PREDICTIVE MODELING OF STOCK PRICES

Project by : Teja Vuppu, Begimai Zhumakova, Swetha Voora

Abstract

This project aims to develop a machine learning model for predicting future stock prices using historical market prices and other relevant market data. The report addresses the challenges of dealing with noisy and complex data, and identifying the most important features for predicting stock prices. The success of the model is evaluated by its ability to outperform simple baseline models, generalize to new test data. The proposed model has significant potential to inform trading decisions and identify potential profit opportunities.

Introduction

In the modern world, the stock market is a highly dynamic and complex system that poses significant challenges for investors to make informed decisions. The ability to predict stock prices accurately is crucial for effective investment management. In recent years, data mining techniques such as time series analysis, machine learning, and deep learning have emerged as powerful tools for predicting stock prices. This research project aims to develop a stock price prediction model using various data mining techniques. The project will compare and analyze the performance of three different models: Linear Regression, Random Forest, and Long Short Term Memory (LSTM) methods on time series data. The goal is to gain insights into which technique is the most effective for predicting stock prices. To accomplish this, we will focus on analyzing the stock prices of the widely known company, Apple (AAPL), over the past few years. We will use historical data from Yahoo Finance to train and test our models. The results of this research will have significant implications for the investment community, providing valuable insights into which data mining technique can offer the most accurate stock price predictions.

Project Requirements

To ensure the successful completion of the project, the following requirements must be met:

1. Access to reliable and relevant financial data from Yahoo Finance, which can be obtained through the Yahoo Finance API or by downloading historical data from the website.

2. A Jupyter Notebook environment, which can be installed through Anaconda or directly through Python.

3. The installation of relevant Python packages for data analysis and visualization, such as pandas, numpy, matplotlib, seaborn, and scikit-learn.

4. A stable and fast Internet connection to ensure seamless data retrieval and analysis.

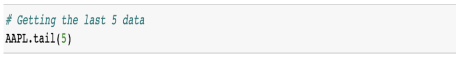
Additionally, the use of a containerization platform such as Docker can provide a more streamlined and reproducible environment for the project. The required packages and dependencies can be easily installed and managed through a Docker image.

Data Collection

In this project, we collected the stock data for Apple Inc. from Yahoo Finance using the yfinance package in Python. By setting the "max" parameter, we retrieved the entire daily historical data available for Apple Inc.

Dataset Import. We imported the dataset as shown in Figure 1.

```
# Importing the yfinance package
import yfinance as yf
# Importing the dataset from yfinance
AAPL = yf.Ticker("AAPL")
# Getting all time stock data
AAPL = AAPL.history(period = "max")
```

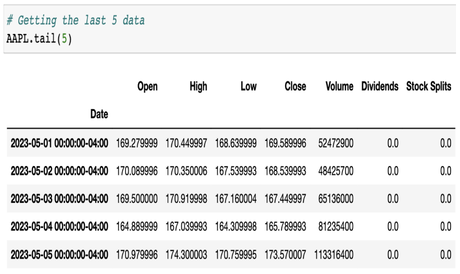


```
# Getting the last 5 data
AAPL.tail(5)
```

Fig. 1. Importing the Apple Inc. stock data

The data consists of various columns such as opening price, closing price, high price, low price, trading volume, and more for each day's trading. We utilized this data to train and evaluate the effectiveness of our stock price prediction models.

Dataset Contents. The contents of the dataset are shown in Figure 2.



```
# Getting the last 5 data
AAPL.tail(5)
```

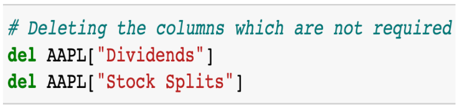
Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2023-05-01 00:00:00-04:00	169.279999	170.449997	168.639999	169.589996	52472900	0.0	0.0
2023-05-02 00:00:00-04:00	170.089996	170.350006	167.539993	168.539993	48425700	0.0	0.0
2023-05-03 00:00:00-04:00	169.500000	170.919998	167.160004	167.449997	65136000	0.0	0.0
2023-05-04 00:00:00-04:00	164.889999	167.039993	164.309998	165.789993	81235400	0.0	0.0
2023-05-05 00:00:00-04:00	170.979996	174.300003	170.759995	173.570007	113316400	0.0	0.0

Fig. 2. Contents of the Apple Inc. stock dataset

Data Preprocessing

In order to conduct a thorough analysis of historical stock price data for Apple Inc., it was essential to preprocess the data and eliminate any features that could potentially impact the analysis in an adverse manner. Accordingly, one of the primary preprocessing steps involved removing the "Dividends" and "Stock Splits" columns from the AAPL Dataframe, as these columns were determined to be extraneous to the stock price prediction based on the available information in the Dataframe.

