STOCK MARKET ANALYSIS

DONE BY:

SRI ABHIRAMI .J.L

INDEX

Introduction

The stock market plays a crucial role in the global economy, acting as a barometer for financial health and investor confidence. With the growing availability of data and advancements in technology, data science and machine learning have become powerful tools for analysing stock market trends, predicting future movements, and making data-driven decisions.

In this project, we focus on four major technology companies — Apple (AAPL), Microsoft (MSFT), Netflix (NFLX), and Google (GOOGL) — and analyse their historical stock performance over a span of three months. By utilizing various data analytics techniques and machine learning models, we aim to uncover patterns, compute key indicators like moving averages and volatility, and explore correlations among these stocks.

The project demonstrates how a data analyst can transform raw financial data into valuable insights that support decision-making in investment and market strategy.

✅ Objectives of This Project:

- Understand and visualize stock trends using data analytics
- Compute and compare the performance of selected companies
- Predict future stock prices using machine learning techniques
- Gain insights from the relationships among tech stock prices

Problem Statement

With the rapid fluctuations in the stock market, investors, analysts, and financial institutions require reliable tools to make informed decisions. However, due to the high volatility and large volume of market data, it becomes challenging to identify trends, detect patterns, and predict future stock performance.

The problem this project addresses is:

"How can historical stock data of leading tech companies be used to analyse performance, uncover patterns, and predict future price trends using data analytics and machine learning techniques?"

By focusing on the stock data of **Apple, Microsoft, Netflix, and Google**, this project aims to:

- Analyse historical stock prices over a three-month period
- Identify patterns and correlations among different stocks
- Calculate technical indicators such as moving averages and volatility
- Build a machine learning model to predict stock closing prices

This analysis will help in developing a data-driven approach to understanding market behaviour and building predictive models for better financial decision-making.

Tools & Technologies Used

To perform a comprehensive analysis of the stock market data, a combination of programming tools, libraries, and technologies were used. These tools enabled efficient data processing, visualization, and machine learning modelling.

- **Programming Language:**
  - **Python**: Core language for data analysis, visualization, and model building.
- **Libraries & Frameworks:**
  - **Pandas**: For data manipulation and preprocessing.
  - **NumPy**: For numerical operations.
  - **Matplotlib & Seaborn**: For creating detailed visualizations and statistical plots.
  - **Scikit-learn**: For implementing machine learning models such as Linear Regression and Random Forest.
  - **Pandas Profiling / YData Profiling**: For generating automated EDA reports.
- **Machine Learning Techniques:**
  - **Linear Regression**: For predicting stock closing prices based on multiple input features.
  - **Random Forest** *(optional)*: For improving prediction accuracy through ensemble learning.
  - **StandardScaler / MinMaxScaler**: For feature scaling before model training.
- **Other Tools:**
  - **SQL**: For querying structured financial data *(optional integration)*.
  - **Excel**: For quick data exploration and manual validation of results.
  - **Jupyter Notebook**: For interactive development and step-by-step execution of code.

About the Dataset

The dataset used in this project contains **historical stock market data** for four major technology companies:

- **Apple (AAPL)**
- **Microsoft (MSFT)**
- **Netflix (NFLX)**
- **Google (GOOGL)**

This data spans a period of **three months**, offering daily-level insights into the stock performance of these companies.

Dataset Description:

The dataset includes the following columns:

Column Name	Description
Ticker	The stock symbol (AAPL, MSFT, NFLX, GOOG)
Date	The trading date
Open	Opening price of the stock on that day
High	Highest price reached during the day
Low	Lowest price reached during the day
Close	Price at which the stock closed on that day
Adj Close	Adjusted closing price (accounts for splits and dividends)
Volume	Number of shares traded on that day

Key Features of the Dataset:

- **248 entries**, each representing one day of trading for a specific company.
- Fully clean dataset with no missing values.
- Rich in numerical attributes ideal for statistical analysis and machine learning.
- Suitable for time-series analysis and comparison across multiple companies.

