FA 691 DEEP LEARNING IN FINANCE

Written By-

Jayesh Kartik
CWID-20012210
Master's in Financial Analytics
Stevens Institute of Technology, Hoboken, New Jersey
Under the guidance-Dr. Zach Feinstein
Date-1st March'2023

Contents

Abstract

This paper is written to predict stock market prices. The stock prices in the future are the most crucial thing to decide its moving trend. In this paper, I have used various data analysis techniques and machine and deep learning algorithms to accurately predict the stock market movement. Based on the predicted values of the stock, we can compare the accuracy of the model and choose a model to suitably predict future stock prices. Based on the results of this final model, I have finally implemented some trading decisions which can be used by an investor. The paper also gives visuals of the stock price movement which is easily understandable.

Motivation

This paper presents an analysis of stock price movement using historical data of Yahoo Stock from 2015-2020. I chose this dataset because the whole market changed in 2020 because of the Covid-19 pandemic. Investors were in a dilemma of putting money into the market or taking the money out of the market. The paper analyzed the trends in their prices using statistical and machine learning models. I divided the dataset into two parts-80% is the training dataset and 20 percent is the testing dataset. I evaluated the performance of different models with R^2 value and identified the most effective models for predicting stock price movement. Our results show that these techniques can be highly effective in predicting stock price movement, with some models achieving accuracy rates of over 90%.

I have chosen yahoo stock because:
Availability of data: Yahoo stock has been publicly traded for several years, providing us with a wealth of historical data that can be used to train our models and make predictions about future stock prices.
Market volatility: Yahoo stock has experienced significant market volatility in recent years, with fluctuations in stock prices. This volatility makes Yahoo stock an interesting and challenging subject for analysis, providing opportunities to test the effectiveness of my model.
Market capitalization: Yahoo stock has a relatively high market capitalization, making it a widely traded stock with significant investor interest. This high market capitalization means that our analysis of Yahoo stock has the potential to provide valuable insights for investors and traders alike.

Introduction

STOCK MARKET PREDICTION-

The stock market is an ever-changing landscape that can be difficult to navigate for investors and traders alike. With the constant changes in the market, predicting future stock prices can seem like an impossible task. But predicting future stock prices is necessary because it-

Financial gain: Every person who has invested in the stock market has an intention of getting maximum profit from it. If they can predict future stock prices, they can maximize their profit and minimize risks. The predicted value can give them an idea of the movement of the stock and hence they can make an informed decision.

Risk management: Accurately predicting stock market trends can also let investors manage their risk at the right time. Investing money into a particular stock and taking money out of that stock is one of the most crucial decisions to make. The right decision at right time is only possible after having a lot of experience and predicting the stock price can help investors take the right decision at the most important time.

Business strategy: Now, if we look from a different point of view, companies may also use stock market predictions to inform their business strategies, such as deciding when to enter or exit markets or industries. This can help them segment their market in a structured manner and not taking decisions intuitively but with solid figures.

About the Dataset-

The dataset has been taken from Kaggle. The link to the dataset is attached here-

https://www.kaggle.com/code/georgesaavedra/time-series-forecasting/data

I have put it in a google drive folder. The google drive link is attached here-

https://drive.google.com/file/d/19FPjDiR_t1MN970AklaD2gm2JzyAs8dr/view?usp=share_link

The dataset has six columns-

Date: It refers to a specific day or period on which the market data is recorded or reported.

High: The highest price at which a security or index trades during a day.

Low: The lowest price at which a security or index trades during a day.

Open: The opening price at which a security or index is traded at the beginning of a specific period, usually a day.

Close: The closing price at which a security or index is traded at the end of a specific period, usually a day.

Volume- It refers to the total number of shares or contracts of a particular security or index that were traded during a specific period, typically a day. Volume is an important metric for investors and traders as it provides an indication of the level of interest in a particular security or index. A high volume indicates that there is significant trading activity, which may be a result of significant news or events affecting the security or index.

Adj. Close-The high and low prices are important indicators of the range of prices at which a security or index trades during a day. The open price provides an indication of the sentiment of the market at the beginning of the day, while the close price provides an indication of the sentiment of the market at the end of the day.

Data Exploration-

Analysing and comprehending a dataset with the intention of finding intriguing patterns, trends, and insights is the process of data exploration. Data exploration in the context of stock market data often entails looking at historical price and volume data for a certain stock.

I have downloaded the data from the above link. The data is saved in the CSV format. I uploaded the CSV file in the data frame. The data has been explored in python using the NumPy and Panda Library. The following screenshot shows the ‘head’ of the dataset-

	Date	High	Low	Open	Close	Volume	Adj Close
0	2015-11-23	2095.610107	2081.389893	2089.409912	2086.590088	3.587980e+09	2086.590088
1	2015-11-24	2094.120117	2070.290039	2084.419922	2089.139893	3.884930e+09	2089.139893
2	2015-11-25	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
3	2015-11-26	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
4	2015-11-27	2093.290039	2084.129883	2088.820068	2090.110107	1.466840e+09	2090.110107

Data Cleaning-

The process of locating and fixing or deleting errors, inconsistencies, and inaccuracies in a dataset is known as data cleaning, sometimes known as data cleansing or data scrubbing. Data cleaning in the context of stock market data often entails locating and fixing flaws or discrepancies in historical price and volume data for a specific stock or group of stocks.

