LAB06: Data Visualization and Trees

Objectives

- Understand data visualization in AI.
- Use Matplotlib to create line graphs and bar charts.
- Use Treelib to create and display simple tree structures.
- Implement simple trees like binary trees.

1. Introduction to Data Visualization with Matplotlib

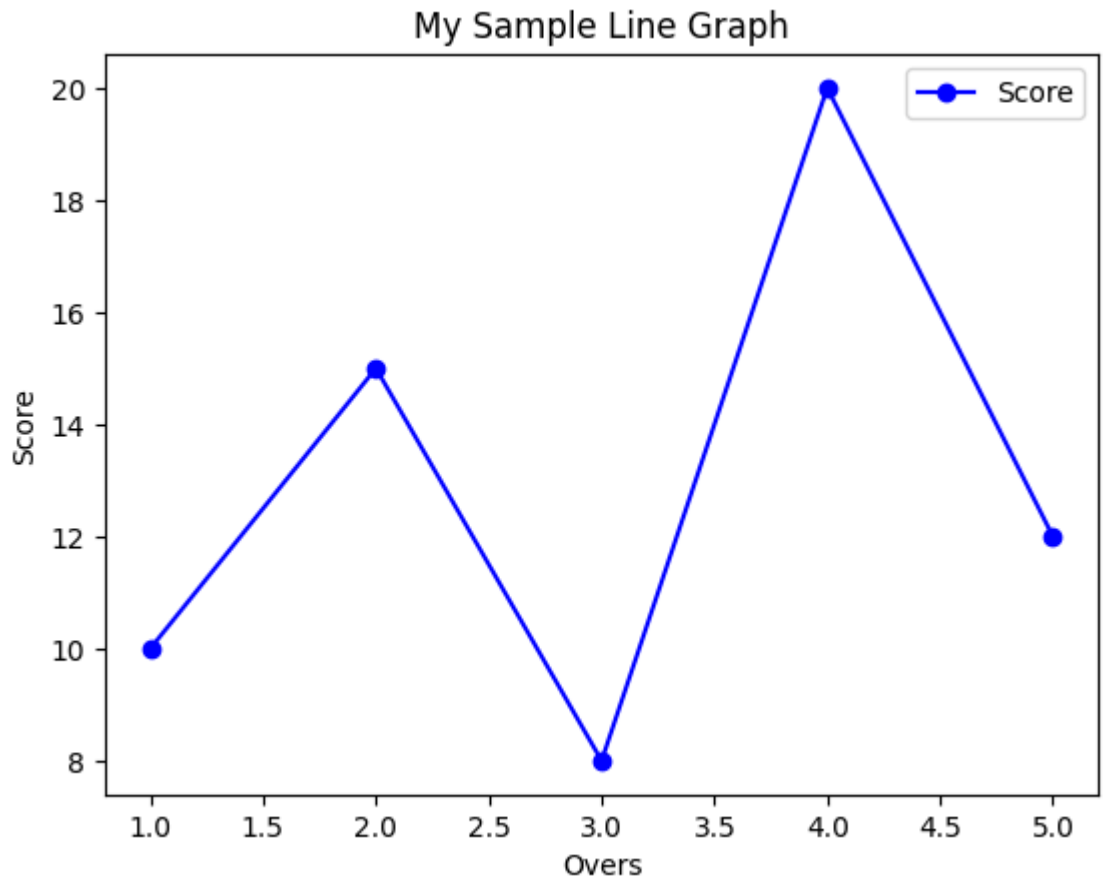
- Data visualization helps to understand patterns and relationships in data.
- Examples: Line graphs, bar charts, tree diagrams.
- Matplotlib is a Python library used to create static, interactive, and animated visualizations such as line graphs, bar charts, and scatter plots.

1.1 Hands-On 1: Line Graph

```
# Import the Matplotlib library for plotting
import matplotlib.pyplot as plt

# ------------------------------
# Step 1: Prepare the data
# ------------------------------

# X-axis values: Test numbers (1 to 5)
x = [1, 2, 3, 4, 5]
```

```
# Y-axis values: Scores obtained in each test
y = [10, 15, 8, 20, 12]

# ------------------------------
# Step 2: Create the line graph
# ------------------------------

# plt.plot() is used to create a line plot
# Parameters:
# x -> values on X-axis
# y -> values on Y-axis
# marker='o' -> show a circle at each data point
# color='blue' -> color of the line
# label='Score' -> label for the line to use in the legend
plt.plot(x, y, marker='o', color='blue', label='Score')

# ------------------------------
# Step 3: Add title and labels
# ------------------------------

# Title of the graph
plt.title('My Sample Line Graph')

# Label for X-axis
plt.xlabel('Overs')

# Label for Y-axis
plt.ylabel('Score')

# ------------------------------
# Step 4: Show legend
# ------------------------------

# Displays the legend on the graph using the label defined in plt.plot()
plt.legend()
```

