STOCK PRICE PREDICTION USING MACHINE LEARNING [RANDOM FOREST REGRESSION MODEL]

Ghanashyam Vagale[1], Matur Rohith Kumar[2], Bhanuprakash Darbha[3], Durga Shankar Dalayi[4]

---***---

Abstract - ***The process of stock price prediction has gained significant attention in recent years due to the potential benefits it can offer to investors. This paper discusses the use of machine learning in stock price prediction by leveraging historical data to identify trends and make predictions. The application of machine learning can automate the trading process by providing insights and predictions based on statistical models. By collecting and analyzing large amounts of structured and unstructured data, suitable algorithms can be applied to identify patterns and make informed decisions. However, the volatile nature of the financial stock market poses a significant challenge in accurately predicting stock prices. Factors such as current trends, politics, and the economy can have a profound impact on stock prices, making it difficult to decide when to buy, sell, or hold. Despite these risks, machine learning can help reduce them by providing valuable insights to investors.***

Key Words: **Stock, Price, Prediction, Machine Learning, Random Forest, Regression, Artificial Intelligence, future, market.**

1. INTRODUCTION

The act of predicting stock prices based on past data is known as stock price prediction. To identify trends and comprehend the current market, we employed machine learning on previous data. Through the use of statistical models to generate predictions and draw inferences, machine learning automates the trading process. Both structured and unstructured data can be gathered and tested by machine learning. It can use the new data to apply appropriate algorithms, transform, look for trends, and make judgements. Because of the nature of the financial stock market, which involves current trends, politics, and the economy, it is difficult to predict the value of stocks with a high degree of accuracy. They have a significant impact on prices by making it difficult to decide whether to purchase, sell, or hold the stock. Risks must therefore be managed due to the fact that they cannot be eliminated.

This study demonstrates the numerous approaches used to incorporate machine learning into stock forecasting for the NSE nifty 50 index. It was built by us using Python and open-source libraries. We used pre-processing techniques to make the stock data relevant after obtaining it from Yahoo Finance. Additionally, a tuning procedure to validate the model for building, fitting, and training for prediction is used along with randomised grid search cross-validation. Following prediction, error analysis is essential for evaluating the model's effectiveness and the precision of the anticipated values.

Prediction is performed using the random forest regression model. This will forecast the low and high prices for the forthcoming trading days, along with the NSE nifty 50 index's predicted prices for the following month. Based on the expected values, decisions regarding the purchase, sale, or holding of a stock can be made. The gathering, processing, and creation of the trading algorithm for prediction are the main goals of this study.

2. FLOWCHART

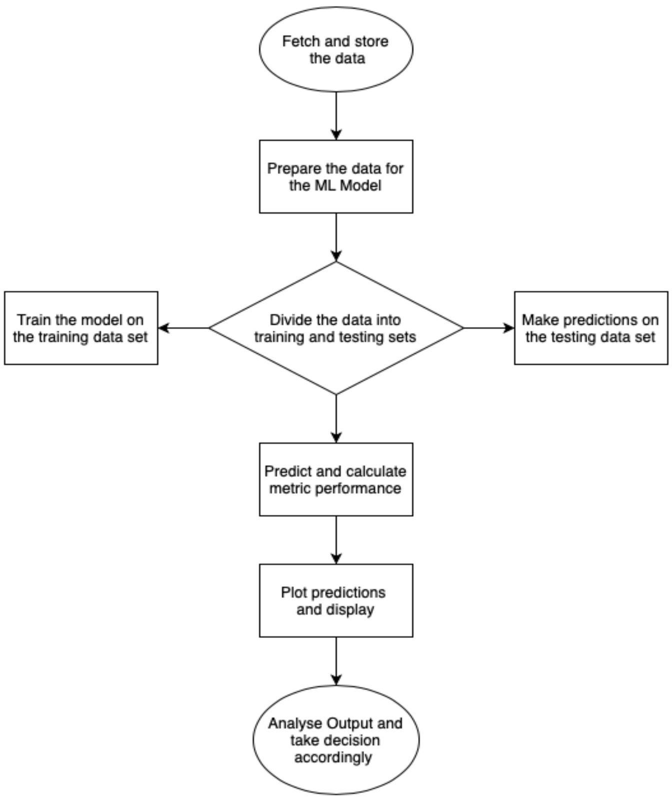

Fig -1: Flowchart of the Algorithm

3. IMPLEMENTATION

3.1 Import libraries:

The following libraries are used:

Pandas — a Python module for data analysis that loads the data file as a pandas data frame.