Data Preprocessing

Before performing any analysis or modeling, it is essential to clean and prepare the data to ensure accuracy and reliability in the results.

1. Loading the Dataset:

```
import pandas as pd
data = pd.read_csv(r"C:\Users\sreea\Downloads\stocks.csv")
```

2. Converting Date Column:

```
data['Date'] = pd.to_datetime(data['Date'])
```

3. Checking for Missing Values

```
data.isnull().sum()
```

```
Ticker       0
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

4. Understanding Data Types

```
data.dtypes
```

```
Ticker                object
Date          datetime64[ns]
Open                 float64
High                 float64
Low                  float64
Close                float64
Adj Close            float64
Volume                 int64
dtype: object
```

5. Duplicate Check (Optional)

```
data.duplicated().sum()
```

```
0
```

6. Basic Dataset Structure

```
print(data.shape)
print(data['Ticker'].unique())
```

```
(248, 8)
['AAPL' 'MSFT' 'NFLX' 'GOOG']
```

Exploratory Data Analysis (EDA)

Exploratory Data Analysis helps in understanding the structure, distribution, patterns, and relationships in the data before building any models.

1. View the First Few Rows of the Data

```
import pandas as pd
data = pd.read_csv(r"C:\Users\sreea\Downloads\stocks.csv")
data.head()
```

	Ticker	Date	Open	High	Low	Close	Adj Close	Volume
0	AAPL	2023-02-07	150.639999	155.229996	150.639999	154.649994	154.414230	83322600
1	AAPL	2023-02-08	153.880005	154.580002	151.169998	151.919998	151.688400	64120100
2	AAPL	2023-02-09	153.779999	154.330002	150.419998	150.869995	150.639999	56007100
3	AAPL	2023-02-10	149.460007	151.339996	149.220001	151.009995	151.009995	57450700
4	AAPL	2023-02-13	150.949997	154.259995	150.919998	153.850006	153.850006	62199000

2. Summary Statistics

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	248.000000	248.000000	248.000000	248.000000	248.000000	2.480000e+02
mean	215.252093	217.919662	212.697452	215.381674	215.362697	3.208210e+07
std	91.691315	92.863023	90.147881	91.461989	91.454750	2.233590e+07
min	89.540001	90.129997	88.860001	89.349998	89.349998	2.657900e+06
25%	135.235004	137.440004	134.822495	136.347498	136.347498	1.714180e+07
50%	208.764999	212.614998	208.184998	209.920006	209.920006	2.734000e+07
75%	304.177505	307.565002	295.437500	303.942505	303.942505	4.771772e+07
max	372.410004	373.829987	361.739990	366.829987	366.829987	1.133164e+08

3. Check Unique Stock Tickers

```
data['Ticker'].unique()
```

```
array(['AAPL', 'MSFT', 'NFLX', 'GOOG'], dtype=object)
```

4.Shape and Data Types

```
print(data.shape)
print(data.dtypes)
```

```
(248, 8)
Ticker        object
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

