  # Lines of Code

bugs\_found = [5, 10, 20, 25, 40, 5, 1]  # Number of bugs

# Create the plot

plt.figure(figsize=(8, 5))

# Plot Code Written

plt.plot(days, code\_written, linestyle='-', color='b', label="Code Written (LOC)")

# Plot Bugs Found

plt.plot(days, bugs\_found, linestyle='--', color='r', label="Bugs Found")

# Add labels, title, and legend

plt.xlabel("Days of the Week")

plt.ylabel("Lines of Code / Bugs Found")

plt.title("Code Written vs. Bugs Found Over a Week")

plt.legend()  # Show legend

plt.grid(True, linestyle="--", alpha=0.6)

# Show the plot

plt.show()