Stock Price Prediction using Machine Learning

Kazi Israrul Karim

Project Report

BRAC University

CSE422: Artificial Intelligence

Table of Contents

Introduction

In the dynamic landscape of financial markets, the ability to foresee stock price movements is a pursuit that engages both seasoned investors and cutting-edge technologies. This project embarks on a journey to forecast stock prices by leveraging the wealth of historical market data, presenting a comparative analysis of three distinct AI machine learning algorithms. The primary goal is to discern the most efficacious model capable of predicting stock prices across diverse stocks, using comprehensive historical data sets.

The core objective of this endeavor is to unravel the intricacies of predictive analytics in the context of stock markets. By delving into the performance metrics of various machine learning algorithms, we seek to uncover valuable insights that can redefine our understanding of market dynamics. Through meticulous examination and comparison, this project endeavors to illuminate the strengths and weaknesses of AI models in predicting stock prices, ultimately contributing to the advancement of predictive capabilities in financial analytics.

Dataset Description

Source:

Firstly we have obtained our dataset form:

https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction/data

Reference:

Shah, J. (2022, February 5). Netflix stock price prediction. Kaggle.

https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction/data

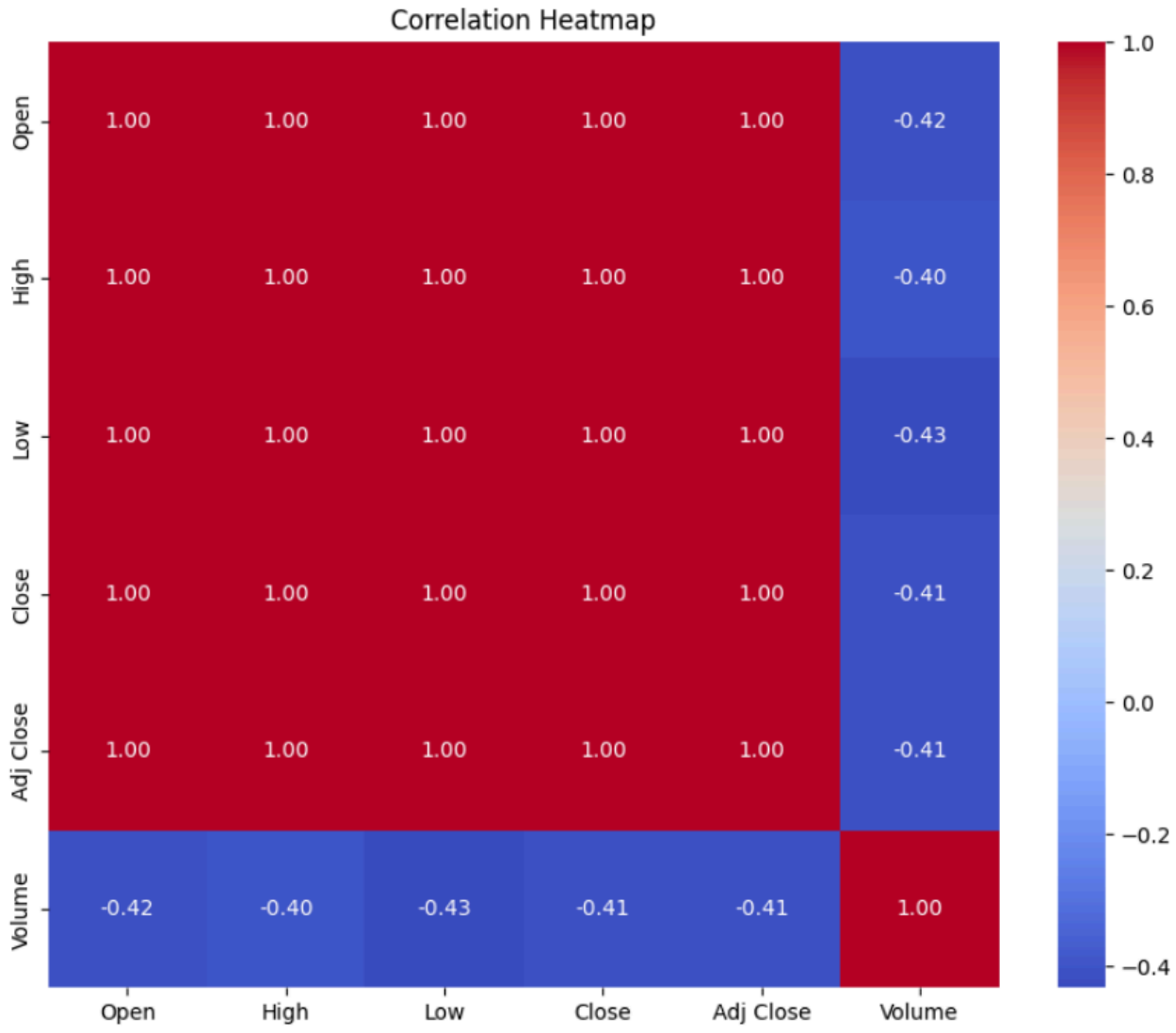
Description:

The dataset we are using is the Netflix stock market values over a span of 5 years. It comprises a total of 1009 entries and 7 columns, including Date, Open, High, Low, Close, Adj Close, and Volume. These columns represent the date of the entry, the opening price on that day, the highest and lowest prices on that day, the closing price on that day, the adjusted closing price, and the volume of stocks traded, respectively.

Out of these columns the Date is categorical and the rest are Quantitative values.

Since we want to predict the closing prices of the stock and keep it as a continuous variable, we have decided to use regression based algorithms to solve our problem

We can take a look at the correlation between each variables with the help of the heatmap below:



As our dataset represents a regression problem, our goal is to predict a continuous numeric value rather than assigning instances to discrete classes, so it is not a problem if our dataset is imbalanced.

Data Preprocessing:

Faults & Solutions:

In our dataset “Date” was the only Categorical variable that needed to be encoded in order to utilize the Date columns as an additional variable that aids us in determining our closing price. Therefore using encoding, we first converted the existing date column to pandas date and time format using pandas built in function and then used that to figure out the day_of_week (Monday = 0, Tuesday = 1,…Sunday = 6) for each of the dates. This additional data provides further insights into the relationships between pandas Date, days of the week and the closing price, which would have been nearly impossible to form with just the given Date as a string.

Furthermore, even with the additional ‘days of the week’ column we did not have enough data to predict our closing prices as precisely as we had wanted, hence why we built four indicators to find trends, momentum and potential reversal points in the financial time series data. The four indicators are:

Rolling Mean Average (Moving Average)

Represented in our database as: MA200 & MA50

Purpose: Smooth out short-term fluctuations to highlight trends.

Calculation: Average of data points within a sliding window.

Use: Identifies trend direction and potential trend reversals.

Relative Strength Index (RSI):

Represented in our database as: RSI

Purpose: Measures the speed and change of price movements.

Range: 0 to 100; values above 80 suggest overbought conditions, while values below 30 suggest oversold conditions.

Use: Identifies potential reversal points based on market momentum.

Bollinger Bands:

Represented in our database as: Lower_Band, Middle_Band & Upper_Band

Purpose: Measures volatility and identifies potential overbought or oversold conditions.

Components: Middle band (SMA), upper band (SMA + standard deviation), lower band (SMA - standard deviation).

Use: Helps identify price extremes and potential reversal points.

Moving Average Convergence Divergence (MACD):

Represented in our database as: MACD & MACD_signal

Purpose: Identifies the relationship between two moving averages of a security's price.

Components: MACD line (difference between short-term and long-term EMAs), signal line (9-day EMA of MACD).