Deleting Columns. Step1 of preprocessing is shown in Figure 3.

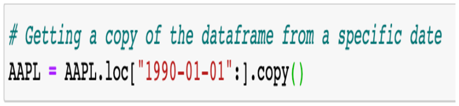


```
# Deleting the columns which are not required
del AAPL["Dividends"]
del AAPL["Stock Splits"]
```

Fig. 3. Deleting the extraneous columns

To enhance the accuracy and comprehensiveness of the analysis of historical stock prices for Apple Inc., we performed a series of preprocessing steps. One of these steps involved creating a new copy of the AAPL Dataframe, starting from 1990, using the loc function. This approach allowed us to focus on the stock market trend over the past 30 years, which provided a more detailed and informative view of the historical stock prices and market trends for Apple Inc.

using loc[]. Step2 of preprocessing are shown in Figure 4.



```
# Getting a copy of the dataframe from a specific date
AAPL = AAPL.loc["1990-01-01":].copy()
```

Fig. 4. using loc

The preprocessing steps were aimed at ensuring that the data was in a consistent and suitable format for analysis. The main objective was to eliminate any irrelevant or extraneous features and focus on a relevant and comprehensive time frame. This approach helped us optimize the accuracy and effectiveness of our analysis of the historical stock prices for Apple Inc.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial stage in any data analysis project, aimed at gaining insights into the data. In this study, we conducted EDA to gain insights into the Apple Inc. stock data obtained from Yahoo Finance. The purpose of this exercise was to understand the information contained in the dataset, its statistics, and shape. To start the EDA process, we examined the trend of Apple's stock price over the past two decades, which helped us to understand the general trend in the stock price of the company over time and identify any possible outliers or anomalies. Using the matplotlib library, we plotted a line chart to better visualize the stock price trend, enabling us to gain insights into the historical performance of the stock and identify any patterns or trends that may help inform our stock price predictions.

Visualization. Curve of the stock prices is shown in Figure 5.

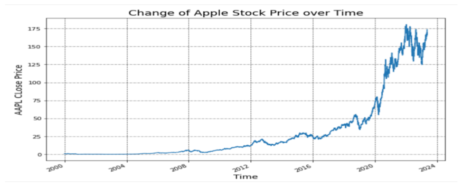


Fig. 5. Apple Inc. stock's curve

We observe that the stock performance has been increasing from 2008 to 2022.
Our analysis also involved creating correlation graphs to understand the relationship between various columns in the dataset. Specifically, we plotted the correlation between the 'Open', 'High', 'Close', 'Volume', and 'Close' columns using matplotlib.

Correlation. - between dependent and independent variables is shown in Figure 6.

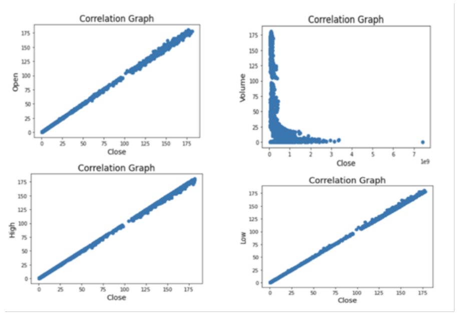
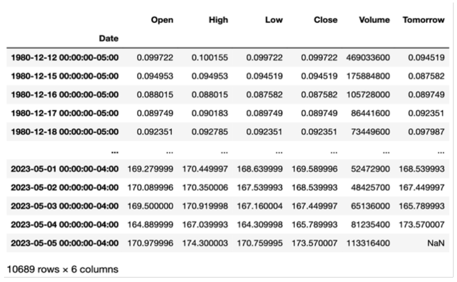


Fig. 6. Correlation graphs

These graphs helped us to identify any strong correlations between the variables and understand how they influence each other. We observed a strong positive correlation between the opening price, high price, and low price with the closing price, while the volume traded had a weak negative correlation with the closing price.

For this analysis, we added a new column called "**Tomorrow**" to the existing AAPL DataFrame, which contains the closing price of the previous day for each row. This was achieved using the shift() function which moves each row down by one and fills the first row with a NaN value.

New Columns. The added column is shown in Figure 7.



