- Module 3_2 - Data Visualization using Python

Outline

- Data Visualization
  - Matplotlib Package
- Plot Types
  - Line Plot
  - Bar Chart
  - Scatter Plot
  - Histogram
- Figure Parts

Data Visualization (1)

- One of the easiest ways to discern important relationships in data is through easy-to-understand **visualizations**

- **Data visualization** is an important skill in applied statistics and machine learning
  - uses tools to gain a **qualitative understanding** of the data
  - helps with exploring a dataset to identify patterns and anomalies in the data
  - determine which machine learning model to use
  - evaluate the performance of a trained machine learning model

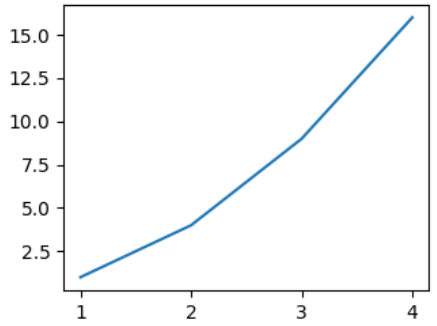
Data Visualization: Matplotlib Package (2)

- Matplotlib is a plotting package for **Python**
  - the matplotlib.pyplot module provides a simple interface for generating various plots quickly
  - allows for data visualization using Python
  - To use this module

    ```
    import matplotlib.pyplot as plt
    ```

  - To plot a line

    ```
    plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
    plt.show()
    ```



- Many plot types exist
  - **line, bar, scatter** and **histogram** plots will primarily be used in this course

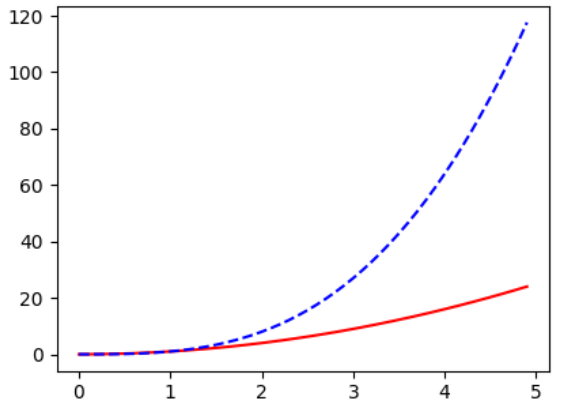
Source: https://matplotlib.org/

Plot Types: Line (1)

- **Line Plot**
  - shows the relationship of one variable to another
  - used when the values on the X-axis are of a continuous quantity
  - stacking lines are used to compare trends for several variables

```
import matplotlib.pyplot as plt
import numpy as np
# generate an array from 0.0 to 4.9 with a step
# size of 0.1
x = np.arange(0.0, 5.0, 0.1)
# create a line plot (red solid line)
plt.plot(x, x**2, 'r-')
# add another line plot (blue dashed line)
plt.plot(x, x**3, 'b--')
# show line plots
plt.show()
```

Output:

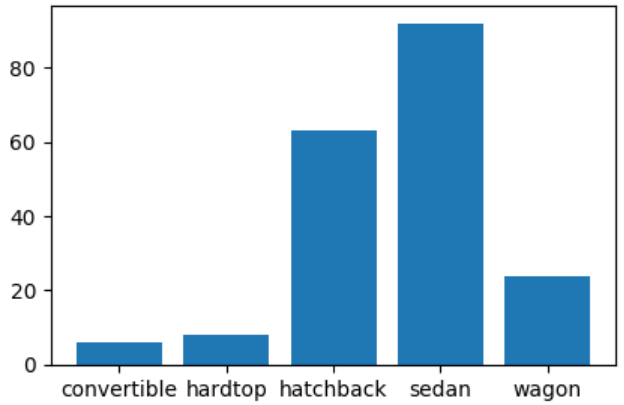


Plot Types: Bar (2)

