2023 International Conference on Computer Communication and Informatics (ICCCI) | 979-8-3503-4821-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCCI56745.2023.10128471

Stock Price Prediction using Random Forest Classifier and Backtesting

Venkata Sai P Bhamidipati [B-Tech Student],

Department of Network and Communications,
School of Computing,
SRM Institute of Science and Technology,
Kattankulathur, Tamil Nadu, India.
vb6199@srmist.edu.in

Mrs. D. Saisanthiya,
Assistant professor,

Department of Network and Communications,
School of Computing,
SRM Institute of Science and Technology,
Kattankulathur, Tamil Nadu, India.
saisantd@srmist.edu.in

Abstract - Stock price prediction is an attempt to determine the future price of company stock and(or) other categories traded on the National Stock Exchange[NSE]. A stock's future price forecast that is accurate might result in a sizable profit. The prediction of the price of a stock is a very important topic considering that the raw data is ambiguous. Though many consider this to be an ideal machine learning problem, the solution for it is far from perfect. Many people are actively researching/funding research to derive a solution for this problem.

Usually the methodologies for prediction fall into three broad categories which can (and might) overlap:

1. Core research (Manual evaluation): Core research (or) Manual evaluation is a procedure by which investors of the stock market manually calculate the returns on their invested amount by estimating the stock price based on news articles, market movement, etc.

2. Technical Research (Tracking): Technical Research (or) Tracking is a procedure in which the stock is monitored using a software application which can track the stock value according to historical data and displays relevant information for investors.

3. Technical Techniques (Machine Learning): Technical Techniques (or) Machine Learning is a procedure by which an application is enabled to become more accurate at predicting outcomes. With the use of these techniques, investors can get a better-estimated prediction.

We aim to use machine learning as a procedure and overlap it with a concept of tracking known as backtesting to provide stock price predictions.

Keywords - Stock price prediction; Machine learning; Backtesting

1. INTRODUCTION

In the field of data science, time series analysis and forecasting are dark horses. One of the most widely used Data Science approaches is Time Series forecasting which is used in a variety of commercial and industrial processes, including supply chain management, financial analysis, and production planning.

Due to the reason that the stock markets are unpredictable and non-linear, making accurate predictions of the market returns is an extremely difficult endeavor. [1]

The development of artificial intelligence (commonly known as AI) and discoveries in computing power have made the programmed techniques for prediction progressively effective at forecasting stock values. In this study, the closing prices of a business from a random industry sector was predicted using Random Forest methodologies.

To understand the size of economies, stock price prediction is currently a current and fascinating study topic in the financial and academic sectors. No substantial set of guidelines existed to gauge and forecast the size of shares on the stock exchange. There are numerous evolutionary technologies, including technical, fundamental, time, statistical, and series analysis, that assist us in trying to predict the future, but none of these tools has been demonstrated to be a trustworthy and accurate tool for society when estimating stock exchange or share market scales. With the help of machine learning, we tried to conduct some creative work in this study to forecast or sense the behavioral tracking of the stock market.

In the study, we are suggesting the use of machine learning methods and information obtained through markets in India, namely the National Stock Exchange

[NSE] to anticipate shifts in stock prices. The method works with a big collection of information that has been collected through financial markets. Furthermore, the algorithm being used has an ability to control overfitting.

In order to predict the regular trajectory of stocks, several machine learning-based methods have been introduced. Outputs through estimations direct towards a logically acceptable proficiency which can be increased.

The objective is to create an intelligent model that, using machine learning techniques, analyses market data and predicts future patterns in stock price movement. Individuals who invest in stock markets may utilize the forecasted output of our algorithm to help them make decisions.

Although ML algorithms provide high ratings, they may still be improved. More data helps the system learn more effectively. Another way to fine-tune the hyperparameters for the greatest performance on the supplied data set is through hyperparameter optimization. Deep Learning systems may be used to forecast outcomes with accuracy. Future potential includes creating a machine learning web application in Python that would allow a user to enter a stock dataset and receive the most accurate results possible.

2. RELATED STUDIES

The Efficient Market Hypothesis (EMH), a fundamental tenet of finance, is violated when prediction algorithms are used to forecast future patterns in stock market prices. It claims that the stock prices as of today accurately represent all material facts. It indicates that if anyone were to benefit by studying previous stock data, the market as a whole would learn about this benefit, causing the share value to be adjusted. This is a very contentious and much-contested idea. Despite being widely accepted, numerous scholars have disproved this idea by employing algorithms that can simulate the economic system's more intricate processes [2].

Various methods, including SVM, Naive Bayesian Classifier, Linear Regression, KNN, Linear Discriminant Analysis and Neural Network, have been utilized in stock prediction. SVM has been employed the majority of the time in stock prediction studies, according to a study of the literature.