Matplotlib— a python module for plotting graphs.

Scikit-learn — an open-source python module used in data analysis that supports machine learning models, pre-processing, model evaluation, and training utilities. It also acts as a sub-module for train_test_split, RandomForestRegressor, StandardScaler, RandomizedSearchCV, and metrics.

Numpy— a python module that works with arrays.

Yfinance — a python open-source module used to access financial data.

```
[ ] import yfinance as yf
    import datetime as dt
    import pandas as pd
    import numpy as np
    from numpy import arange
    import matplotlib.pyplot as plt
    from pandas import read_csv
    from sklearn import metrics
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import RandomizedSearchCV
```

Fig -2: Importing Libraries

3.2 Import Dataset:

The historical data of the market is the information required for this study. For each trading day, it includes the date, prices, highest and lowest price, and amount of trades. These numbers are used by traders to gauge a stock's volatility.

```
sp500_data = yf.download("^NSEI", start="2021-01-01", end="2023-04-01")
[*********************100%***********************]  1 of 1 completed
```

```
[ ] sp500_df = pd.DataFrame(sp500_data)
    sp500_df.to_csv("sp500_data.csv")
```

Fig -3: Importing Dataset from Yahoo Finance

A Python script is used to obtain the data. The data is obtained using yfinance. It will retrieve NSEI stock data for the period of January 1, 2021, to April 1, 2023. In a data frame, the downloaded stock data is loaded before being transformed into a CSV file. so that we can easily feed it into the algorithm after storing it locally. The data set is saved as sp500_data.csv.

3.3 Visualize the Data:

```
[ ] read_df = pd.read_csv("sp500_data.csv")
    read_df.set_index("Date", inplace=True)
    read_df['Adj Close'].plot()
    plt.ylabel("Adjusted Close Prices")
    plt.show()
```

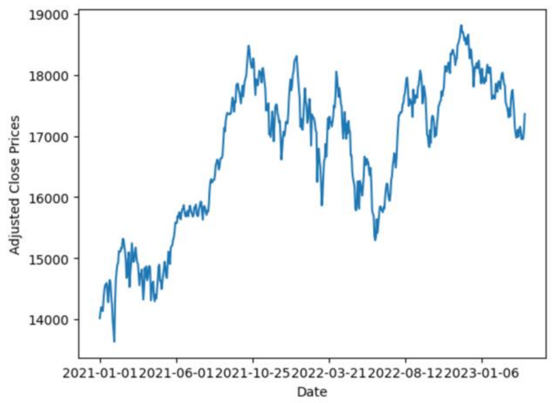

Fig -4: Plotting a line chart of the adjusted close prices over time to visualize the data.

3.3 Data pre-processing:

Preparing the data for the machine learning model involves a number of processes. Pre-processing involves transforming the raw data's format so that the model can use it and work with it. The purpose of this process is to produce a dataset that the model and algorithm can use. A dataset may have missing values, redundant and pointless information, or noisy data. Data cleaning is a type of pre-processing that involves updating the index and eliminating values that are missing or incorrect. Additionally, feature selection, hyperparameter tuning, and data standardisation is also done.

3.3.1 Read the file and set the date as the index:

```
[ ] df = pd.read_csv("sp500_data.csv")
    df.set_index("Date", inplace=True)
    df.dropna(inplace=True)
```

Fig -5: Reading the file and setting index

3.3.2 Feature selection:

The x and y characteristics are chosen at this point in order to create the model's data set. The training and testing data sets each have X and Y features defined.

The dataset's columns are called features. One of the fundamental ideas in machine learning applications, feature selection greatly affects the performance of the model. It won't be required to use every column in feature selection. These chosen features have a bearing and contribute to the outcome of the prediction. The test set performs worse overall because of unnecessary features.

Discovering the most important elements of features is one approach of choosing futures. Feature selector and feature importance modules are available in Sklearn and can be used. Each feature in the data is assigned a score using the feature significance module. The most pertinent features are those with the highest scores, and reliable output variables are always present. Using feature selection can increase accuracy, decrease overfitting, shorten training times, and enhance data visualisation. The likelihood of overfitting increases with the number of features.

