Python (Data Visualization Tools)

Matplotlib

* Maptlotlib is a python library used for creating static, animated, and interactive visualizations.
* It is highly customizable and provides low-level control over plots.
* Wors well with NumPy and Pandas for data visualization.

A Matplotlib figure consists of the following elements:

* + Figure: The overall container for all elements.
  + Axes: The area where data is plotted (can have multiple Axes in a Figure).
  + Axis (X & Y): Defines the scale and limits.
  + Ticks: Marks on the axes.
  + Legend: Describes the elements of a plot.
  + Grid: Background reference lines.

There are different plots in Matplotlib – Line Plot, Scatter Plot, Bar Chart, Histogram, Pie Chart, and a Box Plot.

Matplitlib is better for fine-tuned customization.

Seaborn

* Seaborn is a Python data visualization library built on top of Matplotlib.
* It provides statistical plotting functions and supports Pandas DataFrames directly.
* Seaborn is designed for beautiful, informative visualizations with minimal code.

Differnces between Seaborn and Matplotlib:

* Seaborn is Simple and Concise : Matplotlib requires more customization
* Seaborn works with PandasDataFrames : Matplotlib works with NumPy/Pandas
* Seaborn has Predefined themes : Matplotlib requires manual styling
* Seaborn has Built-in Statistical Functions : Matplotlib requires manual implementation

Seaborn provides themes for aesthetic styling.

Seaborn provides different plots for statistical analysis:

* Line Plot
* Scatter Plot
* Bar Plot
* Histogram
* KDE Plot
* Box Plot
* Violin Plot
* Pair Plot
* Heatmap

Seaborn is used for quick statistical insights and beautiful plots.

Plotly

* Plotly is a python library for interactive visualizations.
* Supports zooming, panning, tooltips, and hover effects.
* Works well for real-time dashboards and data analytics applications.
* Compatible with Jupyter Notebooks, Dash and Web Apps.

Key Features of Plotly:

* Interactive Charts
* Works with Pandas
* Web & Dash Integration
* Supports Multiple Chart Types
* Real-Time Updates

Plotly is best for Interactive & web-based charts.

Common Example:

* Stock market movements are uncertain, and investors need probabilistic models to analyze risk and predict trends. Different probability distributions can help model returns, price jumps, and investor behavior, while vector representation are useful for portfolio analysis.
  + How do daily stock returns behave? (Normal Distribution)
  + What is the probability of large stock price movements? (Bernoulli Distribution)
  + How do investors make buy/sell decisions? (Bernoulli Distribution)
  + How can we represent a portfolio mathematically? (Vector Representation)
* Understanding Stock Return Behavior (Normal Distribution)
  + Stock returns tend to follow a normal distribution.
  + Computed the mean and standard deviation of daily returns.
  + Used Matplotlib to compare real and return distribution vs. theoretical normal distribution.
  + If stock returns are normally distributed, mean and sd can be used to estimate future risk.
* Modeling Stock Price Jumps (Binomial Distribution)
  + Large stock price jumps don’t happen everyday.
  + Binomial distribution models the probability of price jumps occurring (1 = jump, 0 = no jump).
  + Used Seaborn to visualize the number of days with/without jumps.
  + This helps in identifying high-risk trading periods.
* Simulating Investor Decisions (Bernoulli Distribution)
  + Investors make binary decisions (Buy = 1, Sell = 0)
  + Bernoulli Distribution models this decision-making process.
  + Used plotly to create an interactive histogram of buy/sell events.
  + This can be extended to build AI-driven trading strategies.
* Representing a Portfolio with Vectors
  + A portfolio contains multiple stocks with different wight allocations.
  + Used Numpy arrays to represent stock weights and computed expected portfolio returns using dot product.
  + Used Matplotlib to visualize portfolio allocation.
  + Vectors simplify investment strategies and portfolio management.

Graph Interpretations:

* Normal Distribution:
  + Histogram – Shows the real distribution of daily stock returns.
  + Red Curve – Theoretical normal distribution fit to compare against real data.
  + Green Line. – The average daily return.
  + Orange Lines – Boundaries where 68% of the data should lie if normally distributed.
  + Mean specifies if the stock has an expected positive or a negative return.
  + Standard Deviation measures the volatility i.e., how much the stock returns fluctuate. Higher the SD higher volatility (Riskier Stock – High risk, High reward)
  + Skewness – If the histogram follows the red normal distribution curve, then stock returns are normally distributed.
  + Right Skewed – Extreme Positive
  + Left Skewed – Extreme Negative
* Binomial Distribution:
  + X axis – Represents whether a stock price jump occurred (1) or not (0).
  + Y axis – Represents the number if occurrences in a given period.
  + Bins – Only two values (0 = No jump, 1 = Jump)
  + It models the probability of a stock experiencing a large price jump (up or down) on a given day.
  + Each trading day is a Bernoulli trial (Yes/No outcome : Did the price jump?).
  + The binomial distribution then models the total number of jumps over a period.
* Bernoulli Distribution:
  + X axis – Represents the decision (Buy = 1, Sell = 0)
  + Y axis – Represents the number of occurrences (how often each decision happened).
  + The Bernoulli distribution is used for binary (yes/no) decisions.
  + In this case, it models investor behavior:
    - 1(Buy) – Investor chooses to buy the stock.
    - 0(Sell) – Investor chooses to sell the stock.
  + Assumed a 60% chance that an investor buys the stock on any given day.
  + Observed over a period of 252 days.
* Vectors:
  + X axis – Represents the different stocks In the portfolio.
  + Y axis – Represents weight allocation (%) of each stock.
  + Bars – Each stock has a different height based on its portfolio percentage.
  + A portfolio is a collection of multiple stocks with different weight allocation.
  + Vector representation makes it easy to compute expected portfolio return and risk.
  + Expected return of a portfolio is = w1r1 + w2r2 + w3r3
  + W = weight of the stock, R = Expected return of the stock.