Stock Market Prediction

1. Introduction

This documentation presents an analysis of stock market data using machine learning models. The analysis includes data collection, preprocessing, exploratory data analysis (EDA), and the implementation of various machine learning algorithms for stock price prediction. For this project, the data covers the price and volume of shares of 31 NASDAQ companies in the year 2022

Objective:

The primary objective of this project is to build and evaluate machine learning models that can predict stock prices with high accuracy. Specifically, we seek to achieve the following goals:

- Develop predictive models using historical stock market data.
- Evaluate the performance of different machine learning algorithms for stock price prediction.
- Provide insights into the trends and patterns observed in the stock market data.

Statement:

The problem at hand is to develop predictive models capable of accurately forecasting future stock prices in the volatile and uncertain stock market environment.

2. Importing Libraries

Libraries such as pandas, NumPy, matplotlib, seaborn, and sklearn are imported for data manipulation, visualization, and machine learning tasks.

3. Data Collection and Preprocessing

- Data is loaded from a CSV file into a pandas DataFrame.
- Irrelevant columns are dropped, and missing values are handled.
- The data is converted to datetime format for time series analysis.
- Summary statistics are computed to understand the distribution of the data.

- Functions used:
  → pd.read_csv()
  → df.drop()
  → df.isna().sum()
  → df.info()
  → df.fillna()
  → pd.to_datetime()
  → df.head()
  → df.describe()

4. Exploratory Data Analysis (EDA)

4.1 Time Series Analysis/ Change in closing Price

Closing prices of different companies are plotted over time to observe trends and patterns.

Each company's closing price is visualized on individual subplots.

Code:

```
plt.figure(figsize=(15, 30))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    comp_data = data[df['ticker'] == company]
    plt.subplot(8, 4, i)
    plt.plot(comp_data['date'], comp_data['close'])
    plt.ylabel('Close')
    plt.xlabel('Date')
    plt.xticks(rotation=45)
    plt.title(f"Closing Price of {company_list[i - 1]}")
    plt.grid()

plt.tight_layout()
```

Output:

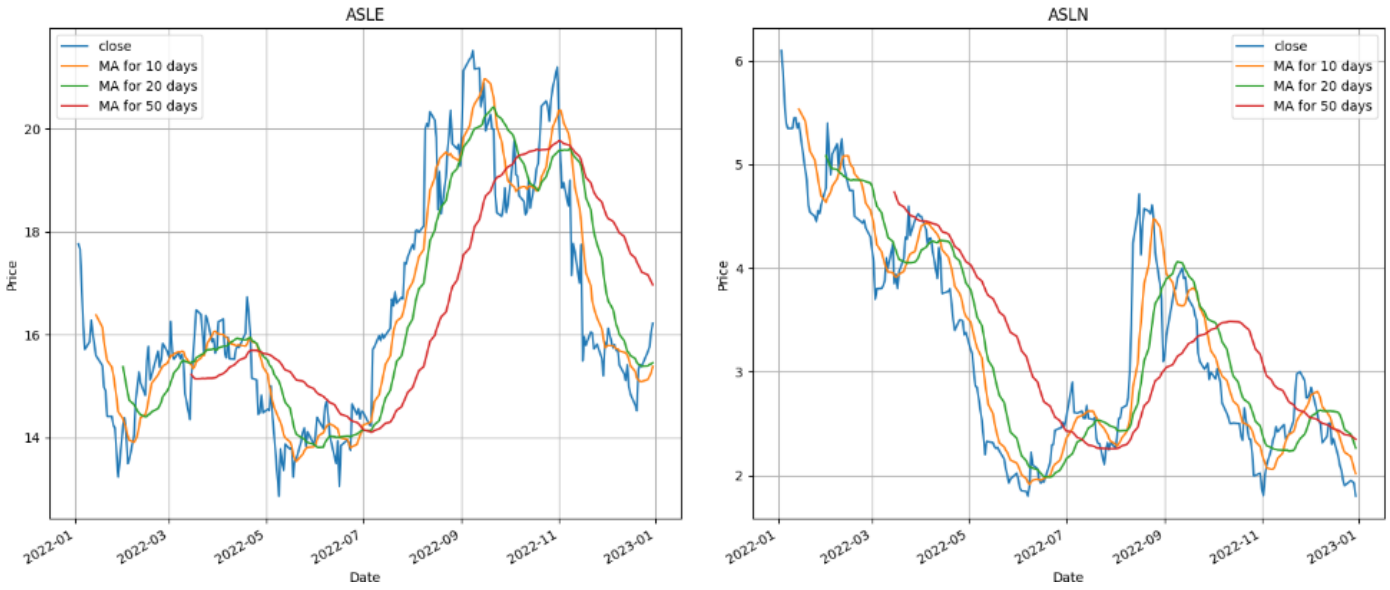
4.2 What was the moving average of the various stocks?