	Open	High	Low	Close	Volume	Tomorrow
Date						
1980-12-12 00:00:00-05:00	0.099722	0.100155	0.099722	0.099722	469033600	0.094519
1980-12-15 00:00:00-05:00	0.094953	0.094953	0.094519	0.094519	175884800	0.087582
1980-12-16 00:00:00-05:00	0.088015	0.088015	0.087582	0.087582	105728000	0.089749
1980-12-17 00:00:00-05:00	0.089749	0.090183	0.089749	0.089749	86441600	0.092351
1980-12-18 00:00:00-05:00	0.092351	0.092785	0.092351	0.092351	73449600	0.097987
...	...	...	...	...	...	...
2023-05-01 00:00:00-04:00	169.279999	170.449997	168.639999	169.589996	52472900	168.539993
2023-05-02 00:00:00-04:00	170.089996	170.350006	167.539993	168.539993	48425700	167.449997
2023-05-03 00:00:00-04:00	169.500000	170.919998	167.160004	167.449997	65136000	165.789993
2023-05-04 00:00:00-04:00	164.889999	167.039993	164.309998	165.789993	81235400	173.570007
2023-05-05 00:00:00-04:00	170.979996	174.300003	170.759995	173.570007	113316400	NaN

10689 rows × 6 columns

Fig. 7. Added Tomorrow column

Once the new column was added, we proceeded to create two separate plots using the matplotlib library to show the closing price of Apple's stock (AAPL) for each day in the DataFrame, and the closing price of the previous day. Both plots were created using the bar() function, and the x-axis was set to the index date of the AAPL DataFrame, while the y-axis was set to the closing price of AAPL for each day in the first plot and the closing price of the previous day in the second plot. The first plot was labeled as "AAPL - Today's Close vs Date - Visualization," while the second plot was labeled as "AAPL - Tomorrow's Close vs Date - Visualization". The data used for the plots covers a period of time starting from 1980.

Subplots. The visualizing subplots for the comparison are shown in Figure 8.

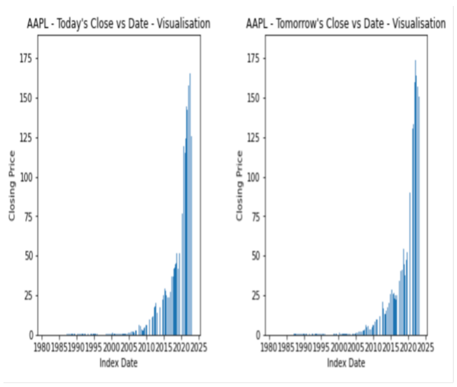


Fig. 8. Visualization graphs

Trading volume is a significant metric for traders, as it indicates the amount of a security being traded over a specific period. In our analysis, we plotted the volume sales for AAPL over the past one year using matplotlib. By visualizing the trading volume for AAPL, we gained a better understanding of the level of demand for the stock over time. This can help traders to identify patterns in trading volume that may be indicative of changes in market sentiment or upcoming trends.

Trading Volumne Metric. visualization is shown in Figure 9.

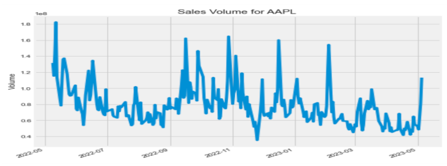


Fig. 9. Visualization graphs

The Moving Average (MA) is a widely-used technical analysis tool that helps in smoothing out price data by calculating a constantly updated average price. We computed the moving average for the Apple stock over a span of 10, 20, and 50 days using the pandas library to gain insights into the stock's performance trends.

Moving Average. The MA plot is shown in Figure 10.

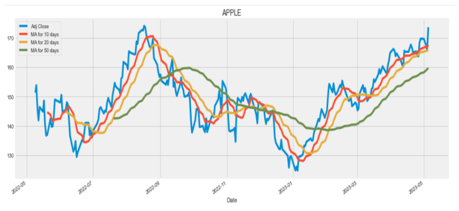


Fig. 10. Daily returns Analysis

As part of our analysis, we examined the risk associated with the Apple stock by focusing on the daily changes in stock prices rather than just its absolute value. To do this, we utilized the pandas library to retrieve the daily returns for Apple stock. We further used seaborn library to create both histogram and KDE (Kernel Density Estimation) plot. These plots helped us to understand the distribution of daily returns and the probability density function, respectively, providing valuable insights into the risk associated with investing in Apple stock.

