Stock Market Analysis

Project Overview

The project is to analyse market trends and predict future market behaviour with the help of historical stock prices using ML techniques for companies like Apple, Microsoft, Netflix and Google.

Tools and Technologies Used

- Language – Python
- Libraries – pandas, numpy, matplotlib, seaborn, scikit-learn, scipy, keras, tensorflow, scikeras, ydata_profiling
- Oher Tools used – Jupyter Notebook

Dataset Overview

- Rows – 248 and Columns – 8
- Companies – Apple, Microsoft, Netflix and Google
- The data was collected in a .csv file with daily Open, High, Low, Close prices, adjusted close, and trading volume of different stocks.

Data Preparation

- We did the following –
    1. Converted Date to datetime
    2. Sorted by Date
    3. Checked duplicates and null values – none were present

Exploratory Data Analysis

- Close Price Trend showed stock fluctuations which were useful to model future prices.
- Open, High, Low, Close and Adjusted Close were highly correlated.
- Volume is less correlated with prices indicating market activity.

Feature Engineering

- We extracted Date-based features from Date column to capture patterns.
- We used Moving Averages (MA) to smooth out noise and highlight trends.
- NaN values were dropped

Model Selection

- We used train_test_split() to split the dataset into Training set (80%) and Test set (20%).
- We used StandardScaler to standardized features to improve model convergence.

Model Training and Evaluation

- Firstly we trained multiple models to compare their performance –
    1. Linear Regression – Assumes a linear relation between features and target.
    2. Random Forest Aggressor – Handles non-linearity and feature interactions.
    3. Gradient Boosting Regressor – Sequentially builds trees to correct errors from previous trees.
    4. Support Vector Regression – Uses kernel functions to model non-linear relations.
- Evaluation Metrics –
    1. Mean Squared Error (MSE)
    2. Root Mean Squared Error (RMSE)
    3. Mean Absolute Error (MAE)
    4. R^2 Score

Model Tuning and Optimization

- GridSearchCV – Tests different combinations of hyperparameters
- Time Series Split – Ensures training data always comes before test data to avoid lookahead bias.

Model Deployment

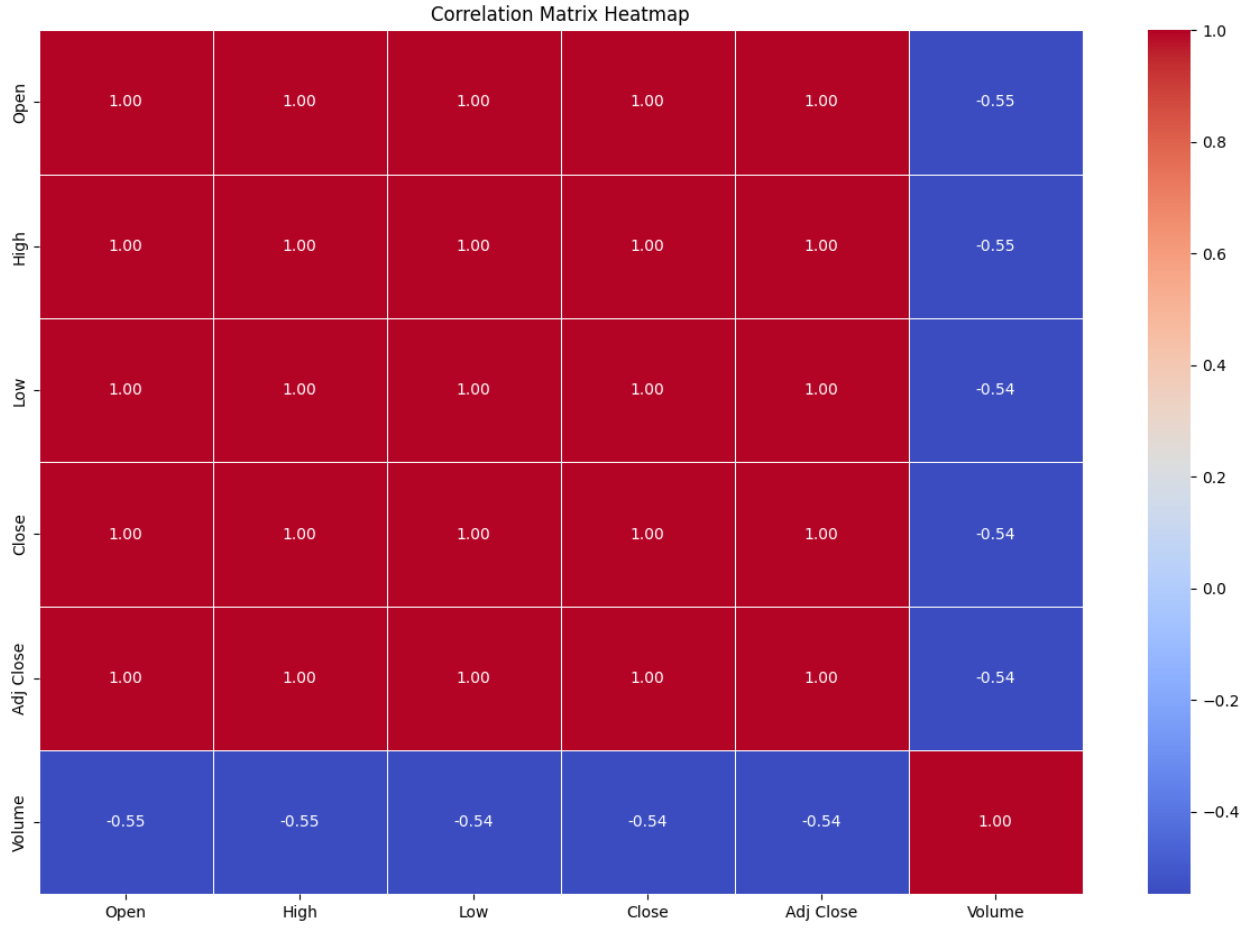
- joblib.dump() to store the trained model
- Endpoint ‘/predict’ to accept JSON input.
- Example Request –

```
{
  "Open": 150.0,
```

