# Stock Price Prediction

Using AI

## ✍️ Abstract

This project focuses on predicting stock prices using machine learning techniques. We used historical data of Apple Inc. (AAPL) collected from Yahoo Finance and applied three different algorithms: Linear Regression, Decision Tree Regressor, and Random Forest Regressor. The goal was to forecast the next day's closing price of the stock. The models were evaluated using regression metrics such as MAE, RMSE, and $R^2$ Score, and the direction of price movement was also assessed using accuracy, precision, recall, and F1 Score.

Predicting stock prices is an important aspect of financial analytics and can help investors make better trading decisions. With the rise of machine learning, it is now possible to detect patterns and trends in historical data that are difficult to identify with traditional methods. This project demonstrates how machine learning can enhance financial forecasting.

## 📝 Introduction

Stock price prediction is a widely studied topic in finance and data science. Investors use forecasts to make better decisions about buying or selling stocks. However, the stock market is influenced by various unpredictable factors such as political events, global market trends, company performance, and investor sentiment. This makes it difficult for traditional methods like moving averages or ARIMA models to deliver accurate predictions.

In recent years, machine learning has become a popular tool in financial analytics. These models can handle large datasets, identify hidden patterns, and learn from historical data. This project aims to apply machine learning models to predict the future closing price of Apple Inc. stock using various features extracted from the data. By comparing different models, we aim to determine which one performs best for this task.

## 💡 Problem Statement

Stock prices are highly volatile and are affected by numerous factors such as news, economic conditions, company performance, and investor behavior. Predicting stock prices is complex due to these dynamic changes and the non-linear nature of market behavior. Traditional models like ARIMA or simple moving averages often fail to capture this complexity.

This project aims to create a machine learning-based model that can learn from historical stock data and predict the future closing price of a stock with good accuracy. In addition to predicting prices, the model also evaluates whether the stock price will go up or down the next day, providing a classification-based perspective for trading decisions.

## 🔗 Objectives

1. **To collect and clean historical stock data:** The first step in any data-driven project is to gather reliable data. We used the Yahoo Finance API to download historical stock data for Apple Inc. This data includes features like open, high, low, close prices, and trading volume.
2. **To perform Exploratory Data Analysis (EDA):** EDA helps in understanding the structure and behavior of the dataset. In this project, we used EDA to visualize trends, detect anomalies, check for missing values, and examine relationships between variables. Charts like line plots, histograms, and correlation heatmaps were used to uncover important insights.
3. **To apply machine learning algorithms for stock price prediction:** The core of the project is to build and evaluate models. We implemented Linear Regression, Decision Tree Regressor, and Random Forest Regressor to predict the next day's closing price.
4. **To compare the performance of different models using evaluation metrics:** Each model was evaluated using regression metrics such as MAE, RMSE, and $R^2$ score. Additionally, we converted the regression output into a binary classification problem to assess direction-based accuracy using accuracy, precision, recall, and F1 score.
5. **To analyze stock movement direction using classification metrics:** Knowing whether the stock price will increase or decrease is valuable for

investors. We evaluated the predicted trend (up or down) and compared it to the actual movement to determine directional accuracy.

## 📊 Methodology

1. **Data Collection:** We used the Yahoo Finance API to download Apple Inc. (AAPL) stock data from 2020 to 2021. The dataset includes columns like Open, High, Low, Close, Adjusted Close, and Volume but we have choose only close and volume columns as they are relevant for our project.

```
stock_data = yf.download("AAPL", start="2020-01-01", end="2021-12-31")
stock_data = stock_data[['Close', 'Volume']]  # Keep only relevant columns
```

2. **Exploratory Data Analysis (EDA):** We visualized trends using line charts, examined distributions using histograms, and checked correlations using heatmaps. This helped us understand the underlying patterns in the data.

   First 5 Rows ->

```
# Display first few rows
print("📌 First 5 rows of the dataset:")
display(stock_data.head())
```

```
📌 First 5 rows of the dataset:
```

| Price | Close | Volume |
|---|---|---|
| Ticker | AAPL | AAPL |
| Date | | |
| 2020-01-02 | 72.716080 | 135480400 |
| 2020-01-03 | 72.009125 | 146322800 |
| 2020-01-06 | 72.582893 | 118387200 |
| 2020-01-07 | 72.241547 | 108872000 |
| 2020-01-08 | 73.403641 | 132079200 |

Total rows and Columns ->

Rows and Coloumns

```
[ ]  print("Shape of the dataset:", stock_data.shape)
```

```
Shape of the dataset: (504, 2)
```