- **Bar Chart**
  - shows categorical data using rectangle bars with heights (vertical bars) or lengths (horizontal bars) proportional to the values they represent
  - used to show comparisons among discrete categories
    - ➔ one axis shows the specific categories being compared, and the other axis shows the measured values

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
data = pd.read_csv('AutomobilePrice.csv')
# count the frequency of unique values in the
# 'body-style' column
body_style, counts = np.unique(data.loc[:, 'body-style'], return_counts=True)
# create a bar chart
plt.bar(body_style, counts)
# show bar chart
plt.show()
```

Output:

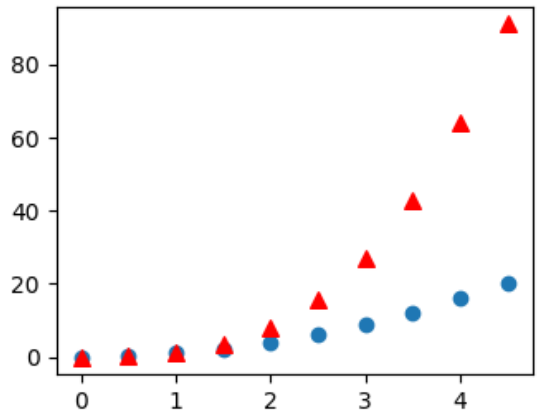


Plot Types: Scatter (3)

- **Scatter Plot**
  - shows the joint variation of two variables
  - each marker (e.g., symbols such as circles, squares or triangles) represents an observation
  - used to visualize how spread out the data might be or how closely related the data points are
  - identify patterns present in the distribution of the data

```
import matplotlib.pyplot as plt
import numpy as np
# generate an array from 0.0 to 4.5 with a step
# size of 0.5
x = np.arange(0.0, 5.0, 0.5)
# create a scatter plot (default marker: blue circle)
plt.scatter(x, x**2)
# add another scatter plot
plt.scatter(x, x**3, s=48, c='r', marker='^')
# show scatter plots
plt.show()
```

Output:



Plot Types: Histogram (4)

- **Histogram**
  - shows the distribution of data
  - the **bins** of a histogram determines how the entire range of values are divided into a series of intervals
    - ➔ number of values that fall into each interval is then counted

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
autoData = pd.read_csv('AutomobilePriceLab3.csv')
# create a historgram of the automobile price
x = autoData.loc[:, 'price']
plt.hist(x, bins=20)
# show historgram
plt.show()
```

Output:

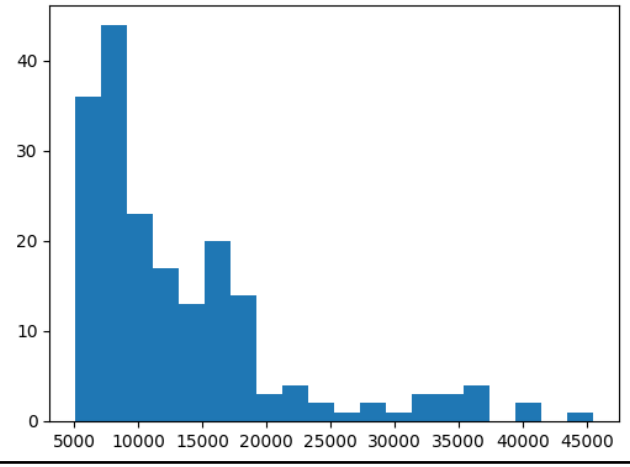
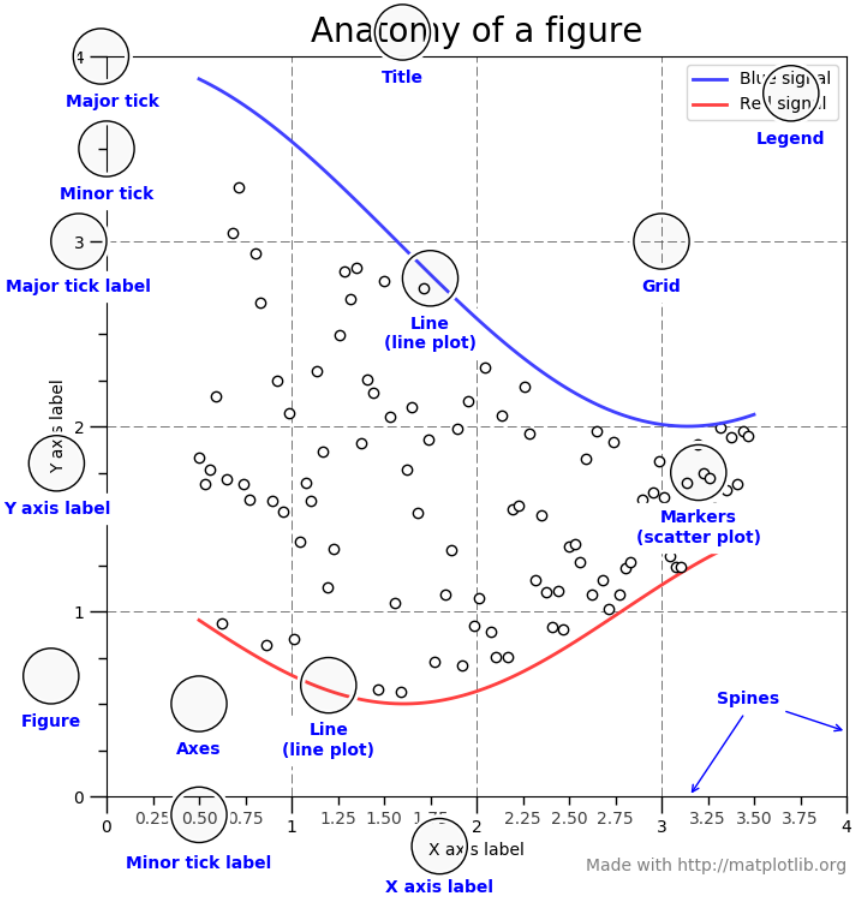
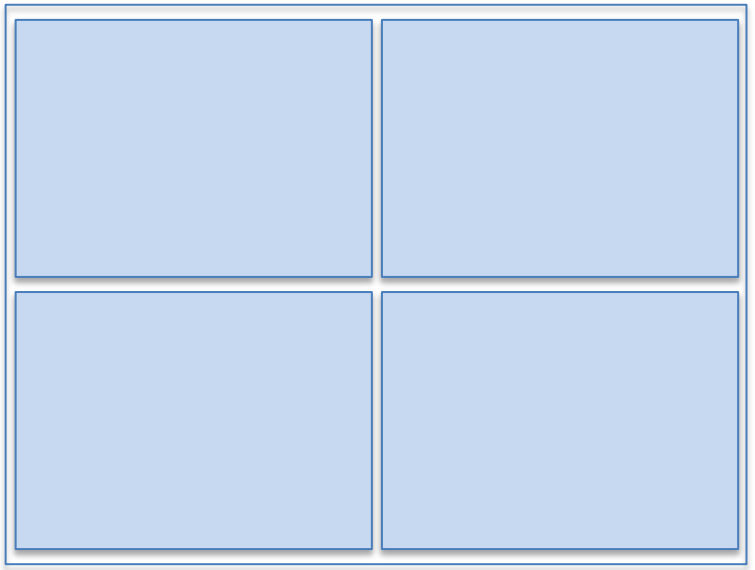


Figure Parts (1)



A figure with 2x2 grid of Axes (SubPlots)



Source: https://matplotlib.org/tutorials/introductory/usage.html

Figure Parts (2)

- For each plot, include
  - **Title** — plt.title('Anatomy of a figure')
  - **X-label** — plt.xlabel('X axis label')
  - **Y-label** — plt.ylabel('Y axis label')
  - **Legend** — plt.plot(x, x**2, label='x square')
plt.plot(x, x**3, label='x cube')
plt.legend()
  - Grid — plt.grid(True)
  - Limit of X-axis — plt.xlim([xmin, xmax])
  - Limit of Y-axis — plt.ylim([ymin, ymax])
  - …