We may conclude that ensemble learning techniques have not been fully utilized in the problem of stock market prediction thanks to the research study. An approach called "ensemble learning" will be used to create our prediction model using the Random Forest algorithm. The output of a random forest, which consists of many decision trees, is the average of the unique trees' outputs.[7]

In contrast to previous methodologies, machine learning methods in this field have shown to increase efficiency by 60–86%. A number of neural network-based approaches, including Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Artificial Neural Networks (ANN), have also produced encouraging outcomes .[3,5,9,10,11]
The majority of the prior research in this field makes use of traditional techniques like linear regression [4].

The use of Random Forest (RF) for forecasting has been suggested in certain papers. An ensemble technique is RF. It often has the ability to carry out both classification and regression tasks. It works by building several decision trees during the training period, which produces the mean regression of each decision tree .[6-8]
In this study, two methods—ANN and RF—have been utilized to forecast an organization's closing price.

Through the use of recurring themes, the purely theoretical and hypothetical aspect of the stock market has been studied over the last few decades. Multiple linear regression, polynomial regression, and other machine learning methods are employed in this. [12]

Business organizations use diverse analysis techniques for predicting, and precision is the main goal in order to increase revenue. A good stock forecast will be a priceless tool for stock market organizations and will offer shareholders effective methods to their issues.[1]
Machine learning uses include consumer sentiment analysis, financial product recommendations, and more. We require the previous stock data in order to forecast stock values. By effectively anticipating future stock values and also recognising their swings in advance to decrease threats, the main objective is to limit the unpredictability of the profits.[3]

3. METHODOLOGY

Through the course of research, we have utilized the random forest classifier and the concept of backtesting in order to build a predictive model and the results of our model are logically acceptable. The algorithm has shown to be reliable in forecasting the future direction of stock price. Through the calculation of several metrics, including accuracy, precision, and specificity, the stability of our model has been assessed.

Decision trees are used for various machine learning applications, but the decision trees that are grown deeply in order to learn highly irregular patterns usually have an inclination to overfit the training sets. A slight random fluctuation in the data might cause the tree to grow in a completely contrasting way. This is due to the core nature of the decision trees, where they have very low slope and high variance.

Random Forest controls this problem by training multiple decision trees on different subspaces of the feature space at the cost of slightly increased slope. Due to this, not even one of the trees in the forest sees the entire training data, as the data is repeatedly split into partitions[2].

At the core of all ensemble learning algorithms, the most commonly used algorithm is Bagging, also known as Bootstrap Aggregating.

Bagging or Bootstrap Aggregating is as simple as the name suggests, bootstrapping is the stage where sampling of the dataset occurs, the sampled data is passed through various decision trees which produce an outcome, the produced outcomes are then aggregated to produce the result.

The method involving Bagging increases the precision and stability of learning algorithms. In addition, this lessens overfitting and variation, which are major issues when building decision trees.

For a labeled training dataset D of size n, bagging generates B new sets of size n^1 by sampling homogeneously from D.

Along with the above concept, we can test only against the last hundred days to provide predictions. In the real world, we would like to test across multiple years of data, because we want to know the way our algorithm handles various situations.

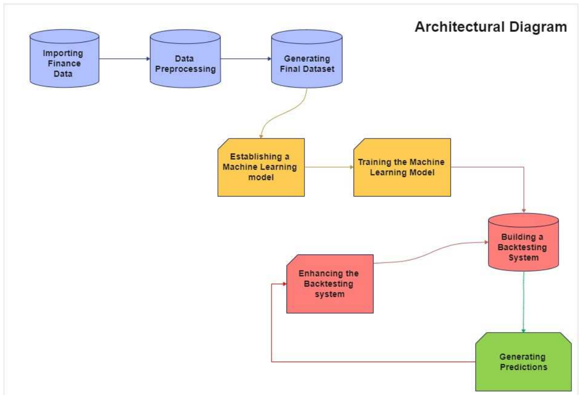
In order to do so, we are going to implement a concept called backtesting, and to establish a backtesting system, we have to create a prediction function which wraps up the whole process until prediction into a single function, and then we have to create a backtest function which takes in our data frame, model, predictors, start, step values and uses the predict function to generate predictions for the required number of days.

With the above explored concepts, we can now compile the algorithm for Random Forest Classifier with a Backtesting system as follows,

Algorithm: Random Forest Classifier with Backtesting system

```
procedure RandomForestClassifier(D)                ▷ D is the labeled training dataset
    forest = new Array()
    for do i = 0 to B
        $D_i$ = Bagging(D)                                  ▷ Bootstrap Aggregation
        $T_i$ = new DecisionTree()
        $features_i$ = RandomFeatureSelection($D_i$)
        $T_i$.train($D_i$,$features_i$)
        forest.add($T_i$)
    end for
    predictors = ["Close", "Volume", "Open", "High", "Low"]
    return forest
end procedure

combined=group(test(target), preds)

procedure Predict(train(D), test(D), predictors, forest)
    forest.fit(train(predictors),train(target))    ▷ Recursive function
    preds = forest.Predict(test(predictors))
    preds = group(preds,test[index])
    return combined
end procedure

procedure Backtest(D, forest, start, step)   ▷ start=number(years); step=number(months)
    predictions = new Array()
    for do i =start , D[0], step
        $T_{train}$ = copy(D(0,i))                          ▷ Backtesting function
        $T_{test}$ = copy(D(i,(i+step)))
        $T_{predictions}$ = Predict($T_{train}$, $T_{test}$, predictors, forest)
        predictions.append($T_{predictions}$)
    end for
    return predictions
end procedure
```