Description ->

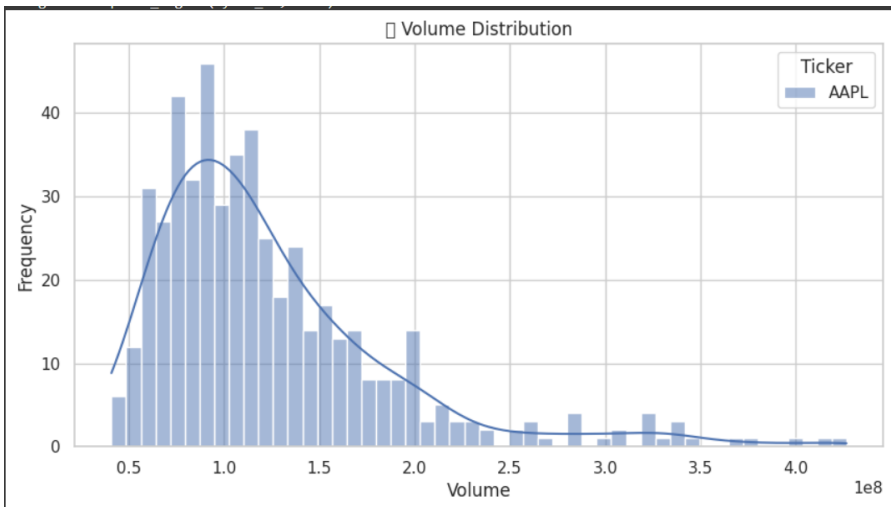
```
# Summary statistics
print("\n📌 Summary Statistics:")
display(stock_data.describe())
```

📌 Summary Statistics:

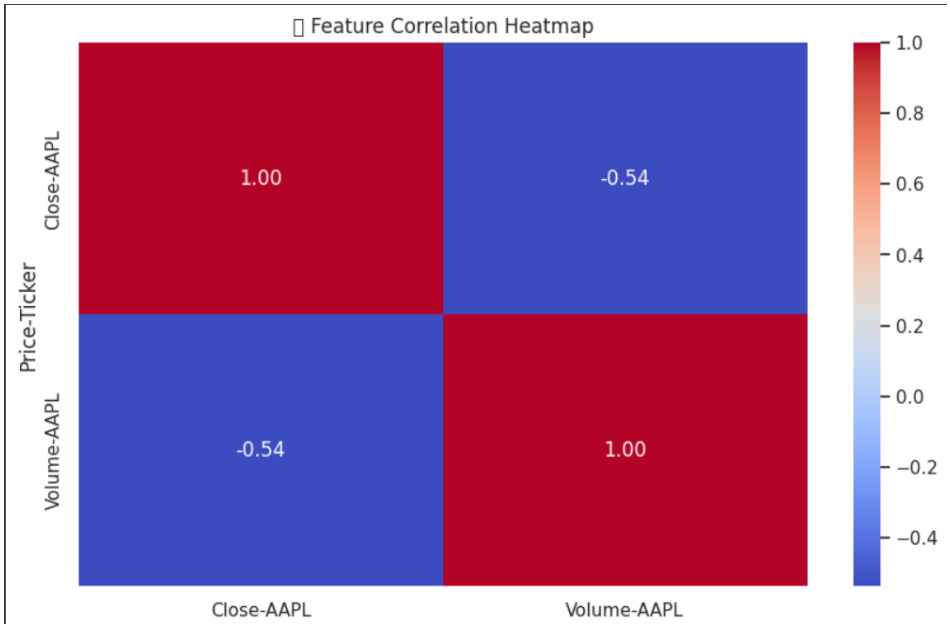
| Price | Close | Volume |
|---|---|---|
| Ticker | AAPL | AAPL |
| count | 504.000000 | 5.040000e+02 |
| mean | 115.353462 | 1.242302e+08 |
| std | 29.085514 | 6.316090e+07 |
| min | 54.449894 | 4.100000e+07 |
| 25% | 89.087120 | 8.119942e+07 |
| 50% | 120.966892 | 1.088506e+08 |
| 75% | 138.333973 | 1.478633e+08 |
| max | 177.228836 | 4.265100e+08 |

Closing Price over time ->

Volume Distribution ->




Correlation heatmap ->



3. **Data Preprocessing:** We handled missing values caused by moving average calculations, normalized features when required, and shifted the closing price column to set it as the next day's prediction target.

```
print("\n📌 Missing Values:")
print(stock_data.isnull().sum())
```

```
📌 Missing Values:
Price   Ticker
Close   AAPL      0
Volume  AAPL      0
dtype: int64
```

4. **Train-Test Split:** We split the data into 80% for training and 20% for testing using scikit-learn's train_test_split function.