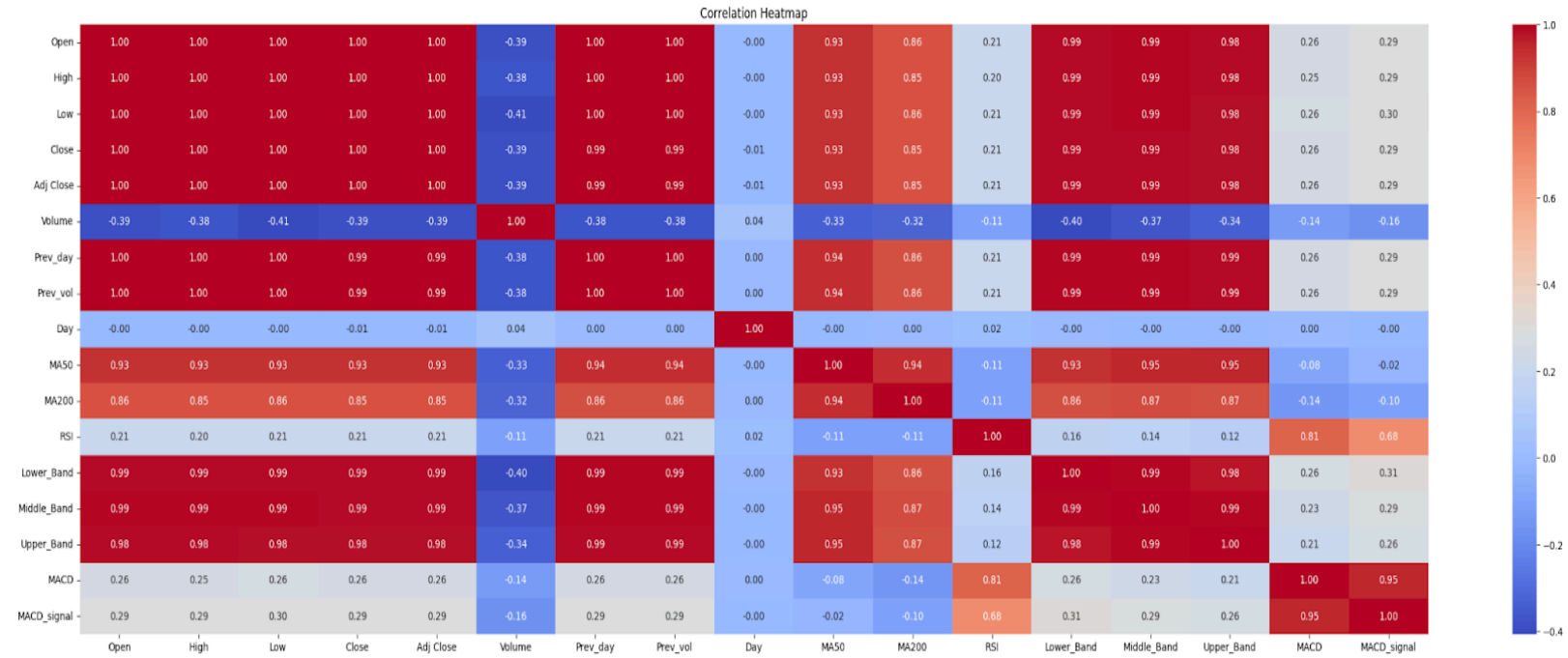
Use: Generates buy/sell signals based on crossovers and identifies changes in trend momentum.

Initially there were no NULL values within the database, however this was not the case after building and entering the four indicators into our database, those NULL values were handled by completely dropping the null rows using the built-in function .dropna(). We did not have to worry about

this approach having a huge impact on our predictions because the majority of these null values were present sequentially at the beginning of our database, therefore dropping them had little to no effect on our database especially because stock is time series based data and the dropped rows had little to no impact on the continuity of our database.