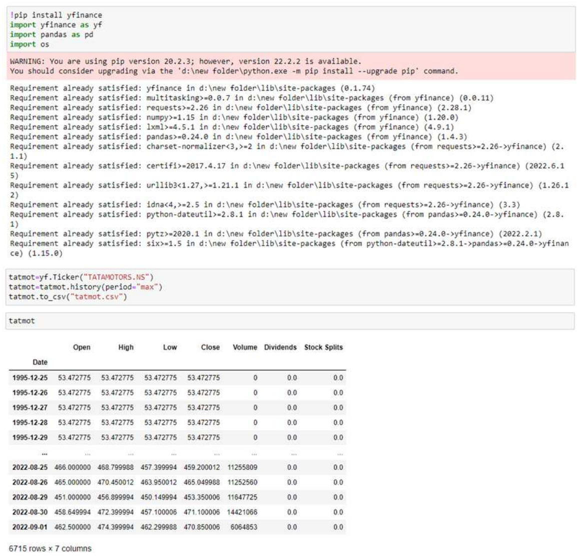
The following is an architectural diagram depicting the data flow among the modules:


Architectural Diagram
Importing Finance Data
Data Preprocessing
Generating Final Dataset
Establishing a Machine Learning model
Training the Machine Learning Model
Building a Backtesting System
Enhancing the Backtesting system
Generating Predictions

3.1 IMPORTING A STOCK'S HISTORICAL DATA

Historical data provides the data of stock prices and volumes for each stock from the listing date of the stock in the stock market until the given trading day. Historical price movements can indicate the future direction of a stock.

A stock's historical data can be imported by using third-party python libraries namely google finance and yfinance.
We have used yfinance library for this project. It is an open-source tool designed for study and teaching that takes advantage of Yahoo's publicly accessible APIs.



```
!pip install yfinance
import yfinance as yf
import pandas as pd
import os
```

```
WARNING: You are using pip version 20.2.3; however, version 22.2.2 is available.
You should consider upgrading via the 'd:\new folder\python.exe -m pip install --upgrade pip' command.
Requirement already satisfied: yfinance in d:\new folder\lib\site-packages (0.1.74)
Requirement already satisfied: multitasking>=0.0.7 in d:\new folder\lib\site-packages (from yfinance) (0.0.11)
Requirement already satisfied: requests>=2.26 in d:\new folder\lib\site-packages (from yfinance) (2.28.1)
Requirement already satisfied: numpy>=1.15 in d:\new folder\lib\site-packages (from yfinance) (1.20.0)
Requirement already satisfied: lxml>=4.5.1 in d:\new folder\lib\site-packages (from yfinance) (4.9.1)
Requirement already satisfied: pandas>=0.24.0 in d:\new folder\lib\site-packages (from yfinance) (1.4.3)
Requirement already satisfied: charset-normalizer<3,>=2 in d:\new folder\lib\site-packages (from requests>=2.26->yfinance) (2.
1.1)
Requirement already satisfied: certifi>=2017.4.17 in d:\new folder\lib\site-packages (from requests>=2.26->yfinance) (2022.6.1
5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\new folder\lib\site-packages (from requests>=2.26->yfinance) (1.26.1
2)
Requirement already satisfied: idna<4,>=2.5 in d:\new folder\lib\site-packages (from requests>=2.26->yfinance) (3.3)
Requirement already satisfied: python-dateutil>=2.8.1 in d:\new folder\lib\site-packages (from pandas>=0.24.0->yfinance) (2.8.
1)
Requirement already satisfied: pytz>=2020.1 in d:\new folder\lib\site-packages (from pandas>=0.24.0->yfinance) (2022.2.1)
Requirement already satisfied: six>=1.5 in d:\new folder\lib\site-packages (from python-dateutil>=2.8.1->pandas>=0.24.0->yfinan
ce) (1.15.0)
```

```
tatmot=yf.Ticker("TATAMOTORS.NS")
tatmot=tatmot.history(period="max")
tatmot.to_csv("tatmot.csv")
```

