**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. R2 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

}