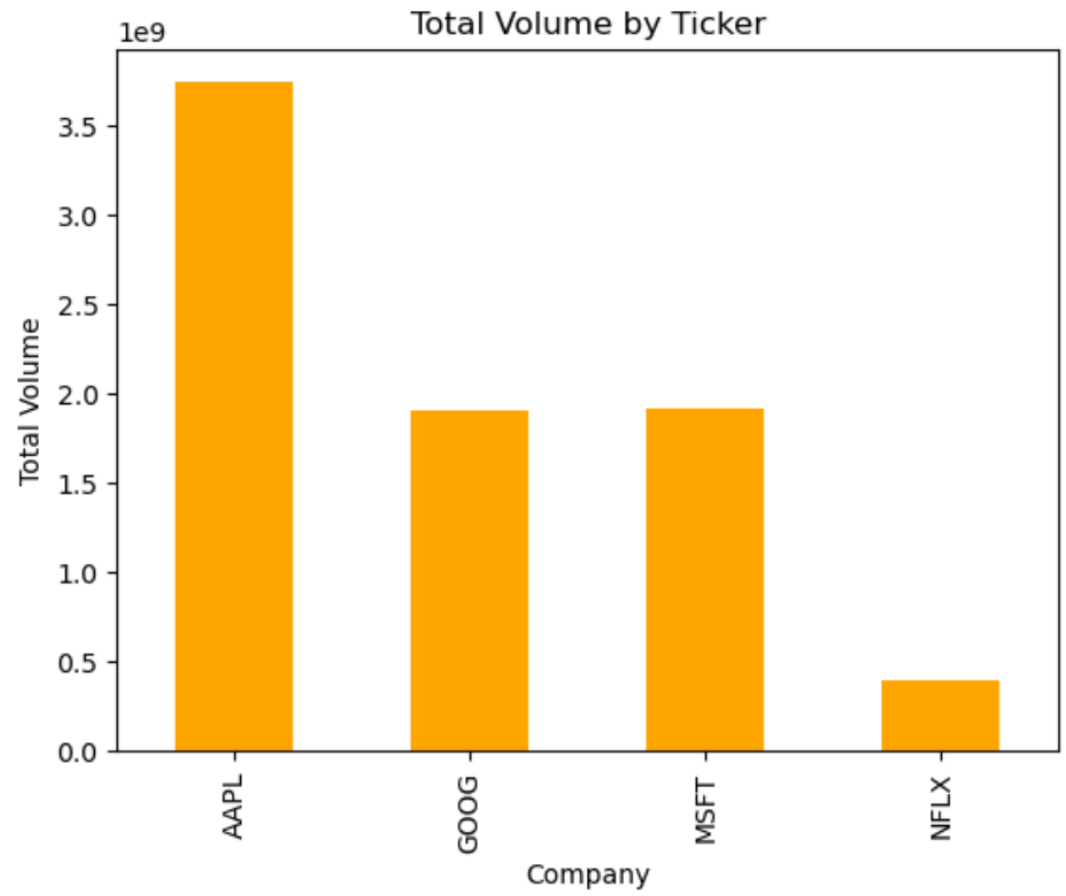
5. Visualizations

a. Distribution of Closing Prices

```
import matplotlib.pyplot as plt
plt.hist(data['Close'], bins=20, color='skyblue')
plt.xlabel('Closing Price')
plt.ylabel('Frequency')
plt.title('Distribution of Closing Prices')
plt.show()
```