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

5. **Model Building:**
   - **Linear Regression:** A basic model assuming linear relationships. Useful for baseline comparison.

```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
```

   - **Decision Tree Regressor:** A non-linear model that splits data based on feature values to predict outcomes.

```
dt_model = DecisionTreeRegressor(max_depth=3)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
```

- **Random Forest Regressor:** An ensemble model that uses multiple decision trees to improve accuracy and reduce overfitting.

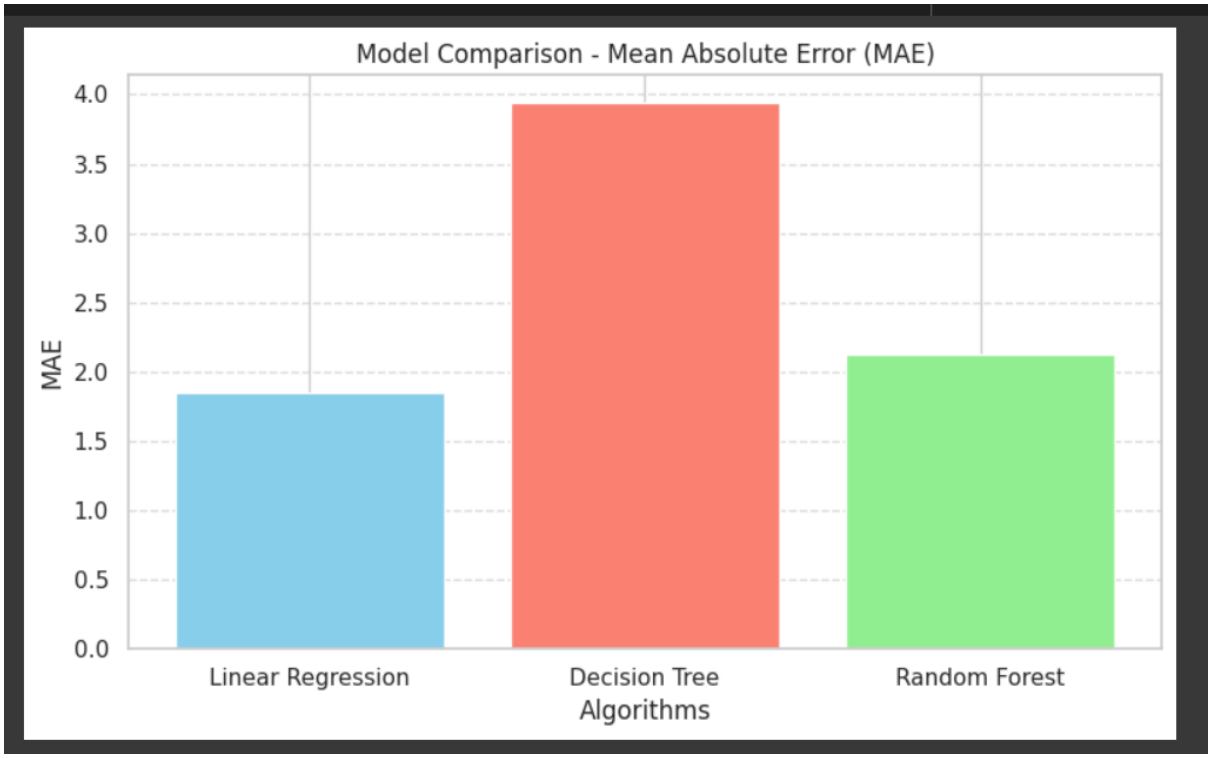
```
rf_model = RandomForestRegressor(n_estimators=10)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

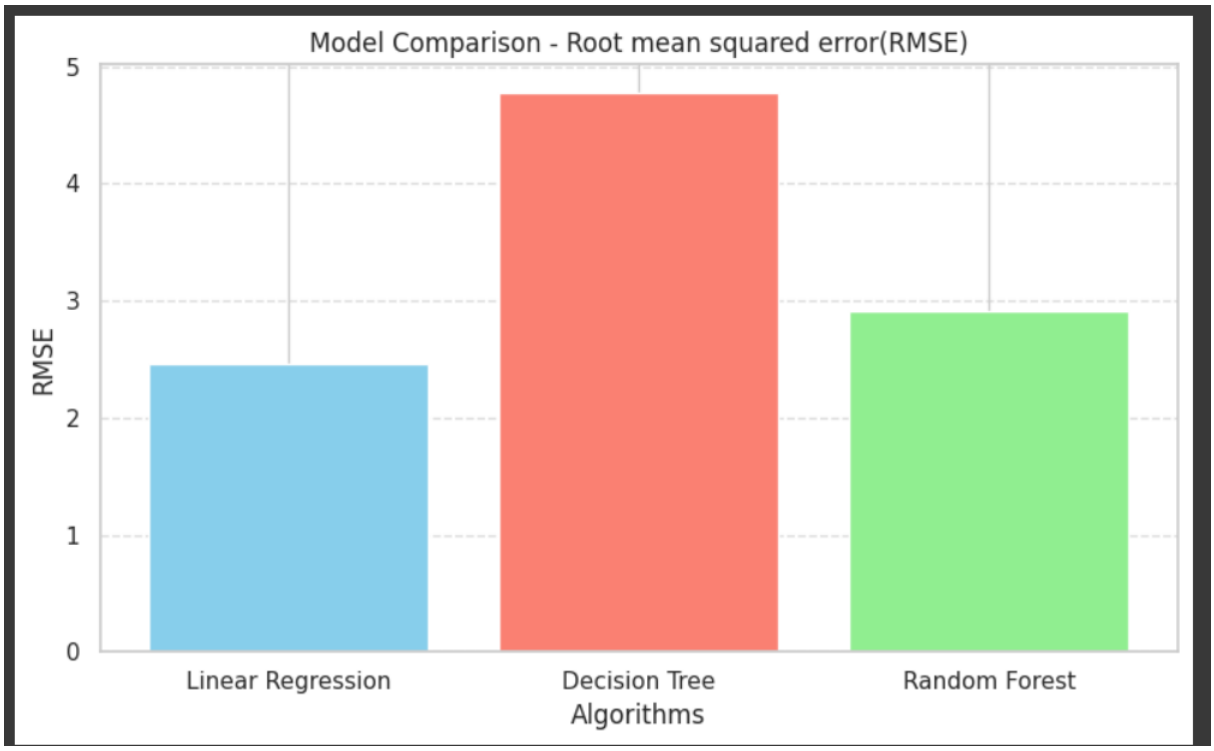
6. **Evaluation:** We evaluated the models using:
   - Regression metrics: MAE, RMSE, $R^2$ Score
   - Classification metrics: Accuracy, Precision, Recall, F1 Score (based on stock movement direction)

## 📈 Results

**Regression Metrics:**

| Model | MAE | RMSE | $R^2$ Score |
|---|---|---|---|
| Linear Regression | 1.845 | 2.460 | 0.9944 |
| Decision Tree | 3.942 | 4.776 | 0.9790 |
| Random Forest | 2.127 | 2.904 | 0.9922 |


Model Comparison - Mean Absolute Error (MAE)
MAE
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Linear Regression
Decision Tree
Random Forest
Algorithms


Model Comparison - Root mean squared error(RMSE)
RMSE
5
4
3
2
1
0
Linear Regression
Decision Tree
Random Forest
Algorithms


**Analysis:** Based on the results, Linear Regression performed the best for predicting the next day's stock price of Apple Inc. This is because the stock's price movements are relatively steady and follow a mostly linear pattern over time, making simpler models like Linear Regression more effective than more complex tree-based models. Random Forest also performed exceptionally well, capturing non-linear patterns slightly better than Decision Trees, but the simplicity and stability of Linear Regression proved to be the most effective for this specific dataset.

## 📌 Conclusion