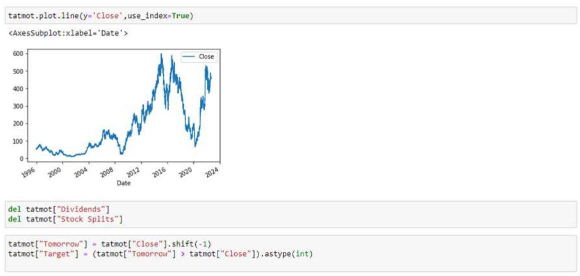
```
tatmot
```

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
1995-12-25	53.472775	53.472775	53.472775	53.472775	0	0.0	0.0
1995-12-26	53.472775	53.472775	53.472775	53.472775	0	0.0	0.0
1995-12-27	53.472775	53.472775	53.472775	53.472775	0	0.0	0.0
1995-12-28	53.472775	53.472775	53.472775	53.472775	0	0.0	0.0
1995-12-29	53.472775	53.472775	53.472775	53.472775	0	0.0	0.0
...	...	...	...	...	...	...	...
2022-08-25	466.000000	468.799988	457.399994	459.200012	11255809	0.0	0.0
2022-08-26	465.000000	470.450012	463.950012	465.049988	11252560	0.0	0.0
2022-08-29	451.000000	456.899994	450.149994	453.350006	11647725	0.0	0.0
2022-08-30	458.649994	472.399994	457.100006	471.100006	14421066	0.0	0.0
2022-09-01	462.500000	474.399994	462.299988	470.850006	6064853	0.0	0.0

6715 rows × 7 columns

3.2 DATA PREPROCESSING

Data Quality Assessment followed by Data Cleaning, Transformation and Reduction can be considered as the steps for Data Preprocessing. Data preprocessing is mentioned as the information handling where information is manipulated or dropped before it is used as an input for the model.
The market data imported for a specific period has to be modified as per requirements so that the Machine Learning Model can recognise and learn accordingly.



```
tatmot.plot.line(y='Close',use_index=True)
```

```
<AxesSubplot:xlabel='Date'>
```

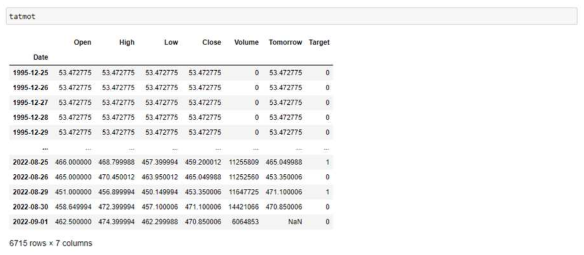
```
del tatmot["Dividends"]
del tatmot["Stock Splits"]
```

```
tatmot["Tomorrow"] = tatmot["Close"].shift(-1)
tatmot["Target"] = (tatmot["Tomorrow"] > tatmot["Close"]).astype(int)
```

3.3 GENERATING DATASET

A Dataset is a collection of various types of data stored in a digital format. Every project using machine learning needs information as its primary input.

Our ideal dataset consists of numerical data which is timestamped for every business day excluding public holidays. The preprocessed data is used to generate a dataset that will be used for the Machine Learning Model.



```
tatmot
```

	Open	High	Low	Close	Volume	Tomorrow	Target
Date							
1995-12-25	53.472775	53.472775	53.472775	53.472775	0	53.472775	0
1995-12-26	53.472775	53.472775	53.472775	53.472775	0	53.472775	0
1995-12-27	53.472775	53.472775	53.472775	53.472775	0	53.472775	0
1995-12-28	53.472775	53.472775	53.472775	53.472775	0	53.472775	0
1995-12-29	53.472775	53.472775	53.472775	53.472775	0	53.472775	0
...	...	...	...	...	...	...	...
2022-08-25	466.000000	468.799988	457.399994	459.200012	11255809	465.049988	1
2022-08-26	465.000000	470.450012	463.950012	465.049988	11252560	453.350006	0
2022-08-29	451.000000	456.899994	450.149994	453.350006	11647725	471.100006	1
2022-08-30	458.649994	472.399994	457.100006	471.100006	14421066	470.850006	0
2022-09-01	462.500000	474.399994	462.299988	470.850006	6064853	NaN	0

6715 rows × 7 columns

3.4 MACHINE LEARNING MODEL

Through the usage of machine learning (ML), a concept of artificial intelligence (AI), software programmes are created, which anticipate results better and accurately without having to be explicitly instructed to perform so. Inorder to speculate potential results, machine learning models take historical price information as input[13].

We are going to be using an ensemble learning concept, the Random Forest Classifier as the algorithm.

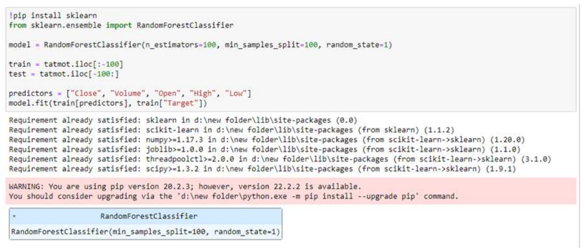
Random forest is the training method employed in this research. Technical indicators are retrieved after the time-series data has been smoothed. Increased weight is assigned to current observations when using exponential smoothing, while less weight is given to older observations.