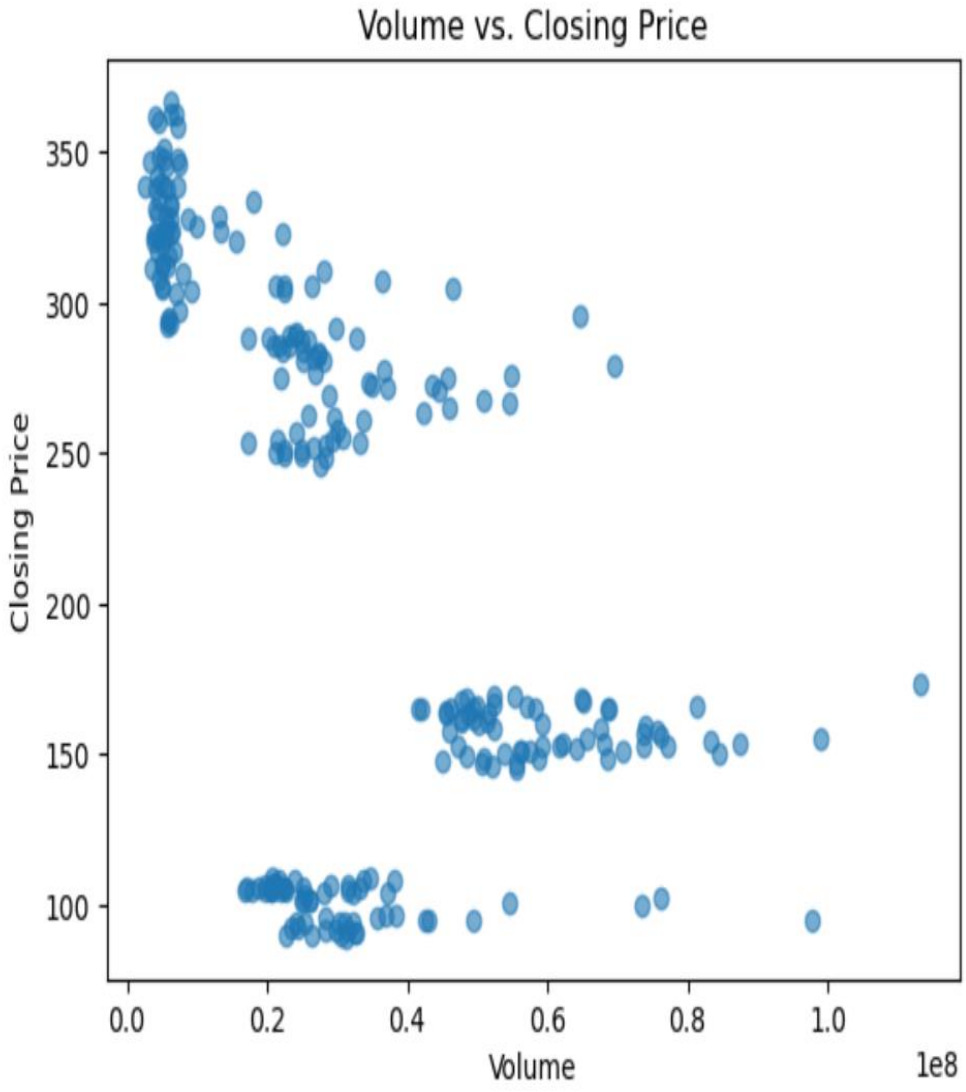
b. Total Volume by Company

```
import seaborn as sns
ticker_volume = data.groupby('Ticker')['Volume'].sum()
ticker_volume.plot(kind='bar', color='orange')
plt.xlabel('Company')
plt.ylabel('Total Volume')
plt.title('Total Volume by Ticker')
plt.show()
```