Feature Selection

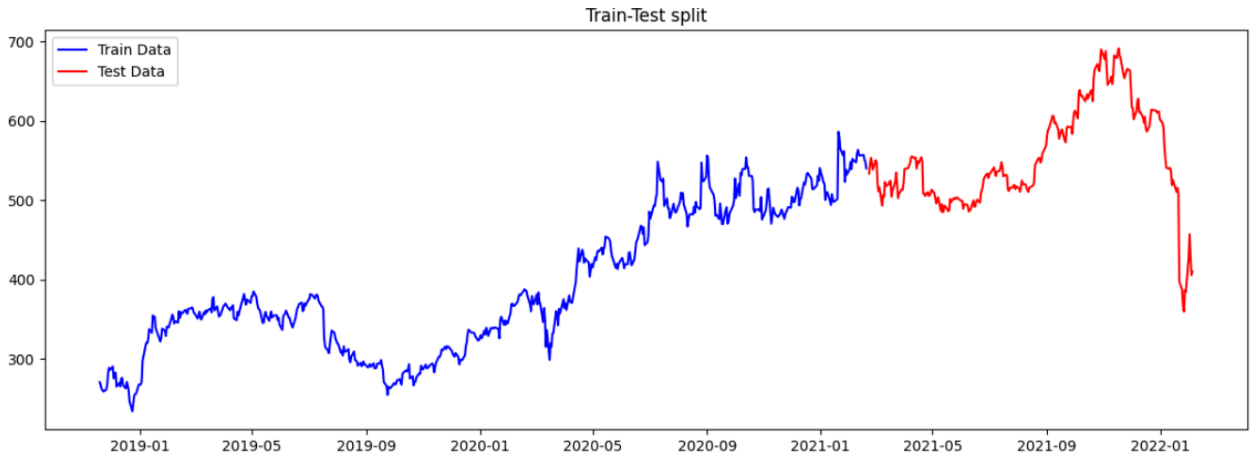
For feature selection we first created a correlation matrix. In the correlation matrix the features which had a correlation between 0.8 to 1 were dropped.



And most importantly, we made sure to drop ‘Adj close’ and ‘close’ when selecting our independent features to fit into our model. This is because “Close” serves as our target/dependent variable and using it as a feature to predict “Close” itself is counterintuitive. Essentially, predicting the current day’s closing price based on the information of the same day’s closing price would not provide meaningful insights especially in the real world . Instead we introduced a separate column “Prev_day” which is the previous day's closing price, which uses the previous day’s close to forecast the future closing price.

Dataset splitting and Scaling

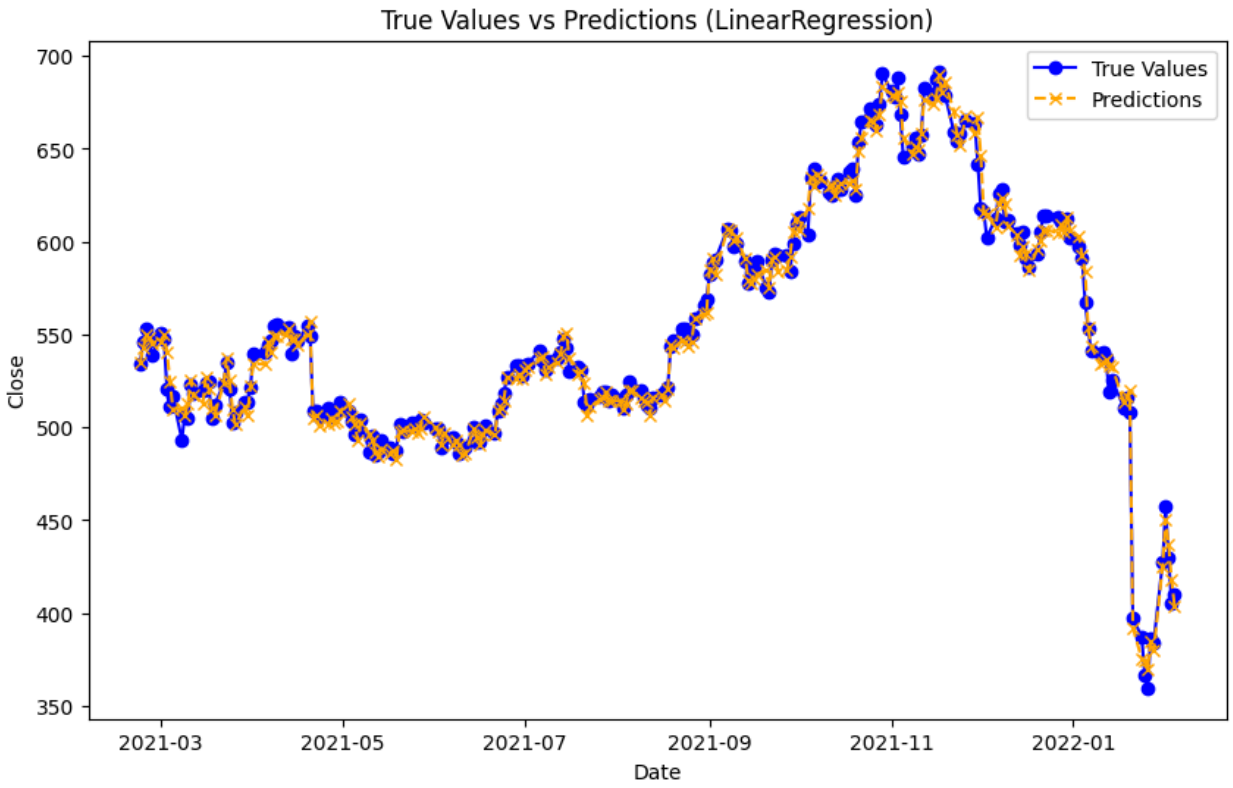
We split the data using Sklearn train_test_split function and set the shuffle to False, as this is a time sensitive dataset we would not want a random split, and used 70% of the data for training and the rest for testing. Visual representation of the training data and testing data split below:



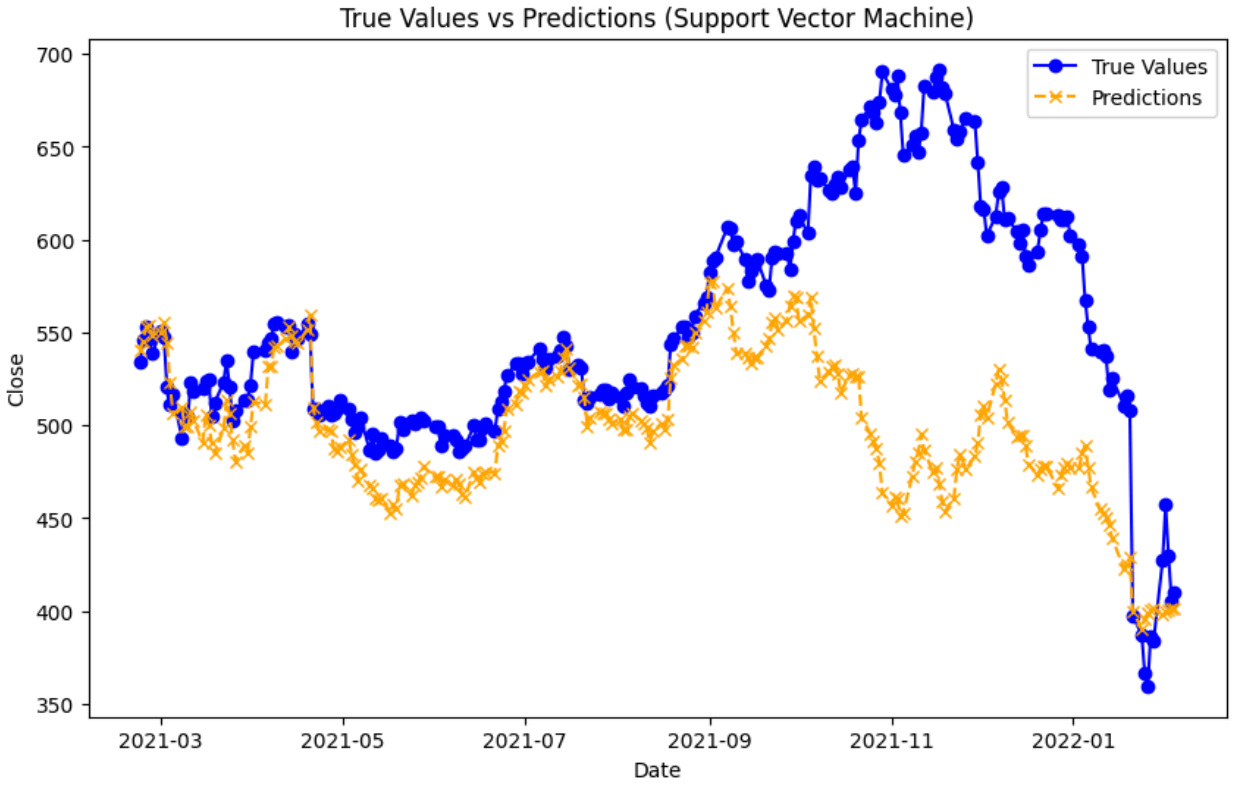
Then we used a standard scaler to scale our X_train and X_test values. We applied both fit and transform to x_train and only transform the x_test values.