By removing random fluctuation through past information, smoothing enables algorithms in order to quickly discover enduring cost trends in the performance of stock prices. The exponentially smoothed time series data, which have then been arranged into feature matrices, are used to create technical indicators.

Technical indicators are factors that offer perceptions of anticipated future stock price behavior. The random forest is then trained using these technical indicators.

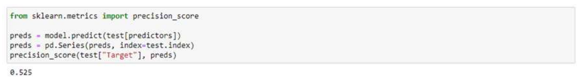
Various ML(Machine Learning) programs could employ decision trees. Nevertheless, the trees that have been developed deeply to acquire extremely random features have an inclination to overfit the training datasets.
In this study, brand-new variables are made available for every decision tree's training, which in turn affects the decisions made

at the tree's nodes. Due to its enormous magnitude, overall disturbance in stock market data is typically fairly strong and therefore can lead the trees to develop very differently than intended.



```
!pip install sklearn
from sklearn.ensemble import RandomForestClassifier

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

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

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

```
Requirement already satisfied: sklearn in d:\new folder\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in d:\new folder\lib\site-packages (from sklearn) (1.1.2)
Requirement already satisfied: numpy>=1.17.3 in d:\new folder\lib\site-packages (from scikit-learn->sklearn) (1.20.0)
Requirement already satisfied: joblib>=1.0.0 in d:\new folder\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\new folder\lib\site-packages (from scikit-learn->sklearn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in d:\new folder\lib\site-packages (from scikit-learn->sklearn) (1.9.1)

WARNING: You are using pip version 20.2.3; however, version 22.2.2 is available.
You should consider upgrading via the 'd:\new folder\python.exe -m pip install --upgrade pip' command.

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



```
from sklearn.metrics import precision_score

preds = model.predict(test[predictors])
preds = pd.Series(preds, index=test.index)
precision_score(test["Target"], preds)
```

```
0.525
```

3.5 BACKTESTING SYSTEM

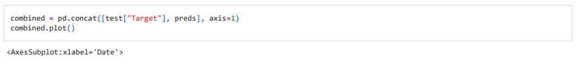
In the stock market, the process of testing a trading strategy on historical data to assess its accuracy, without actually investing real money is termed as backtesting. The main conjecture behind the concept of backtesting is that the trading strategy which performed well for the historic data is likely to perform well for the future.
Backtesting that is carried out correctly and yields encouraging outcomes gives traders more assurance to use the strategy going forward. In the scenario that a backtest reflects negative results, the investors can either improve the system or reject the strategy.

Backtesting is a crucial method for creating explainability and providing studies with a fair chance of being adopted in the real world in the field of financial analysis utilizing stock market data. This applies to the use of historical data to evaluate a model's feasibility in the past and provide users with the assurance to use it going ahead. An in-depth investigation of the trading technique using backtesting is presented in this work as an effective implementation.
AmiBroker, Zerodha Streak, Trade Station, and TradersCockpit are considered some good platforms for backtesting software.

The concept of backtesting can be split into two procedures with the help of Machine Learning, namely Prediction function and Backtesting function.

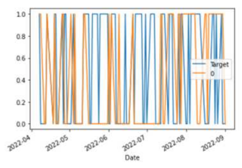
3.5.1 PREDICTION FUNCTION

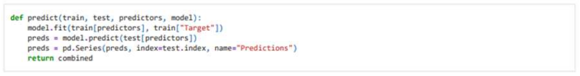
The prediction function is a recursive function, which is built up by fitting the model using the training predictors and target. The model fits with the training predictors and the target is called inside the function with test predictors, to create a recursive function which generates predictions.



```
combined = pd.concat([test["Target"], preds], axis=1)
combined.plot()
```

```
<AxesSubplot:xlabel='Date'>
```





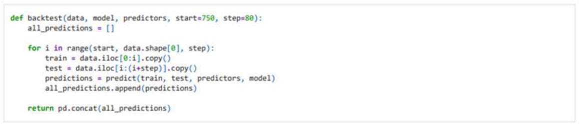
```
def predict(train, test, predictors, model):
    model.fit(train[predictors], train["Target"])
    preds = model.predict(test[predictors])
    preds = pd.Series(preds, index=test.index, name="Predictions")
    return combined
```

3.5.2 BACKTEST FUNCTION

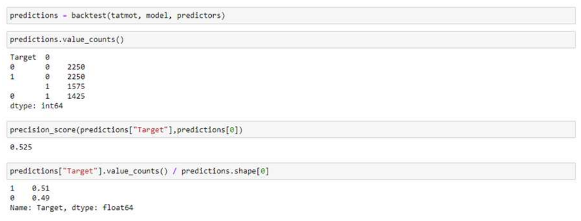
The backtesting function takes in the data frame, the model generated by the random forest classifier, the predictors, the start value and the step value. While backtesting a model, one should have the required amount of data to train the model. A few constants that we are going to be using are specified below:

- A Trading Year has 250 days.
- A Trading Quarter has 60 days.
- A Trading Month has 20 days.
- A Trading Week has 5 days.