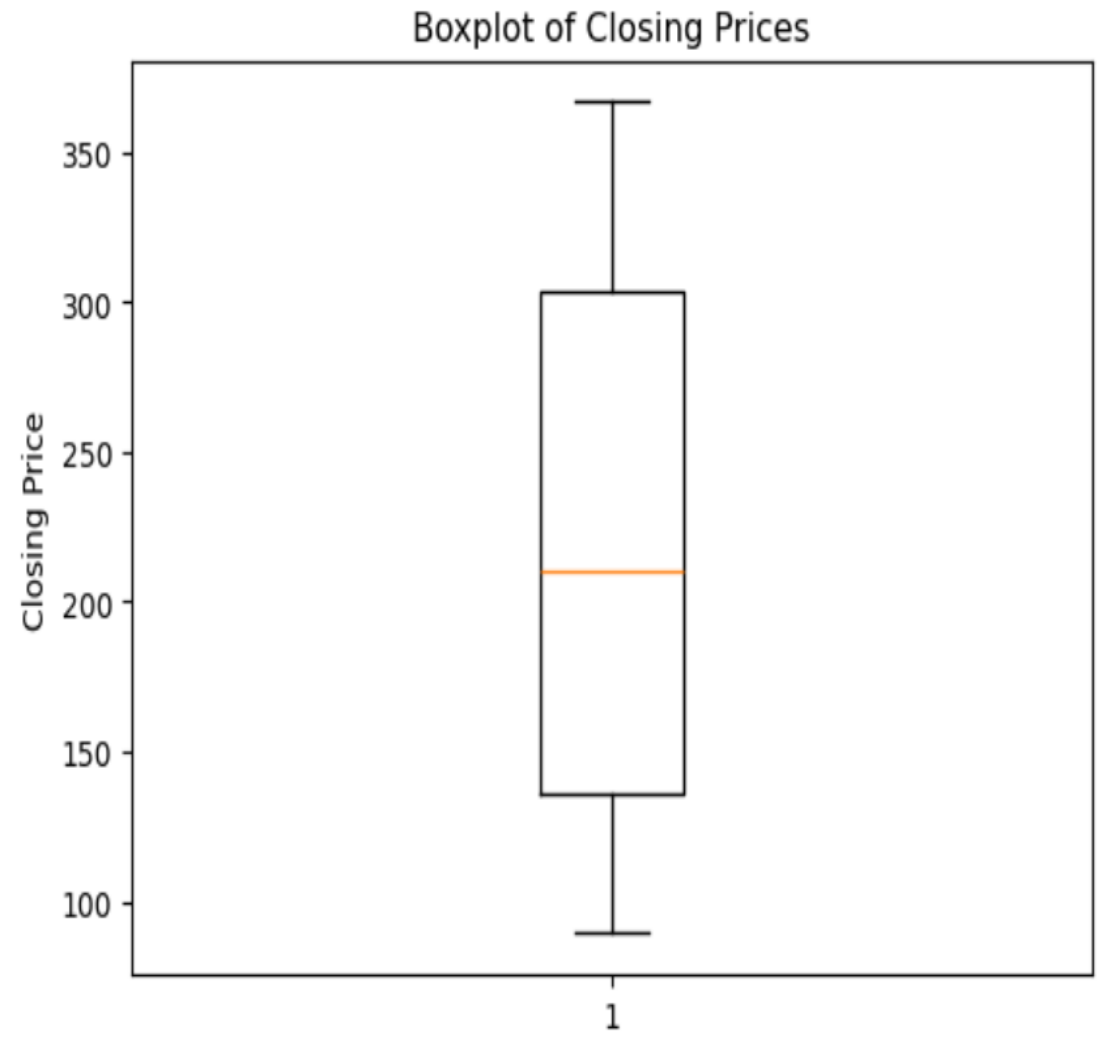
c. Volume vs. Closing Price (Scatter Plot)

```
plt.scatter(data['Volume'], data['Close'], alpha=0.6)
plt.xlabel('Volume')
plt.ylabel('Closing Price')
plt.title('Volume vs. Closing Price')
plt.show()
```