Values for the open, high, low, close, and adj close columns are stored in the variable x. Y is where the adj-close column values are stored. Because they won't be required, the other columns, including the one for volume, weren't chosen for the procedure. Five features are utilized.

```
[ ]  x = df.iloc[:, 0:5].values
     y = df.iloc[:, 4].values
```

Fig -6: Selecting features

3.3.3 Divide into train and test datasets:

Before modelling, the dataset must be divided into a training and testing dataset.

A subset of the dataset used to create and fit prediction models is called the "training set." Building a training dataset script produces a training set by generating the features of the training set using the input options and the raw stock price data. The model is trained using the data. The model runs on the train set and gains knowledge from the data.

A testing set is a subset of the dataset used to gauge how well a model will perform in the future. It is a useful benchmark for assessing the model. The trained model is tested using the testing set with regard to the predicted dataset. This subset of the set has not been viewed by the model. It serves as an evaluation tool.

```
[ ] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.26,  random_state=0)
```

Fig -7: Dividing the dataset into training and testing sets

3.3.4 Scaling the features:

We refer to this as data standardisation. The standard scaler function in Sklearn is used to standardise the dataset. Standardisation has been proven to speed up training and increase the model's numerical stability.

```
[ ]  scale = StandardScaler()
     x_train = scale.fit_transform(x_train)
     x_test = scale.transform(x_test)
```

Fig -8: Scaling the train and test sets

Using the standard scaler, we are scaling the x_train and x_test.

3.4 Apply model and predict:

The model may use the dataset now. Selecting a value for the random state is the initial step, and then the tree is constructed using the number of random states. By randomly selecting subsets of the characteristics and using these subsets to create smaller trees, random forest eliminates overfitting. The training of the data is necessary to construct the random forest. The parameters from the hyperparameter tuning are also used here.

```
model = RandomForestRegressor(n_estimators=500, random_state=42,
min_samples_split=2, min_samples_leaf=1, max_depth=10, bootstrap=True)
model.fit(x_train, y_train)
predict = model.predict(x_test)
print(predict)
print(predict.shape)
```

```
[16576.60171875 15787.98517012 16242.40283008 17878.64545312
 18485.95060156 15688.15098278 14937.26278711 16692.77229687
 15695.51912679 18335.98289063 16640.31414453 17117.32357578
 17893.92367187 16040.26128516 14495.54954492 17247.09105508
 18277.71296914 17653.94935156 14859.45656445 17512.93366016
 17822.85985286 17152.83970469 17231.97636589 17233.20161979
 17352.71079687 17858.16442578 15741.70980863 15684.73697852
 17531.30260053 16215.5139082  15673.18345898 15097.80236523
 17762.6686019  17803.37494922 15288.26098828 17236.28309375
 16329.85924414 18052.24820218 17942.23618099 14387.45968164
 15860.34852344 17202.37213281 17161.82292969 14060.04159961
 17106.30331302 17201.22533984 17853.68992188 15559.0603125
 14951.95543164 17586.3385543  16949.77941406 15941.16767383
 17169.49456641 17606.68992904 17171.63051172 18122.59017187
 17951.62074349 17516.7895     18077.36880378 17341.74558555
 17311.28598398 15902.83012695 15751.02861328 17324.15692181
 17356.64633984 14930.8874375  15801.42692766 15066.97867969
 17886.40607389 17370.10251562 14638.92469049 14908.57050586
 15832.96036751 16396.57664844 18115.11841658 14337.13588672
 17625.1779332  17855.8336875  15854.12851172 17218.9263112
 15768.7799984  14911.48147852 17925.03396094 15460.21474414
 17853.20958301 18436.61702344 17929.75096875 18416.67179688
 17308.29202109 17468.87402734 16304.29701367 17523.17154961
 17657.42672982 17820.31228906 17278.02978906 17914.22289648
 15829.10502663 16498.54126563 17376.94595117 16057.75054492
 14133.89479883 18718.08760547 16581.09248047 14511.96299609
 17464.40987598 17139.14424219 16524.94800391 17734.4317959
 15728.51521973 17892.86165625 14639.28800586 16245.85658789
 18115.64136697 17351.14702734 17577.7034332  16925.69660938
 16952.24079688 17381.00286367 18000.87725677 15325.29205469
 17274.52780469 17218.59277214 17812.01782422 18125.83903145
 15814.13156976 16627.299149   14460.00775    18104.48497018
 15692.64686685 16986.48406893 18494.80717969 18003.61029349
 14883.98186523 18607.84474609 15870.32163672 16987.51579041
 17067.97676087 16796.90647656 17092.99324609 17324.80769806
 15858.02959375 17501.23202344 14556.69156836 17150.53333359
 17665.32294531 17690.17626953]
(146,)
```

Fig -9: The projected values are generated

This generates the projected values for the coming 314 trading days.