Model selection/Comparison analysis

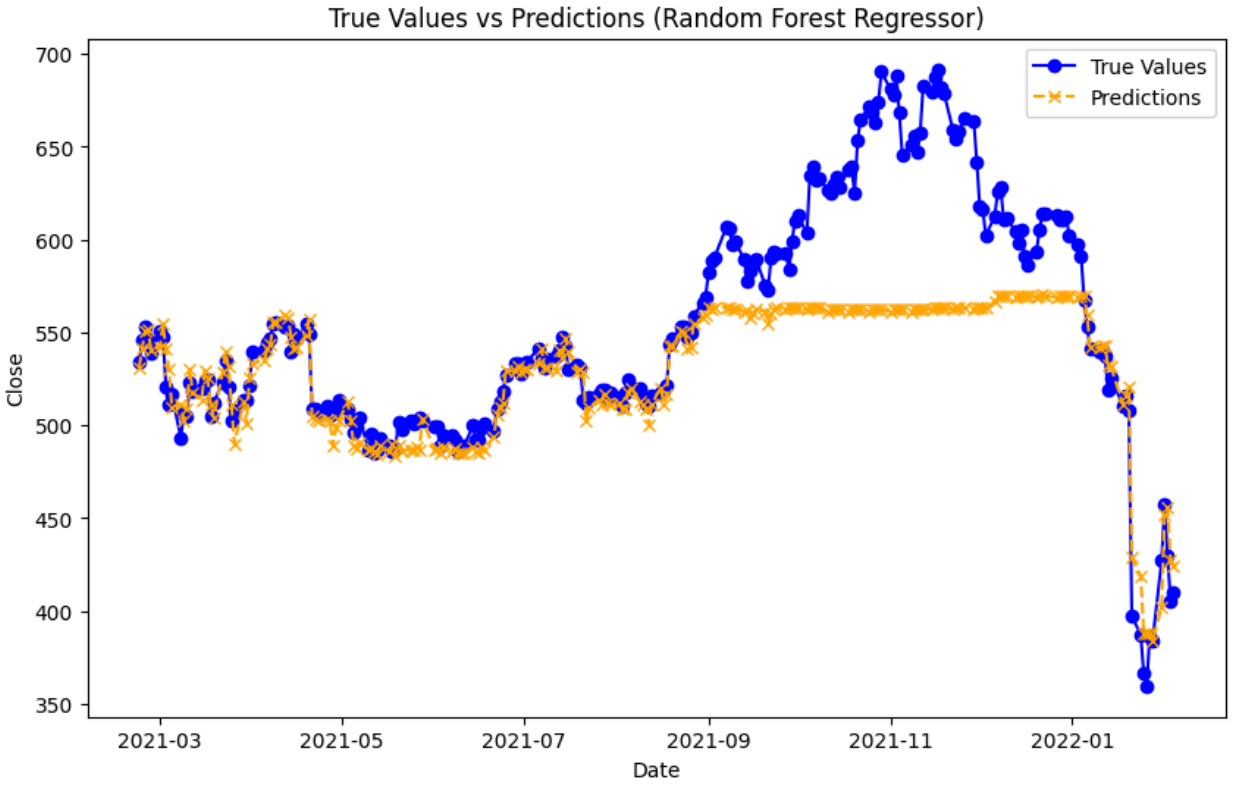
Firstly, we used the Linear Regression model, where using gradient descent we try to find the line of best fit. Below we have attached the graphical analysis of result we predicted vs the actual results:



Secondly, we used Support Vector Regression to find out the predictions. Support vector regression works by creating a hyperplane to separate the values by keeping the gap as far as possible. For kernel function we used Radial Basis Function, because our data was nonlinear and complex. We hypertuned the C parameter to a large value to fit the data more. Below we have attached the graphical analysis of result we predicted vs the actual results:



Thirdly, we used Random forest regression. We created around 100 trees each time, with a maximum depth of 25 and a feature split of 25. Below we have attached the graphical analysis of result we predicted vs the actual results:



Comparison Analysis

As our dataset represents a regression model so instead of using a confusion matrix, we used three commonly used metrics types to evaluate the performance of regression models.