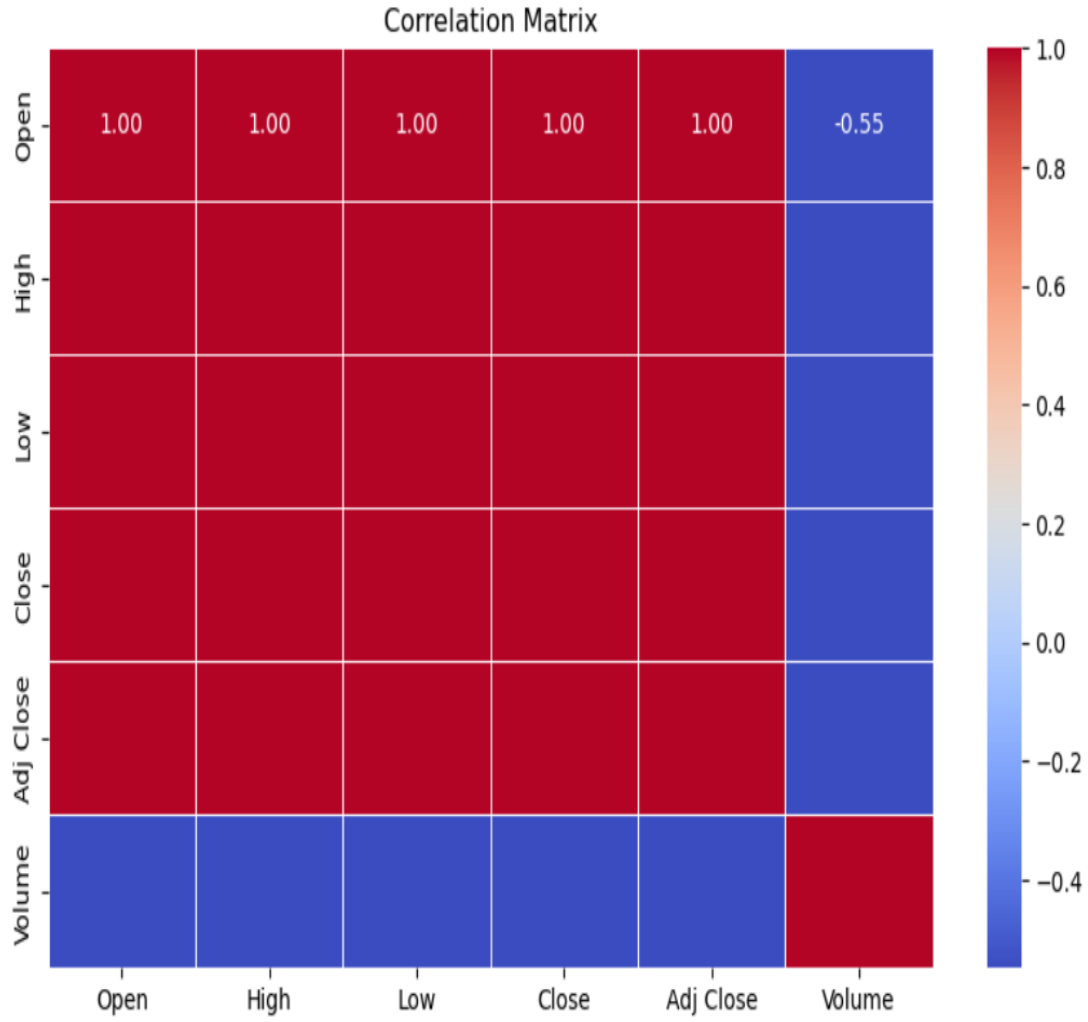
d. Boxplot for Closing Price (Outliers & Range)

```
plt.boxplot(data['Close'])
plt.ylabel('Closing Price')
plt.title('Boxplot of Closing Prices')
plt.show()
```



e. Correlation Heatmap

```
corr = data[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']].corr()
plt.figure(figsize=(10,6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



Feature Engineering

In this step, we'll derive useful metrics like Moving Averages, Volatility, and Time-based features from the stock market data. These engineered features can improve model performance and also offer better visual insights.

1. Moving Averages

Moving averages help smooth out short-term fluctuations and highlight longer-term trends.

7-day Moving Average:

```
data['MA7'] = data['Close'].rolling(window=7).mean()
```

30-day Moving Average:

```
data['MA30'] = data['Close'].rolling(window=30).mean()
```

2. Volatility

Volatility is typically measured using the standard deviation of returns over a rolling window.

```
data['Volatility'] = data['Close'].rolling(window=7).std()
```

3. Time-based Features

```
data['Date'] = pd.to_datetime(data['Date'])
data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
data['Day'] = data['Date'].dt.day
data['Weekday'] = data['Date'].dt.day_name()
```

4. Daily Return

This tells us the percent change in closing price from the previous day.