3.5 Statistical metrics and performance evaluation:

Risks are calculated using statistical metrics, which are error metrics for regression. In order to lower risks and improve model performance, model evaluation is essential.

```
print("Mean Absolute Error:", round(metrics.mean_absolute_error(y_test, predict), 4))
print("Mean Squared Error:", round(metrics.mean_squared_error(y_test, predict), 4))
print("Root Mean Squared Error:", round(np.sqrt(metrics.mean_squared_error(y_test, predict)), 4))
print("(R^2) Score:", round(metrics.r2_score(y_test, predict), 4))
print(f'Train Score : {model.score(x_train, y_train) * 100:.2f}% and Test Score : {model.score(x_test,
y_test) * 100:.2f}% using Random Tree Regressor.')
errors = abs(predict - y_test)
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')

Mean Absolute Error: 7.997
Mean Squared Error: 206.2434
Root Mean Squared Error: 14.3612
(R^2) Score: 0.9998
Train Score : 100.00% and Test Score : 99.98% using Random Tree Regressor.
Accuracy: 99.95 %.
```

Fig -10: Performance evaluation on testing

The standard deviation of the prediction mistakes is known as root mean square error (RMSE). The residuals estimate the deviation of the data points from the regression line. The distribution of these residuals is gauged by the RMSE. It describes how the data is clustered around the line of greatest fit, to put it another way. Additionally, it is MSE's square root. The performance improves with decreasing RMSE values. Given that it measures more errors than the other metrics, it should be low. A RMSE score larger than 0.5 indicates that the model has a poor capacity to reliably forecast the data. When the RMSE score is between 0.5 and 0.3, the model will forecast data with a higher degree of accuracy.

Mean absolute error (MEA) quantifies the average size of errors in a series of predictions without taking into account their directional component. It is the average absolute difference between the predicted and the observed value, where all individual variations are given equal weight. Most significantly, it calculates the difference between the actual and projected values. Assume that the MEA value is 5. The true value is 20, whereas the predicted value is 25. However, MAE does not penalize prediction errors. If errors are to be examined, they should be the mean square error or the root mean squared error. Lower values are preferable.

The absolute value of each error is added to determine the mean squared error (MSE). The model performance is also determined by the mean squared error. Larger mistakes than those found in the MAE are clearly present in this instance. The accuracy of the forecast increases as the MSE value decreases.

In machine learning, performance evaluation is essential for understanding how well the prediction and model are performing. R-squared and accuracy were employed in this study to assess the model. If a model has to be improved, it will be determined by the output value of the model evaluation. To test an alternative algorithm, fine-tune the parameters, add new data, or use feature engineering, among other options.

R squared is a measure of how well a model fits a certain dataset. It shows how closely the plotted expected and actual values match the regression line. The highest number is 1.0. So, the better the model fits the data, the higher the values. When the r-squared values fall between 0.6 and 1.0, the regression line adequately matches the data, and the model performs well. Values over 65% are regarded as favorable.

3.5 Statistical metrics and performance evaluation:

With the expected values for the following year, month, and five days, we generated data frames. A year of trading has 252 days, a month has 21, and a week has 5 trading days. From the expected 341 trade days that actually occurred, we took the necessary future days. Dates and prices are converted to CSV files for these subsequent days.

```
[ ] predictions = pd.DataFrame({"Predictions": predict},
    index=pd.date_range(start=df.index[-1], periods=len(predict), freq="D"))
    predictions.to_csv("Predicted-price-data.csv")
    #colllects future days from predicted values
    oneyear_df = pd.DataFrame(predictions[:252])
    oneyear_df.to_csv("one-year-predictions.csv")
    onemonth_df = pd.DataFrame(predictions[:21])
    onemonth_df.to_csv("one-month-predictions.csv")
    fivedays_df = pd.DataFrame(predictions[:5])
    fivedays_df.to_csv("five-days-predictions.csv")
```

Fig -11: Generating data frames and csv file containing the predictions

Investors look to profit by selling at the highest price, buying at the lowest price, and holding price if neither takes place in order to determine the buy, sell, and hold prices. The selling price is therefore the highest price in this situation, whereas the buy price is the minimum.