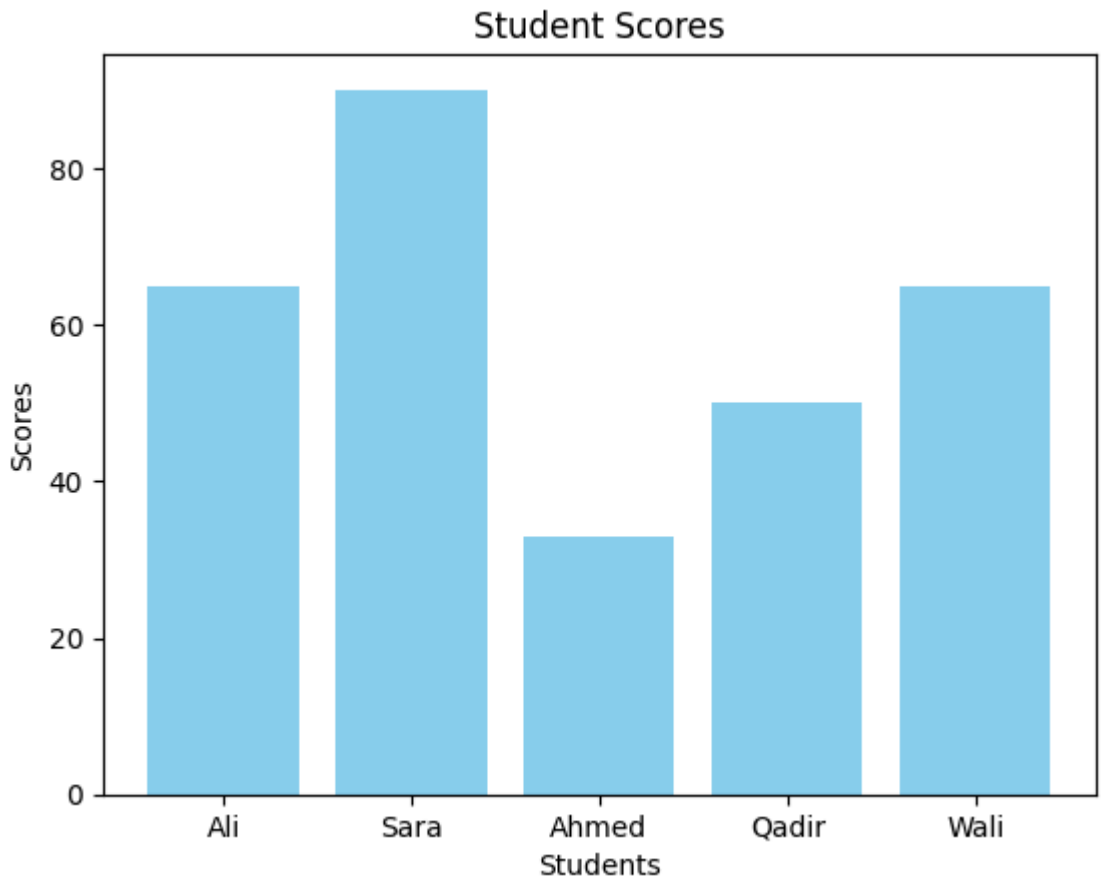
```
# ------------------------------
# Step 5: Display the graph
# ------------------------------

# plt.show() renders the graph on the screen
plt.show()
```



1.2 Hands-On 2: Bar Graph

```
# ------------------------------
# Step 1: Prepare the data
# ------------------------------

# List of student names for X-axis
students = ['Ali', 'Sara', 'Ahmed', 'Qadir', 'Wali']

# List of corresponding scores for Y-axis
scores = [65, 90, 33, 50, 65]

# ------------------------------
# Step 2: Create the bar chart
# ------------------------------

# plt.bar() is used to create a bar chart
# Parameters:
# students -> positions of the bars on X-axis
# scores -> height of each bar (Y-axis value)
# color='green' -> color of the bars
plt.bar(students, scores, color='skyblue')

# ------------------------------
# Step 3: Add title and axis labels
# ------------------------------

# Title of the bar chart
plt.title('Student Scores')

# Label for X-axis
plt.xlabel('Students')

# Label for Y-axis
plt.ylabel('Scores')

# ------------------------------
# Step 4: Display the chart
# ------------------------------
```

```
# plt.show() renders the chart on the screen
plt.show()
```