The moving average (MA) is a technique in technical analysis that smooths out price data by calculating the average price over a specified time period. It helps traders identify trends and reduce noise in the data.

Code:

```
ma_days = [10, 20, 50]

num_rows = 16
num_cols = 2

num_tickers = len(company_list)

num_plots = num_rows * num_cols
num_subplots = min(num_tickers, num_plots)

fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(15, 100))

for i, ticker in enumerate(company_list[:num_subplots]):
    row = i // num_cols
    col = i % num_cols

    ticker_data = data[df['ticker'] == ticker]

    for ma_day in ma_days:
        column_name = f"MA for {ma_day} days"
        ticker_data[column_name] = ticker_data['close'].rolling(ma_day).mean()

    ticker_data.plot(ax=axes[row, col], x='date', y=['close'] + [f"MA for {ma_day} days" for ma_day in ma_days],
                     title=ticker, xlabel='Date', ylabel='Price', grid=True)

if num_tickers < num_plots:
    fig.delaxes(axes.flatten()[num_tickers])

fig.tight_layout()
plt.show()
```

Output:



The graph visualizes the close prices and moving averages (10, 20, and 50 days) for each of the 31 tickers in the dataset.

5. Machine Learning Models

5.1 Linear Regression

Explanation: Linear regression is a simple and widely used regression technique that models the relationship between the independent variables (features like opening price, high, low, volume) and the dependent variable (target - closing price.) by fitting a linear equation to observed data.

Application: Linear regression is suitable when the relationship between the features and target is expected to be linear. It's a good starting point for modeling and provides insights into the direction and strength of the relationships.

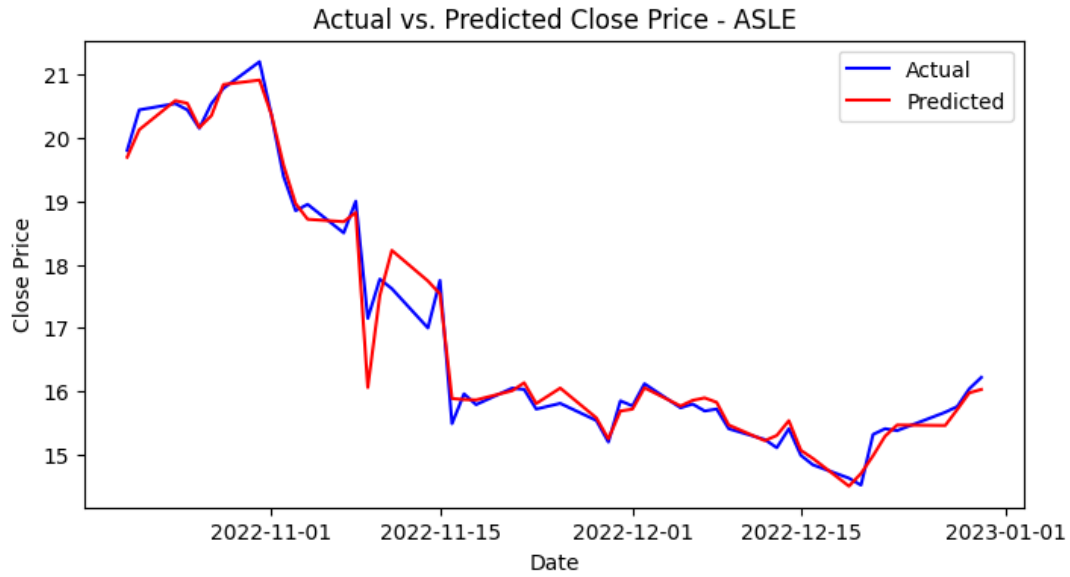
For Stock market prediction:

1. **Model Training and Evaluation:**
   Linear regression models are individually trained and meticulously evaluated for each company in the dataset.
2. **Performance Metrics Calculation:**
   Two crucial metrics, namely Mean Squared Error (MSE) and R-squared (R^2) scores, are computed for each model. These metrics serve as quantitative measures of the model's accuracy and predictive power.
3. **Visual Insights:**
   Actual vs. predicted closing prices are visualized for each company.