3. RESULTS

One month prediction result:

```
onemonth_df_pred = pd.read_csv("one-month-predictions.csv")
onemonth_df_pred.set_index("Unnamed: 0", inplace=True)
buy_price = min(onemonth_df_pred["Predictions"])
sell_price = max(onemonth_df_pred["Predictions"])
onemonth_buy = onemonth_df_pred.loc[onemonth_df_pred["Predictions"] == buy_price]
onemonth_sell = onemonth_df_pred.loc[onemonth_df_pred["Predictions"] == sell_price]
print("Buy price and date")
print(onemonth_buy)
print("Sell price and date")
print(onemonth_sell)
onemonth_df_pred["Predictions"].plot(figsize=(10, 5), title="Forecast for the next
1 month", color="blue")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.show()
```

Fig -12: Code to displaying the highest and lowest prices in the upcoming month along with a graph showing the predicted values

```
Buy price and date
             Predictions
Unnamed: 0
2023-04-14  14495.549545
Sell price and date
             Predictions
Unnamed: 0
2023-04-04  18485.950602
```

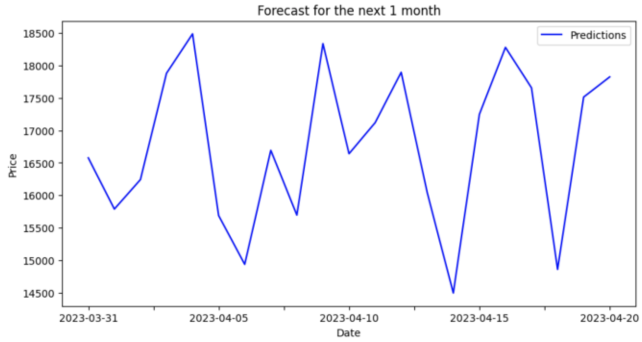

Fig -13: The highest and lowest prices in the upcoming month and their respective dates are displayed along with a graph showing the predicted values.

```
[17] print(onemonth_df_pred)

     Unnamed: 0   Predictions
0    2023-03-31  16576.601719
1    2023-04-01  15787.985170
2    2023-04-02  16242.402830
3    2023-04-03  17878.645453
4    2023-04-04  18485.950602
5    2023-04-05  15688.150983
6    2023-04-06  14937.262787
7    2023-04-07  16692.772297
8    2023-04-08  15695.519127
9    2023-04-09  18335.982891
10   2023-04-10  16640.314145
11   2023-04-11  17117.323576
12   2023-04-12  17893.923672
13   2023-04-13  16040.261285
14   2023-04-14  14495.549545
15   2023-04-15  17247.091055
16   2023-04-16  18277.712969
17   2023-04-17  17653.949352
18   2023-04-18  14859.456564
19   2023-04-19  17512.933660
20   2023-04-20  17822.859853
```

Fig -14: Displaying the predicted values in the csv file in the form of a table

3. CONCLUSIONS

In order to solve this challenge, various methods can be used. From sentiment analysis, financial news stories, and expert reviews to quantitative analysis for prediction, their performance can vary. However, there are no perfect or reliable prediction techniques due to how unpredictable the stock market is. If you need to create a model rapidly, the algorithm is a fantastic option. It gives a reasonably accurate indication of how much weight your attributes are given. The majority of the time, random forest is quick, easy, and adaptable.

REFERENCES

[1] A comparative study of machine learning algorithms for stock price prediction" by Anirudh Kumar and Arnav Kumar Jain (2021): https://www.sciencedirect.com/science/article/pii/S2405452620311239

[2] Stock Price Prediction using LSTM and Machine Learning Techniques" by Xiaoyu Liu and Lei Wang (2021): https://ieeexplore.ieee.org/abstract/document/941818

[3] Stock price prediction using machine learning algorithms: A case study on the Australian stock market" by Minh Triet Tran, Hien T. Nguyen, and Thanh Duc Nguyen (2020): https://www.sciencedirect.com/science/article/pii/S095 7417420300561

[4] A Deep Learning Approach to Predict Stock Prices from Financial News" by Md. Rafiqul Islam, Muhammad Masudur Rahman, and Khandaker Tabin Hasan (2021): https://www.mdpi.com/2076-3417/11/9/3959