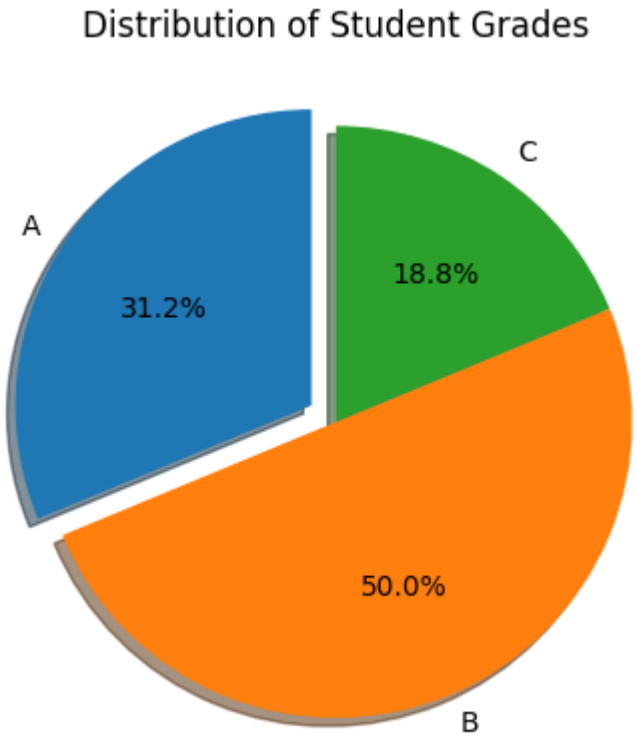


1.3 Hands-On 2: Pie Chart

A pie chart is a circular chart divided into slices to illustrate proportions or percentages.

Each slice represents a part of the whole dataset.

```
# Example data: Grades of students
grades = ['A', 'B', 'C']
counts = [5, 8, 3]  # Number of students in each grade

# Optional: Highlight the 'A' slice
explode = (0.1, 0, 0)  # only "explode" the 1st slice (A)

# Plot pie chart
plt.pie(counts, labels=grades, explode=explode, autopct='%1.1f%%', shadow=True, startangle=90)

# Add a title
plt.title("Distribution of Student Grades")

# Display the chart
plt.show()
```




```
Start coding or generate with AI.
```

2. Introduction to Trees with Treelib

- Tree: hierarchical structure with root, children, and leaf nodes.
- Used in AI for decision trees, game trees, family trees.

2.1 Hands-On 3: Simple Binary Tree

```
!pip install treelib
```

```
Requirement already satisfied: treelib in /usr/local/lib/python3.12/dist-packages (1.8.0)
Requirement already satisfied: six>=1.13.0 in /usr/local/lib/python3.12/dist-packages (from treelib) (1.17.0)
```

```
from treelib import Node, Tree

# Create tree
tree = Tree()
tree.create_node('Root', 'root')
tree.create_node('Left', 'left', parent='root')
tree.create_node('Right', 'right', parent='root')
tree.create_node('Left.Left', 'left.left', parent='left')
tree.create_node('Right.Right', 'right.right', parent='right')

# Display tree
tree.show()
```

```
Root
├── Left
│   └── Left.Left
└── Right
    └── Right.Right
```

2.2 Hands-On 4: Adding More Levels

```
tree.create_node('Left.Left.Left', 'left.left.left', parent='left.left')
tree.show()
```

```
Root
├── Left
│   └── Left.Left
│       └── Left.Left.Left
└── Right
    └── Right.Right
```

Lab Exercises

1. Matplotlib Exercises

1.1 Line Graph Exercises

- Plot a **line graph** of monthly temperatures for 6 months.
- Add **markers** for each point and label the axes.
- Change the **line color** and **style** (e.g., dashed line).

1.2 Bar Chart Exercises

- Plot a **bar chart** of favorite fruits and their counts (e.g., apple: 10, banana: 7, orange: 5).
- Change the **color of bars**.

1.3 Pie Chart

- Visualize the **percentage of students in different grade categories** (A, B, C).
- Add **labels and percentages** to the pie chart.

---

2. Treelib Exercises

2.1 Binary Tree Exercise

- Create a **binary tree** with 3 levels.
- Print the tree using `.show()`.
- Add **one more level** and display it again.

2.2 Family Tree Exercise

- Create a **family tree** with at least 6 members.
- Assign parents and children properly.
- Display the tree and **highlight the youngest member**.

2.3 School Hierarchy Tree

- Create a tree representing a **school hierarchy**: Principal ⟶ Teachers ⟶ Students.
- Add **two teachers**, each with **3 students**.