Code:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

mse_Linear_list = []
r2_score_Linear_list = []

for ticker in company_list:
    ticker_data = company_data[ticker]

    train_data = ticker_data.iloc[:-50]
    test_data = ticker_data.iloc[-50:]

    X_train, y_train = train_data[['open', 'high', 'low', 'volume']], train_data['close']
    X_test, y_test = test_data[['open', 'high', 'low', 'volume']], test_data['close']

    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    dates_test = data.iloc[X_test.index]['date']

    plt.figure(figsize=(8,4))
    plt.plot(dates_test, y_test, label='Actual', color='blue')
    plt.plot(dates_test, y_pred, label='Predicted', color='red')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.title(f'Actual vs. Predicted Close Price - {ticker}')
    plt.legend()
    plt.show()

    r2_score_Linear = r2_score(y_test, y_pred)
    mse_Linear = mean_squared_error(y_test, y_pred, squared=False)
    r2_score_Linear_list.append(r2_score_Linear)
    mse_Linear_list.append(mse_Linear)
    print(f'r2_score for {ticker}: {r2_score_Linear:.5f}')
    print(f'MSE for {ticker}: {mse_Linear:.5f}')
```

Output:




```
r2_score for ASLE: 0.98348
MSE for ASLE: 0.25735
```

5.2 Ridge Regression

Explanation: Ridge regression is a regularization technique that extends linear regression by adding a penalty term to the loss function. This penalty term helps prevent overfitting by shrinking the coefficients towards zero. It's particularly useful when dealing with multicollinearity among the predictor variables.

Application: Ridge regression is beneficial when the dataset has multicollinearity issues, meaning some features are highly correlated with each other. It helps stabilize the model and reduces the variance of the estimates, leading to better generalization performance.

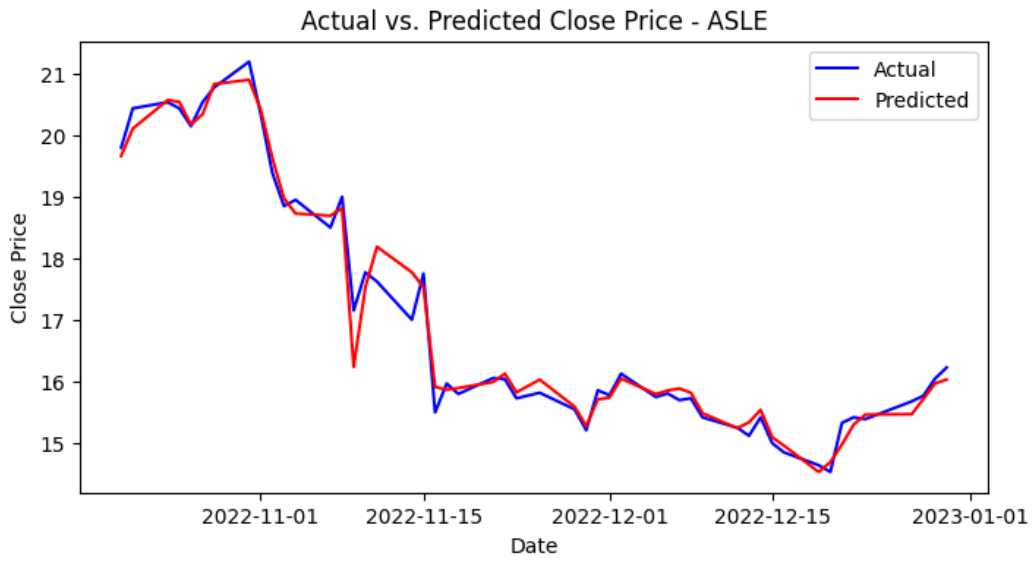
For Stock market prediction:

1. **Model Training and Evaluation:**
   Linear regression models are individually trained and meticulously evaluated for each company in the dataset.
2. **Performance Metrics Calculation:**
   Two crucial metrics, namely Mean Squared Error (MSE) and R-squared (R^2) scores, are computed for each model. These metrics serve as quantitative measures of the model's accuracy and predictive power.
3. **Visual Insights:**
   Actual vs. predicted closing prices are visualized for each company.

Code:

```
from sklearn.linear_model import Ridge
mse_Ridge_list = []
r2_score_Ridge_list = []

for ticker in company_list:
    ticker_data = company_data[ticker]

    train_data = ticker_data.iloc[:-50]
    test_data = ticker_data.iloc[-50:]

    X_train, y_train = train_data[['open', 'high', 'low', 'volume']], train_data['close']
    X_test, y_test = test_data[['open', 'high', 'low', 'volume']], test_data['close']

    model = Ridge()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    dates_test = data.iloc[X_test.index]['date']

    plt.figure(figsize=(8, 4))
    plt.plot(dates_test, y_test, label='Actual', color='blue')
    plt.plot(dates_test, y_pred, label='Predicted', color='red')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.title(f'Actual vs. Predicted Close Price - {ticker}')
    plt.legend()
    plt.show()

    r2_score_Ridge = r2_score(y_test, y_pred)
    mse_Ridge = mean_squared_error(y_test, y_pred, squared=False)
    r2_score_Ridge_list.append(r2_score_Ridge)
    mse_Ridge_list.append(mse_Ridge)
    print(f'r2_score for {ticker}: {r2_score_Ridge:.5f}')
    print(f'MSE for {ticker}: {mse_Ridge:.5f}')
```

Output:




```
r2_score for ASLE: 0.98482
MSE for ASLE: 0.24667
```

5.3 Random Forest Regression

Explanation: Random forest regression is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mean prediction of the individual trees. Each tree in the forest is trained on a random subset of the training data and a random subset of the features.

Application: Random forest regression is robust to overfitting and capable of capturing complex nonlinear relationships in the data. It's suitable for high-dimensional datasets with both numerical and categorical features.

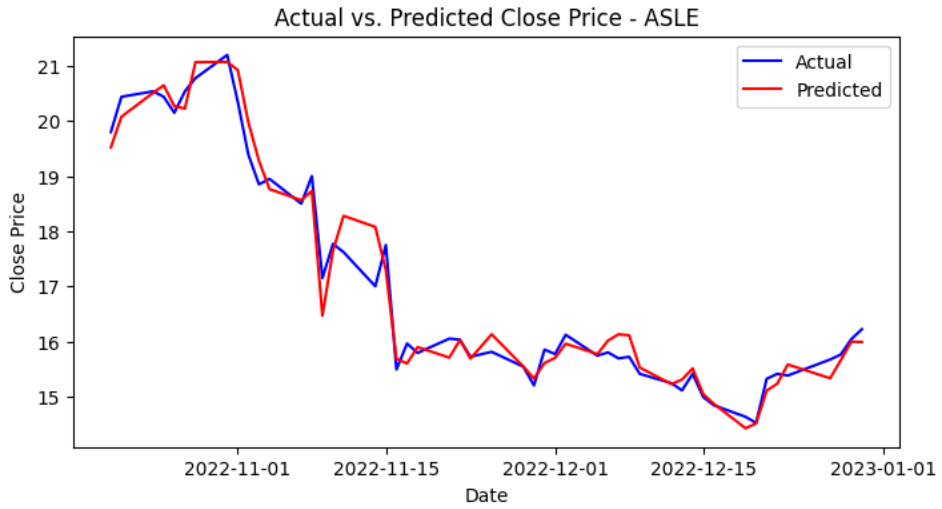
For Stock market prediction:

1. **Model Training and Evaluation:**
   Linear regression models are individually trained and meticulously evaluated for each company in the dataset.
2. **Performance Metrics Calculation:**
   Two crucial metrics, namely Mean Squared Error (MSE) and R-squared (R^2) scores, are computed for each model. These metrics serve as quantitative measures of the model's accuracy and predictive power.
3. **Visual Insights:**
   Actual vs. predicted closing prices are visualized for each company.

Code:

```
from sklearn.ensemble import RandomForestRegressor
mse_RandomForest_list = []
r2_score_RandomForest_list = []

for ticker in company_list:
    ticker_data = company_data[ticker]

    train_data = ticker_data.iloc[:-50]
    test_data = ticker_data.iloc[-50:]

    X_train, y_train = train_data[['open', 'high', 'low', 'volume']], train_data['close']
    X_test, y_test = test_data[['open', 'high', 'low', 'volume']], test_data['close']

    model = RandomForestRegressor()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    dates_test = data.iloc[X_test.index]['date']

    plt.figure(figsize=(8, 4))
    plt.plot(dates_test, y_test, label='Actual', color='blue')
    plt.plot(dates_test, y_pred, label='Predicted', color='red')
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.title(f'Actual vs. Predicted Close Price - {ticker}')
    plt.legend()
    plt.show()

    r2_score_RandomForest = r2_score(y_test, y_pred)
    mse_RandomForest = mean_squared_error(y_test, y_pred, squared=False)
    r2_score_RandomForest_list.append(r2_score_Linear)
    mse_RandomForest_list.append(mse_Linear)
    print(f'r2_score for {ticker}: {r2_score_RandomForest:.5f}')
    print(f'MSE for {ticker}: {mse_RandomForest:.5f}')
```

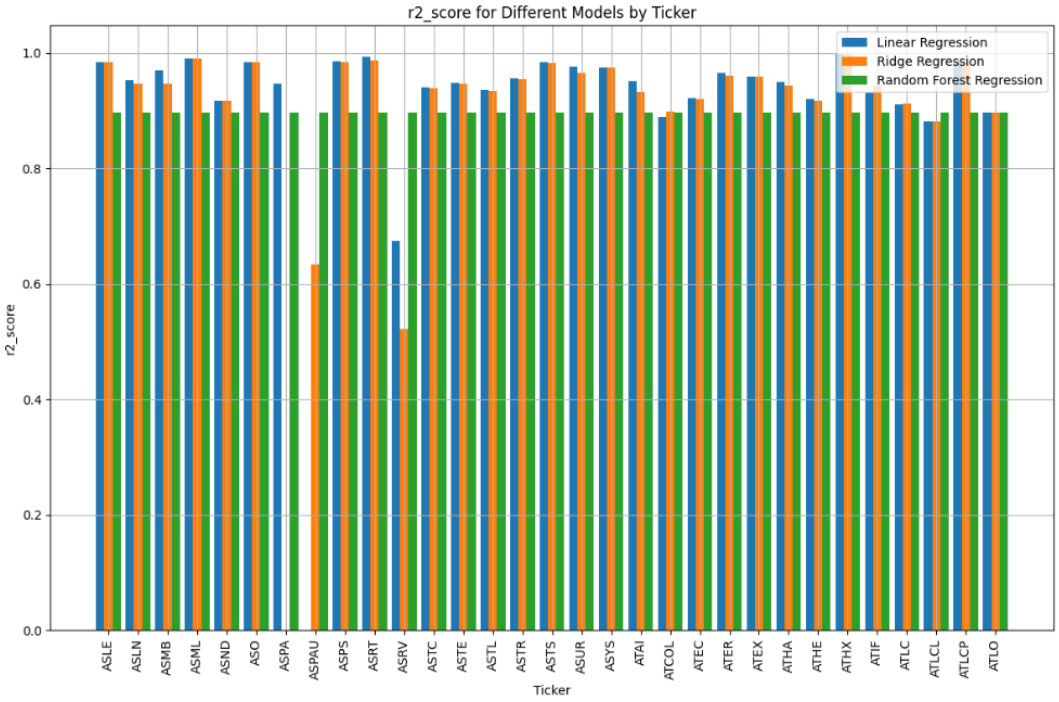
Output:




```
r2_score for ASLE: 0.97487
MSE for ASLE: 0.31740
```

6. Analysis of Model Performance

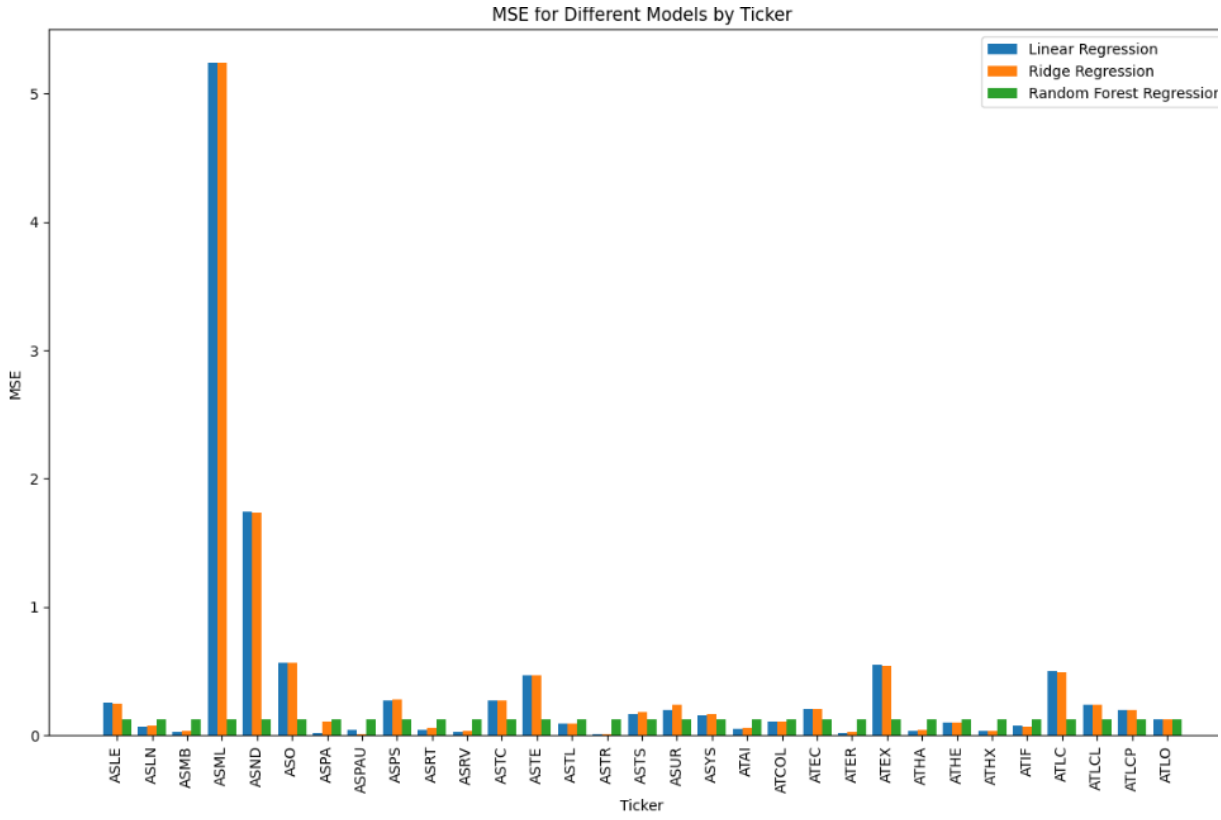
R^2 score: The performance of different machine learning models is compared using bar charts. Mean R^2 score of linear Regression, Ridge Regression, and Random Forest Regression are computed.



```
mean of R2 scores of linear regression 0.7765220324812391
mean of R2 scores of Ridge regression 0.8739864328884659
mean of R2 scores of Random Forest regression 0.8966612935629498
```

The Random Forest regression model demonstrates the highest mean R-squared score, indicating superior predictive performance compared to Linear Regression and Ridge Regression .

MSE: The performance of different machine learning models is compared using bar charts, where the Mean Squared Error (MSE) for Linear Regression, Ridge Regression, and Random Forest Regression models is computed and visualized.




```
mean of mse of linear regression 0.7765220324812391
mean of mse of Ridge regression 0.8739864328884659
mean of mse of Random Forest regression 0.8966612935629498
```

random forest regression model shows best MSE values for all tickers as compared to linear and ridge regression.

7. Conclusion

In the evaluation of machine learning models for stock market prediction, the Random Forest regression model emerges as the top performer, boasting the highest mean R-squared score of 0.8967. This signifies its superior predictive accuracy in capturing the variance in stock prices compared to both Linear Regression (0.7765) and Ridge Regression (0.8740). Therefore, for stakeholders seeking robust and accurate forecasting models, the Random Forest regression model stands out as the optimal choice.