We are going to take the values for 3 years [3 * 250 = 750 days] as input and provide predictions for the next 4 months [4 * 20 = 80 days]. These values are used in the function for *start* and *step* values.
The predictions are generated by using the predict function [3.5.1], these predictions are concatenated using the panda's python library.



```
def backtest(data, model, predictors, start=750, step=80):
    all_predictions = []

    for i in range(start, data.shape[0], step):
        train = data.iloc[0:i].copy()
        test = data.iloc[i:(i+step)].copy()
        predictions = predict(train, test, predictors, model)
        all_predictions.append(predictions)

    return pd.concat(all_predictions)
```



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

```
predictions.value_counts()
```

```
Target  0
0       0    2250
1       0    2250
        1    1575
0       1    1425
dtype: int64
```

```
precision_score(predictions["Target"],predictions[0])
```

```
0.525
```

```
predictions["Target"].value_counts() / predictions.shape[0]
```

```
1    0.51
0    0.49
Name: Target, dtype: float64
```

3.6 ENHANCING THE MODEL

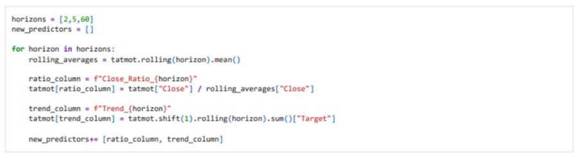
The whole model can be further enhanced by creating a variety of rolling averages.

Let us consider a human analyst trying to predict if a stock will go up the next day, knowing the stock prices 2 days back, 1 week back and 3 months back help in predicting whether the stock goes up or down.

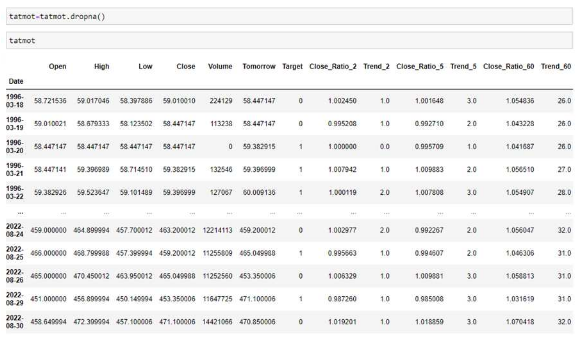
We have coined the term *horizons* for the implementation of this concept, *horizons* are used to provide the algorithm with the rolling averages, i.e. the mean close price for each horizon.

Calculating the ratio between the current day's closing price and the closing price in those periods can be useful in checking whether the stock price has gone up a significant amount, because if so, it may be due for a downturn. Similarly, if the stock price has gone down a significant amount, it may be due for an upswing.

The ratios calculated for each horizon are created and inserted into a new column. The ratios calculated can be used to calculate the number of days a stock price has gone up and add those as new predictors.



```
horizons = [2,5,60]
new_predictors = []

for horizon in horizons:
    rolling_averages = tatmot.rolling(horizon).mean()

    ratio_column = f"Close_Ratio_{horizon}"
    tatmot[ratio_column] = tatmot["Close"] / rolling_averages["Close"]

    trend_column = f"Trend_{horizon}"
    tatmot[trend_column] = tatmot.shift(1).rolling(horizon).sum()["Target"]

    new_predictors+= [ratio_column, trend_column]
```



```
tatmot=tatmot.dropna()
```