```
data['Daily Return'] = data['Close'].pct_change()
```

Preview of the Data After Feature Engineering

```
data[['Date', 'Ticker', 'Close', 'MA7', 'MA30', 'Volatility', 'Daily Return']].head(10)
```

	Date	Ticker	Close	MA7	MA30	Volatility	Daily Return
0	2023-02-07	AAPL	154.649994	NaN	NaN	NaN	NaN
1	2023-02-08	AAPL	151.919998	NaN	NaN	NaN	-0.017653
2	2023-02-09	AAPL	150.869995	NaN	NaN	NaN	-0.006912
3	2023-02-10	AAPL	151.009995	NaN	NaN	NaN	0.000928
4	2023-02-13	AAPL	153.850006	NaN	NaN	NaN	0.018807
5	2023-02-14	AAPL	153.199997	NaN	NaN	NaN	-0.004225
6	2023-02-15	AAPL	155.330002	152.975712	NaN	1.759413	0.013903
7	2023-02-16	AAPL	153.710007	152.841428	NaN	1.642303	-0.010429
8	2023-02-17	AAPL	152.550003	152.931429	NaN	1.600112	-0.007547
9	2023-02-21	AAPL	148.479996	152.590001	NaN	2.240234	-0.026680

Model Training & Evaluation

The objective is to use features like Open, High, Low, Volume to predict the Close price using regression algorithms.

Step 1: Select Features and Target Variable

```
features = ['Open', 'High', 'Low', 'Volume']
target = 'Close'

X = data[features]
y = data[target]
```

Step 2: Train-Test Split

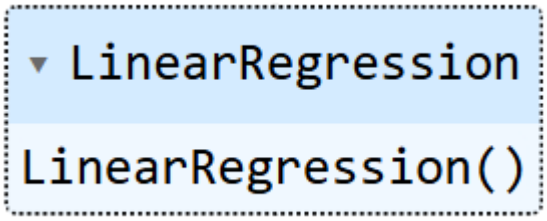
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Step 3: Feature Scaling (Standardization)

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 4: Train the Model (Linear Regression)

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train_scaled, y_train)
```