MAE (Mean Absolute Error):

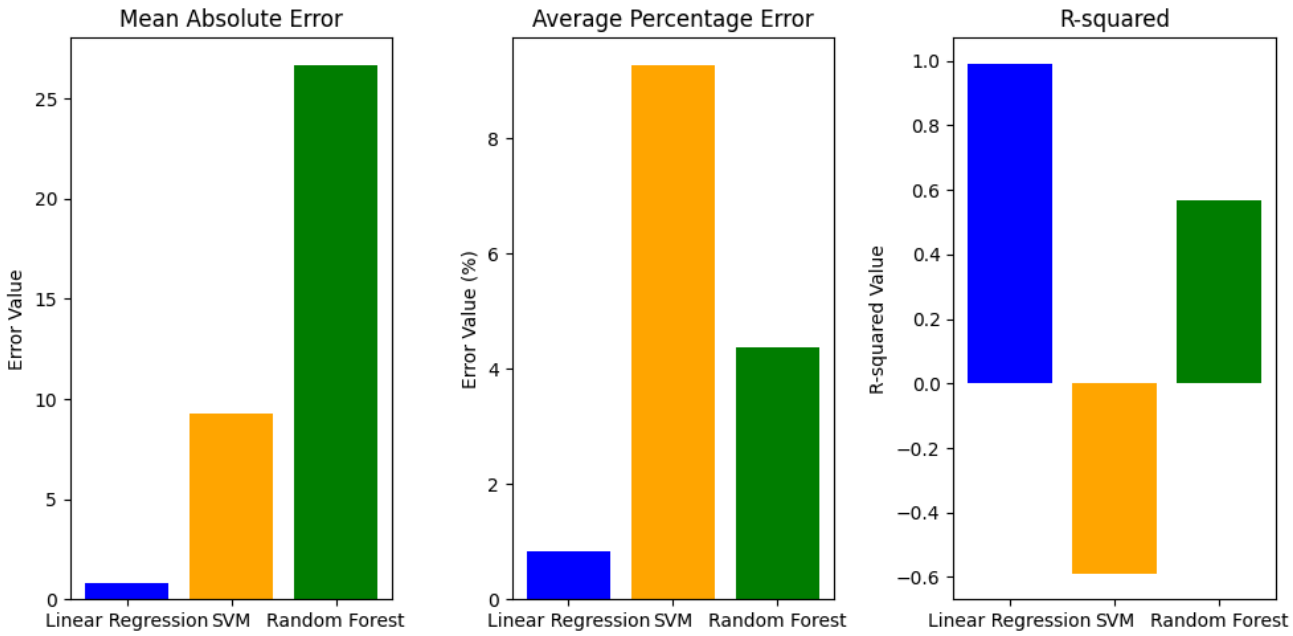
The Mean Absolute Error is the average of the absolute differences between the predicted and actual values. It provides a measure of the average magnitude of errors without considering their direction.

Average Percentage Error:

The Average Percentage Error is the average of the differences between the predicted and actual values. It provides a measure of the average errors while also considering their direction.

R^2 (R-squared):

R-squared represents the percentage of the response variable's variability that is captured by the model. A higher R-squared value indicates a better fit



Conclusion

After apply the models respectively, we can summarize the results below

Linear Regression Model
1.Mean Absolute Error using LR: 4.55 2.Average Percentage Error using Linear Regression: 0.83% 3.R-squared (LR): 0.99
Support Vector Machine
1.Mean Absolute Error (SVM): 279.4 2.Average Percentage Error using SVM: 45.92% 3.R-squared (SVM): -43.6
Random Forest Regression
1.Mean Absolute Error using Random Forest: 26.68 2.Average Percentage Error using Random Forest Regressor: 4.37% 3.R-squared (RFR): 0.56

Verdict:

From the results, we can conclude Linear Regression performs the best for our given dataset and can be used for predicting the stock prices in the future