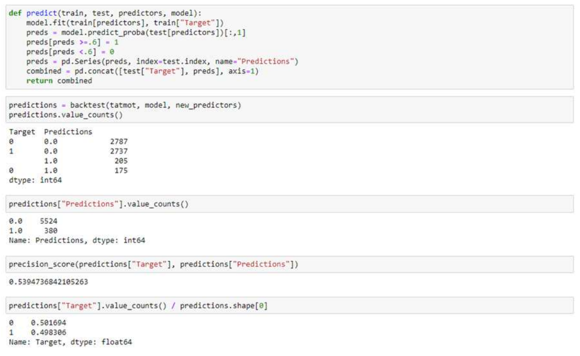
```
tatmot
```

Date	Open	High	Low	Close	Volume	Tomorrow	Target	Close_Ratio_2	Trend_2	Close_Ratio_5	Trend_5	Close_Ratio_60	Trend_60
1996-03-18	58.721536	59.017046	58.397886	59.010010	224129	58.447147	0	1.002450	1.0	1.001648	3.0	1.054836	26.0
1996-03-19	59.010021	58.679333	58.123502	58.447147	113238	58.447147	0	0.995208	1.0	0.992710	2.0	1.043228	26.0
1996-03-20	58.447147	58.447147	58.447147	58.447147	0	59.382915	1	1.000000	0.0	0.995709	1.0	1.041687	26.0
1996-03-21	58.447141	59.396989	58.714510	59.382915	132546	59.396999	1	1.007942	1.0	1.009883	2.0	1.056510	27.0
1996-03-22	59.382926	59.523647	59.101489	59.396999	127067	60.009136	1	1.000119	2.0	1.007808	3.0	1.054907	28.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-08-24	459.000000	464.899994	457.700012	463.200012	12214113	459.200012	0	1.002977	2.0	0.992267	2.0	1.056047	32.0
2022-08-25	466.000000	468.799988	457.399994	459.200012	11255809	465.049988	1	0.995663	1.0	0.994607	2.0	1.046306	31.0
2022-08-26	465.000000	470.450012	463.950012	465.049988	11252560	453.350006	0	1.006329	1.0	1.009881	3.0	1.058813	31.0
2022-08-29	451.000000	456.899994	450.149994	453.350006	11647725	471.100006	1	0.987260	1.0	0.985008	3.0	1.031619	31.0
2022-08-30	458.649994	472.399994	457.100006	471.100006	14421066	470.850006	0	1.019201	1.0	1.018859	3.0	1.070418	32.0

We can further improve the model by improving the model being generated by the Random Forest Classifier by increasing the number of estimators. We can also improve the model by enhancing the prediction function [3.5.1].

In the above function [3.5.1] we have used the predict function inside to return the prediction for the target price as 0 (for the stock price going down) or 1 (for the stock price going up). We can improve that by implementing an alternative known as *predict_proba*, which returns the probability that the stock price would go up or down.

For the new predictions, we checked the model for 60% (0.6) probability and returned the prediction for the target price as 0 or 1 based on the probability.



```
def predict(train, test, predictors, model):
    model.fit(train[predictors], train["Target"])
    preds = model.predict_proba(test[predictors])[:,1]
    preds[preds >=.6] = 1
    preds[preds <.6] = 0
    preds = pd.Series(preds, index=test.index, name="Predictions")
    combined = pd.concat([test["Target"], preds], axis=1)
    return combined
```

```
predictions = backtest(tatmot, model, new_predictors)
predictions.value_counts()
```

```
Target  Predictions
0       0.0            2787
1       0.0            2737
        1.0             205
0       1.0             175
dtype: int64
```

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

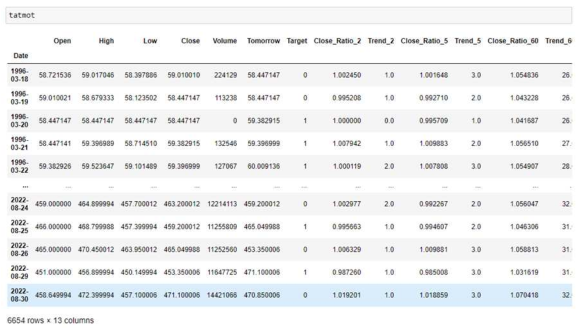
```
0.0    5524
1.0     380
Name: Predictions, dtype: int64
```

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

```
0.5394736842105263
```

```
predictions["Target"].value_counts() / predictions.shape[0]
```

```
0    0.501694
1    0.498306
Name: Target, dtype: float64
```