Daily Returns. The Risk Analysis using Daily Returns is shown in Figure 11.

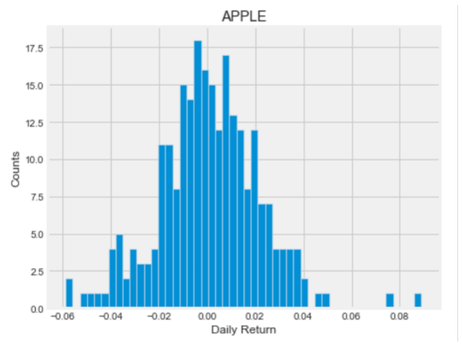


Fig. 11. Daily Returns of Apple Inc. stocks

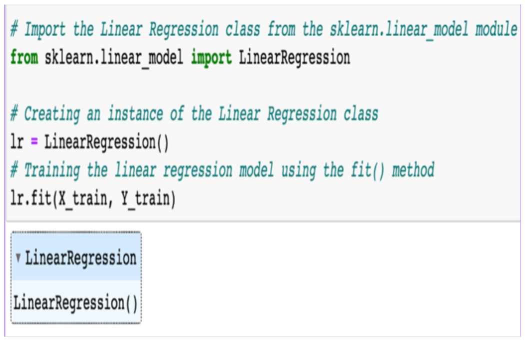
Model Selection

In order to evaluate the performance of the Apple stock, we employed three different models.

1. Linear Regression, which assumes a linear relationship between the input variables and the target variable. This model is often used in finance and economics to make predictions based on historical data. By fitting a line to the data points, it seeks to find a relationship between the input variables and the target variable. This model can be useful in understanding the trends of the stock over time and making predictions about future prices.

   Classifier. The LR classifier is shown in Figure 12.

2. Random Forest, which is an ensemble model that uses multiple decision trees to make predictions. Unlike Linear Regression, Random Forest is capable of handling non-linear relationships between variables, and it is less prone to overfitting than a single decision tree. It



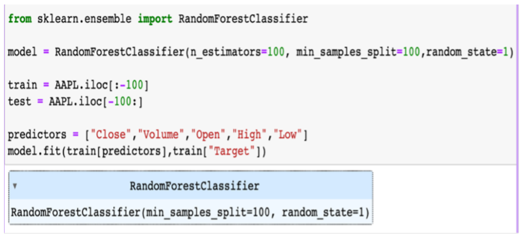
```
# Import the Linear Regression class from the sklearn.linear_model module
from sklearn.linear_model import LinearRegression

# Creating an instance of the Linear Regression class
lr = LinearRegression()
# Training the linear regression model using the fit() method
lr.fit(X_train, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Fig. 12. LR Model

is a popular choice for predicting stock prices because it can capture complex patterns in the data that may not be immediately obvious. By combining the predictions of multiple decision trees, it can provide more accurate and reliable predictions.

Classifier. The RF classifier is shown in Figure 13.



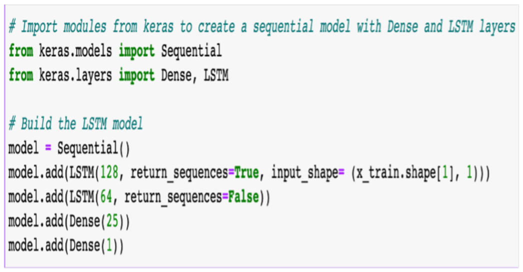
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, min_samples_split=100,random_state=1)

train = AAPL.iloc[:-100]
test = AAPL.iloc[-100:]

predictors = ["Close","Volume","Open","High","Low"]
model.fit(train[predictors],train["Target"])
```

```
▾ RandomForestClassifier
RandomForestClassifier(min_samples_split=100, random_state=1)
```

Fig. 13. RF Model

3. Long Short Term Memory (LSTM), which is a type of recurrent neural network that is specifically designed for time series data. LSTM is capable of learning long-term dependencies in the data, which makes it a powerful tool for predicting stock prices. It is particularly useful for predicting stock prices because it can capture the relationships between different variables over time.

Classifier. The LSTM classifier is shown in Figure 14.