```
📊 Direction-Based Metrics for Linear Regression:
Accuracy : 0.95
Precision: 0.9583333333333334
Recall   : 0.9387755102040817
F1 Score : 0.9484536082474226
```

```
📊 Direction-Based Metrics for Decision Tree:
Accuracy : 0.91
Precision: 1.0
Recall   : 0.8163265306122449
F1 Score : 0.898876404494382
```

```
📌 Direction-Based Classification Metrics (Random Forest):
Accuracy: 0.94
Precision: 0.9574468085106383
Recall: 0.9183673469387755
F1 Score: 0.9375
```

This project demonstrated how machine learning models can be effectively used to predict stock prices and trends. Among the three models tested, Linear Regression gave the best performance in terms of accuracy, robustness, and generalization on the given dataset. It was able to handle the steady and relatively linear behavior of Apple stock prices better than tree-based models.

While no model can completely eliminate the uncertainty of the stock market, machine learning provides tools that make forecasting more data-driven and reliable. In real-world applications, combining models and using more advanced features like news sentiment or macroeconomic indicators could further improve performance.

## References


- Yahoo Finance: https://finance.yahoo.com
- Scikit-learn Documentation: https://scikit-learn.org
- Python Libraries: pandas, numpy, matplotlib, seaborn, yfinance
- Towards Data Science Articles on Stock Forecasting
- Machine Learning Documentations