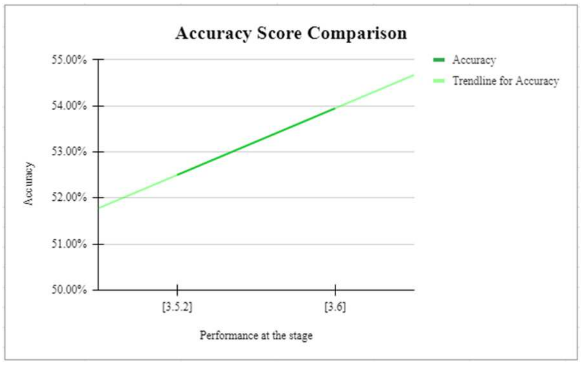
```
tatmot
```

Date	Open	High	Low	Close	Volume	Tomorrow	Target	Close_Ratio_2	Trend_2	Close_Ratio_5	Trend_5	Close_Ratio_60	Trend_6
1996-03-18	58.721536	59.017046	58.397886	59.010010	224129	58.447147	0	1.002450	1.0	1.001648	3.0	1.054836	26.
1996-03-19	59.010021	58.679333	58.123502	58.447147	113238	58.447147	0	0.995208	1.0	0.992710	2.0	1.043228	26.
1996-03-20	58.447147	58.447147	58.447147	58.447147	0	59.382915	1	1.000000	0.0	0.995709	1.0	1.041687	26.
1996-03-21	58.447141	59.396989	58.714510	59.382915	132546	59.396999	1	1.007942	1.0	1.009883	2.0	1.056510	27.
1996-03-22	59.382926	59.523647	59.101489	59.396999	127067	60.009136	1	1.000119	2.0	1.007808	3.0	1.054907	28.
...	...	...	...	...	...	...	...	...	...	...	...	...	.
2022-08-24	459.000000	464.899994	457.700012	463.200012	12214113	459.200012	0	1.002977	2.0	0.992267	2.0	1.056047	32.
2022-08-25	466.000000	468.799988	457.399994	459.200012	11255809	465.049988	1	0.995663	1.0	0.994607	2.0	1.046306	31.
2022-08-26	465.000000	470.450012	463.950012	465.049988	11252560	453.350006	0	1.006329	1.0	1.009881	3.0	1.058813	31.
2022-08-29	451.000000	456.899994	450.149994	453.350006	11647725	471.100006	1	0.987260	1.0	0.985008	3.0	1.031619	31.
2022-08-30	458.649994	472.399994	457.100006	471.100006	14421066	470.850006	0	1.019201	1.0	1.018859	3.0	1.070418	32.

6654 rows × 13 columns

4. RESULT

Through the process done, we have proved and increased the accuracy of the Random Forest Classifier with the Backtesting system by enhancing the model.
In the model[3.4], we have calculated the accuracy score and noted it to be 52.5%. After checking the dataset with the function[3.5.2], we have calculated the accuracy score and noted it to be 52.5% again, using the above information we have confirmed that the research work aligns with the preexisting model.
After enhancing the model[3.6], we have calculated the accuracy score again and noted it to be 53.947%.
With the above information, we can validate the increase in accuracy of the models, which can be shown in the graph below,



We can also estimate that with the required historical data, the exact number of estimators, and the accurate percentage we can achieve higher accuracy.

5. CONCLUSION

It is quite challenging to predict the stock market because of its nonlinear, dynamic, and complicated character. Yet, in recent times, stock forecasting has found machine learning approaches to be beneficial.
Several algorithms, including SVM, LSTM and others, have been investigated for their reliability in stock market forecasting. However, there hasn't been much use of ensemble learning techniques in this area. Our prediction model in this study, which was developed using a random forest classifier, yielded some quite outstanding outcomes. Through the calculation of several metrics, including accuracy the stability of our approach has been assessed.

REFERENCES

1. Ahmad, Mohamed. (2018). Stock price prediction.

2. Khaidem, Luckyson & Saha, Snehanshu & Basak, Suryoday & Kar, Saibal & Dey, Sudeepa. (2016). Predicting the direction of stock market prices using random forest.

3. Li, Lei, Yabin Wu, Yihang Ou, Qi Li, Yanquan Zhou, and Daoxin Chen. (2017) “Research on machine learning algorithms and feature extraction for time series.” IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC): 1-5.

4. Seber, George AF and Lee, Alan J. (2012) “Linear regression analysis.” John Wiley & Sons 329

5. J. Kumari, V. Sharma and S. Chauhan, "Prediction of Stock Price using Machine Learning Techniques: A Survey," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2021, pp. 281-284, doi: 10.1109/ICAC3N53548.2021.9725685.

6. Kumar, Manish, and M. Thenmozhi. (2006) “Forecasting stock index movement: A comparison of support vector machines and random forest” In Indian institute of capital markets 9th capital markets conference paper

7. R. Zi, Y. Jun, Y. Yicheng, M. Fuxiang and L. Rongbin, "Stock price prediction based on optimized random forest model," in 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML), Hangzhou, China, 2022 pp. 777-783. doi: 10.1109/CACML55074.2022.00134

8. A. Murakami and Y. Shirota, "Time Series Analysis of Global Automakers Stock Price Clustering," in 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI), Kitakyushu, Japan, 2020 pp. 588-591. doi: 10.1109/IIAI-AAI50415.2020.00122

9. Y. Wei and V. Chaudhary, "The Directionality Function Defect of Performance Evaluation Method in Regression Neural Network for Stock Price Prediction," 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA),

2020, pp. 769-770, doi: 10.1109/DSAA49011.2020.00108.

10. L. Sayavong, Z. Wu and S. Chalita, "Research on Stock Price Prediction Method Based on Convolutional Neural Network," 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), 2019, pp. 173-176, doi: 10.1109/ICVRIS.2019.00050.

11. G. Bathla, "Stock Price prediction using LSTM and SVR," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2020, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.

12. L. Mathanprasad and M. Gunasekaran, "Analysing the Trend of Stock Marketand Evaluate the performance of Market Prediction using Machine Learning Approach," 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), 2022, pp. 1-9, doi: 10.1109/ACCAI53970.2022.9752616.

13. Raju Narayana Swamy, “Strong AI to Super-intelligence: How is AI placed vis-à-vis Intellectual Property Rights”, International Journal Of Computer Communication And Informatics, volume- 3, issue- 2, 1-11, 2021