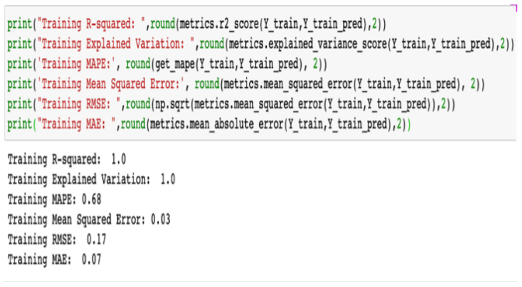
```
# Import modules from keras to create a sequential model with Dense and LSTM layers
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Fig. 14. LSTM Model

Model Evaluation

To assess the performance of each of our models, we employed different evaluation metrics.
For our multiple linear regression model, we utilized metrics such as Co-efficient of determination (R-squared), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). R-squared measures how well the model fits the data, while RMSE and MAE help to quantify the difference between the predicted and actual values.

LR Evaluation. LR Model Evaluation is shown in Figure 15.



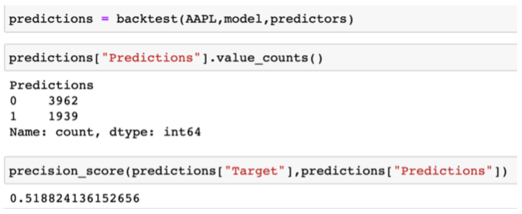
```
print("Training R-squared: ",round(metrics.r2_score(Y_train,Y_train_pred),2))
print("Training Explained Variation: ",round(metrics.explained_variance_score(Y_train,Y_train_pred),2))
print('Training MAPE:', round(get_mape(Y_train,Y_train_pred), 2))
print('Training Mean Squared Error:', round(metrics.mean_squared_error(Y_train,Y_train_pred), 2))
print("Training RMSE: ",round(np.sqrt(metrics.mean_squared_error(Y_train,Y_train_pred)),2))
print("Training MAE: ",round(metrics.mean_absolute_error(Y_train,Y_train_pred),2))
```

```
Training R-squared:  1.0
Training Explained Variation:  1.0
Training MAPE: 0.68
Training Mean Squared Error: 0.03
Training RMSE:  0.17
Training MAE:  0.07
```

Fig. 15. LR Model Evaluation

In the case of our random forest model, we first trained the model and then performed backtesting using various predictors to test the accuracy and precision score of the model. We used these results to evaluate the model's performance and refine its predictions. We performed a backtesting strategy twice, and our initial accuracy obtained in the first backtesting improved in the second backtesting strategy.

RF Evaluation. First backtesting: Accuracy 51.88% is shown in Figure 16.



```
predictions = backtest(AAPL,model,predictors)
```

```
predictions["Predictions"].value_counts()
```

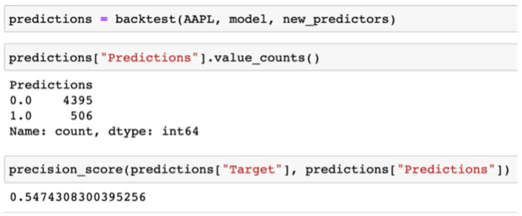
```
Predictions
0    3962
1    1939
Name: count, dtype: int64
```

```
precision_score(predictions["Target"],predictions["Predictions"])
```

```
0.518824136152656
```

Fig. 16. First Backtesting Result

RF Evaluation. Second backtesting: Accuracy (54.74%) is shown in Figure 17.



```
predictions = backtest(AAPL, model, new_predictors)
```

```
predictions["Predictions"].value_counts()
```

```
Predictions
0.0    4395
1.0     506
Name: count, dtype: int64
```

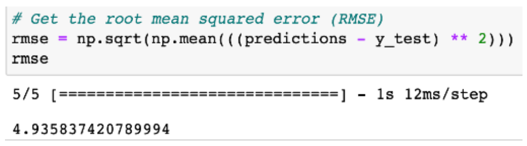
```
precision_score(predictions["Target"], predictions["Predictions"])
```

```
0.5474308300395256
```

Fig. 17. Second Backtesting Result

For our LSTM model, we calculated the Root Mean Squared Error (RMSE) as a metric for assessing its performance. This metric provides insight into how much the model's predictions deviate from the actual values. By using this evaluation metric, we were able to gauge the accuracy and reliability of our LSTM model in predicting future stock prices.

LSTM Evaluation. Ths Evaluation is shown in Figure 18.