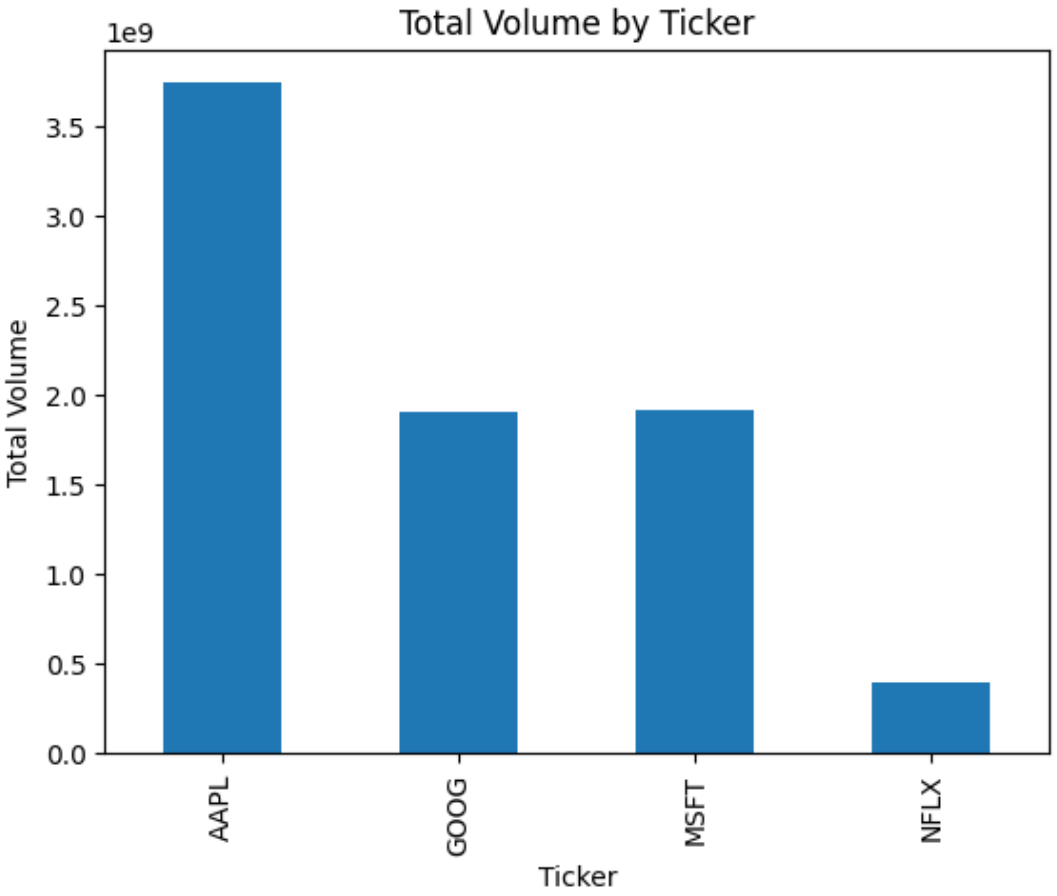
```
  "High": 152.0,
  "Low": 149.5,
  "Volume": 5000000,
  "Year": 2023,
  "Month": 7,
  "Day": 15,
  "Day_of_week": 5,
  "Daily_Return": 0.01,
  "Price_Change": 1.5,
  "High_Low_Pct": 1.67,
  "Close_Open_Pct": 1.0,
  "MA_5": 149.8,
  "MA_10": 148.5,
  "MA_20": 147.2,
  "Volatility": 0.015
}
```