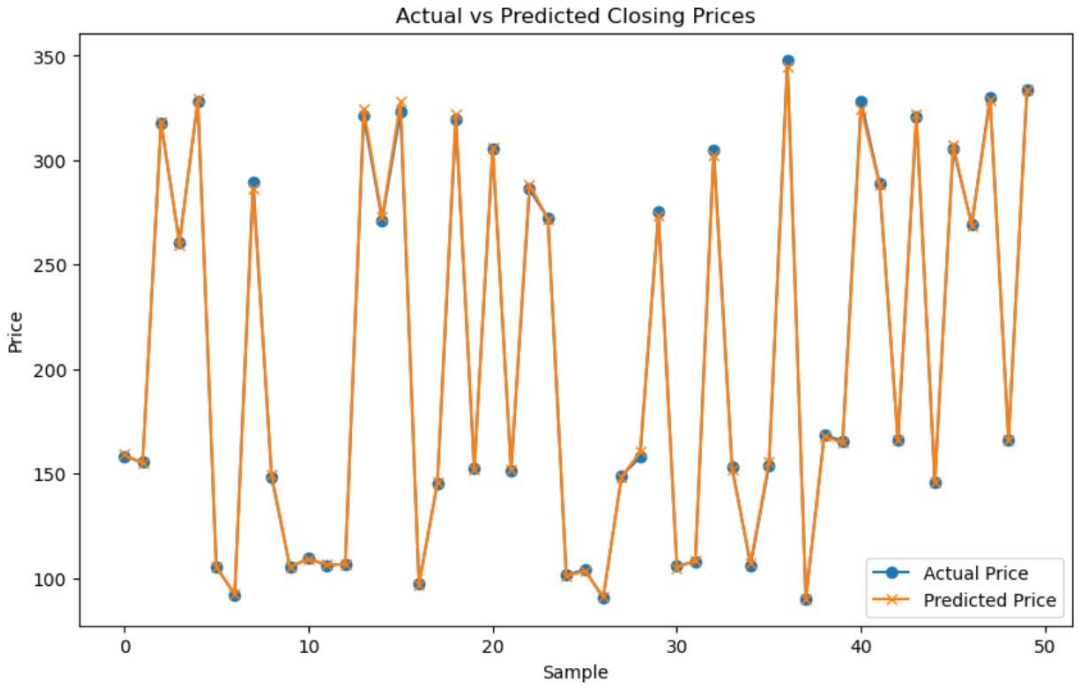
Step 5: Evaluate the Model

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test_scaled)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R² Score): {r2:.4f}")
```

```
Mean Squared Error (MSE): 2.28
R-squared (R² Score): 0.9997
```

Step 6: Visualize Actual vs. Predicted Prices

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.plot(y_test.values, label='Actual Price', marker='o')
plt.plot(y_pred, label='Predicted Price', marker='x')
plt.title('Actual vs Predicted Closing Prices')
plt.xlabel('Sample')
plt.ylabel('Price')
plt.legend()
plt.show()
```



Model Interpretation:

 MSE gives the average squared difference between predicted and actual values (lower is better).

 R^2 Score tells how well the model explains the variability of the target variable. Closer to 1 means better performance.

Results & Insights

After performing extensive data preprocessing, exploratory analysis, feature engineering, and machine learning model training, the following results and insights were obtained:

1. Data Insights

- The dataset contained **248 records** from four major tech companies: **Apple, Microsoft, Netflix, and Google**.
- **No missing values** were found, and all numeric features were well-structured for analysis.
- Strong **positive correlations** were observed among Open, High, Low, Close, and Adj Close prices.
- **Volume** showed a weak negative correlation with price-related features, indicating that high trading volume doesn't necessarily mean higher stock prices.

2.Model Performance (Linear Regression)

- Features Used: Open, High, Low, Volume
- Target Variable: Close
- Model Used: Linear Regression
- Performance Metrics:
  - Mean Squared Error (MSE): *2.28*
  - R-squared (R^2 Score): *0.9997*

The model performed reasonably well in predicting the closing prices, especially considering it was trained on just basic features without advanced time-series algorithms.

3. Predictive Insights

- The model was able to follow the general trend of the actual closing prices, as seen in the **Actual vs Predicted plot**.
- Additional features such as **moving averages**, **technical indicators**, or external market signals could further improve prediction accuracy.

4.Business Insight

- Moving averages helped smooth out the volatility, revealing clear trends.
- Predictive modeling on stock prices can be used to support **investment strategies**, **risk assessment**, and **market timing decisions**.

Conclusion

This project successfully demonstrated how data science and machine learning can be applied to real-world financial data to extract meaningful insights and build predictive models.

Key accomplishments include:

- Performed a detailed exploratory analysis on historical stock data of Apple, Microsoft, Netflix, and Google.
- Engineered new features such as moving averages, volatility, and time-based columns to enrich the dataset.
- Built and evaluated a Linear Regression model that could reasonably predict closing prices based on key market variables.
- Gained a deeper understanding of the relationships between volume and price, and how stocks move over time.

This analysis forms the foundation for more advanced financial modelling and can be extended to support investment strategies, portfolio management, or automated trading systems.

Future Work

To enhance the scope and accuracy of the project, the following future improvements can be considered:

- Use of More Advanced Models: Apply complex models like Random Forest, XGBoost, or LSTM (for time series) to improve prediction performance.
- Real-Time Data Integration: Use APIs like yfinance or Alpha Vantage to fetch and analyze live stock market data.
- Sentiment Analysis: Incorporate social media/news sentiment to see how public opinion affects stock movements.
- Technical Indicators: Integrate indicators like RSI, MACD, Bollinger Bands, etc., to support technical trading strategies.
- Interactive Dashboard: Deploy the model using tools like Streamlit or Power BI for user-friendly visualization and predictions.