```
# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
rmse

5/5 [==============================] - 1s 12ms/step

4.935837420789994
```

Fig. 18. LSTM Model Evaluation

In addition, we created a visual representation of our analysis by plotting the data into Train and Validation sets. We plotted the "Close Price" against the Date and included the predictions, train, and validation sets on the same plot. This allows us to observe the trend of the stock over time and the accuracy of our model's predictions.

Additional. LSTM Model Evaluation 2 is shown in Figure 19.

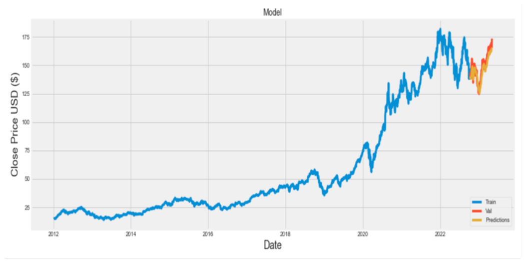


Fig. 19. LSTM Model Evaluation 2

Conclusion

After partitioning the data into training, validation, and testing sets, We trained the linear regression model using the training set. Following this, we predicted the closing price for the y-test values and plotted a graph of the actual values vs the predicted values for the test data, which provides a visual representation of the model's performance.

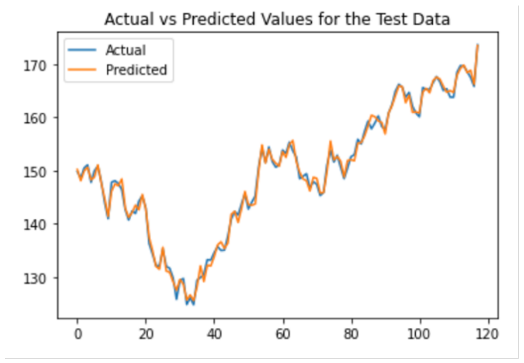
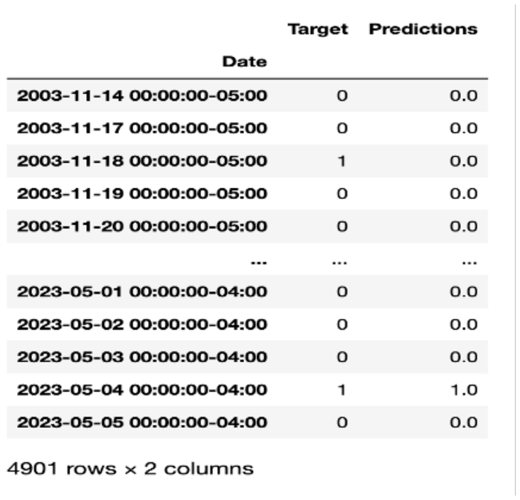


Fig. 20. Linear Regression Result

Following the second backtesting with new predictors in our random forest model, we obtained binary predictions that indicated whether the stock price would go up or not. We then compared these predictions with the actual data to assess the performance of the model.
And for the Long Short Term Memory Model, we generated a plot that shows the comparision between the actual closing and the predicted prices from our LSTM model.



Date	Target	Predictions
2003-11-14 00:00:00-05:00	0	0.0
2003-11-17 00:00:00-05:00	0	0.0
2003-11-18 00:00:00-05:00	1	0.0
2003-11-19 00:00:00-05:00	0	0.0
2003-11-20 00:00:00-05:00	0	0.0
...	...	...
2023-05-01 00:00:00-04:00	0	0.0
2023-05-02 00:00:00-04:00	0	0.0
2023-05-03 00:00:00-04:00	0	0.0
2023-05-04 00:00:00-04:00	1	1.0
2023-05-05 00:00:00-04:00	0	0.0

4901 rows × 2 columns

Fig. 21. Random Forest Result

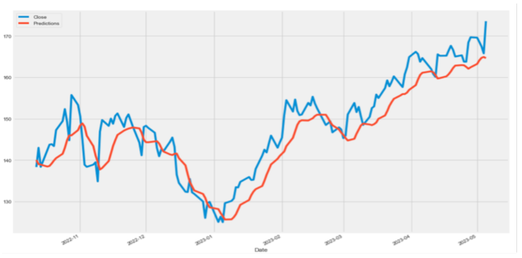


Fig. 22. LSTM Model Result

Future Scope

1. **Feature Engineering:** We want to consider exploring more features related to the company, such as news articles, social media sentiment, financial reports, and economic indicators to capture more information that might affect stock prices.

2. **Ensemble learning methods:** Exploring the combination of multiple models to improve the overall performance. For, example combining the results from a Random Forest model, a Multiple Linear Regression model, and an LSTM model may lead to better predictions.

3. **Real-time prediction:** Adapting the model to make real-time predictions by creating a pipeline that can take in live data and output stock price predictions. This would be beneficial for traders who need to make quick decisions based on current market conditons.