Response –


```
{
  "predicted_price": 151.23
}
```

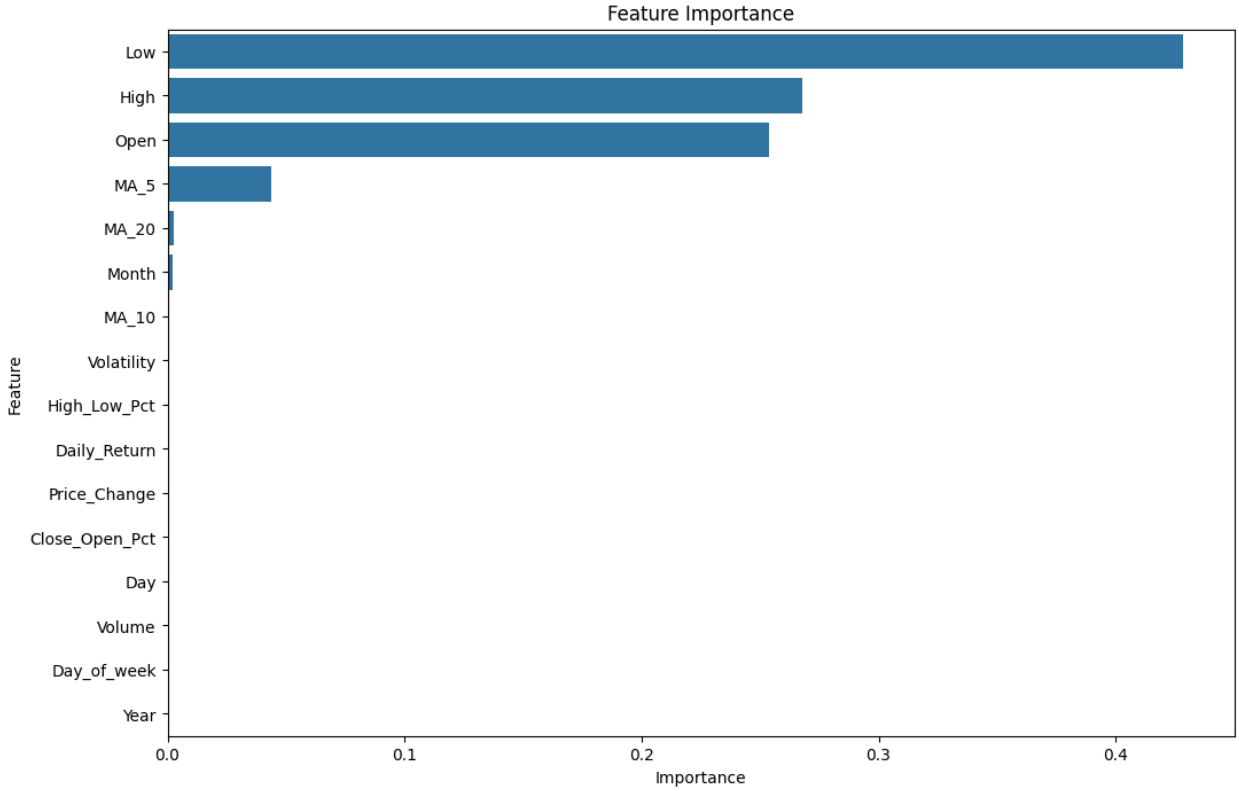




Closing Price Distribution
Frequency
Closing Price
0
5
10
15
20
25
30
100
150
200
250
300
350


Total Volume by Ticker
1e9
Total Volume
Ticker
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
AAPL
GOOG
MSFT
NFLX


Volume vs. Closing Price
Closing Price
350
300
250
200
150
100
0.0
0.2
0.4
0.6
0.8
1.0
Volume
1e8


Closing Price Distribution
Closing Price
350
300
250
200
150
100
1


Feature Importance
Low
High
Open
MA_5
MA_20
Month
MA_10
Volatility
High_Low_Pct
Daily_Return
Price_Change
Close_Open_Pct
Day
Volume
Day_of_week
Year
Feature
0.0
0.1
0.2
0.3
0.4
Importance