I am going to study data my data, fill in missing values if any, set the date as an index (if it is unique), remove duplicates if any, and lastly add features if needed. The following screenshot shows that while extracting the information about the data frame.

The following screenshot has given information about the dataset. As you can see, you have seven columns and none of the columns has null values (Fig.2). The data frame consists of 1825 rows (Fig 3) which correspond to 1825 days (5 years). The data is sufficient to predict the future prices of a stock market. The date column is the “Object” datatype. All the other columns of the data frame of “Float” datatype.

```
yahoo_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1825 entries, 0 to 1824
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1825 non-null   object 
 1   High       1825 non-null   float64
 2   Low        1825 non-null   float64
 3   Open       1825 non-null   float64
 4   Close      1825 non-null   float64
 5   Volume     1825 non-null   float64
 6   Adj Close  1825 non-null   float64
dtypes: float64(6), object(1)
memory usage: 99.9+ KB
```

```
yahoo_df.isna().sum()
```

```
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
Adj Close    0
dtype: int64
```

```
len(yahoo_df['Date'].unique())
```

```
1825
```

I have set date as an index for the data frame. This means that I have removed the serial number of the data frame and considered “Date” an index. The following screenshot is attached here-

```
# Set Date as index
yahoo_df.set_index('Date',inplace=True)
```

```
yahoo_df.head()
```

Date	High	Low	Open	Close	Volume	Adj Close
2015-11-23	2095.610107	2081.389893	2089.409912	2086.590088	3.587980e+09	2086.590088
2015-11-24	2094.120117	2070.290039	2084.419922	2089.139893	3.884930e+09	2089.139893
2015-11-25	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
2015-11-26	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
2015-11-27	2093.290039	2084.129883	2088.820068	2090.110107	1.466840e+09	2090.110107

DATA VISUALISATION

Data visualization is the graphical representation of data in a way that helps to communicate insights and patterns in the data to users. In the context of stock market data, data visualization involves creating charts, graphs, and other visual representations of historical price and volume data for a particular stock.

High, Low, Open and Close are plotted in the following time series. The time-series is the visual representation of change in the high, low, open, and close over time. The significant drop in March’2020 can be considered because of Covid-19 pandemic. It was the most confusing time for investors in the market as most of them were inclined towards the thought of taking money out of the market. Using different types of visualizations, we can understand our data in a better way.

The volume of yahoo being traded is shown in the second figure in which we can easily see how the volume of the stock being traded changes with respect to open, high, low, and close. Though they are following the same pattern over the last 5 years. The prices seem to be non-stationary. **I am considering high to be one of the features for predictions.**

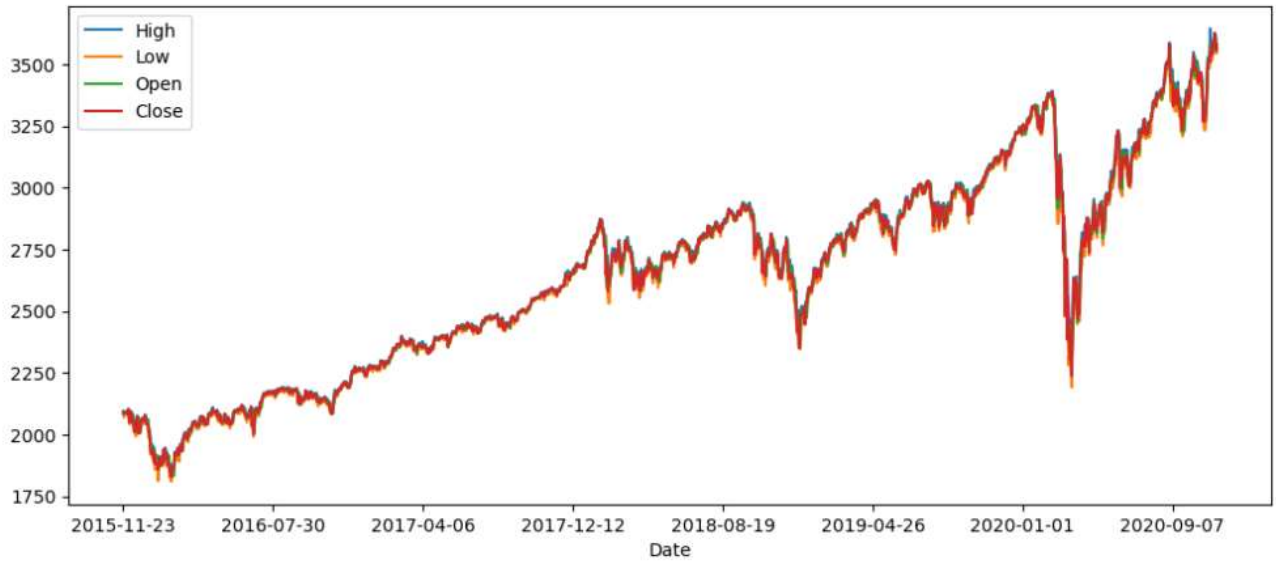


Figure 1 shows the time series graph from 2015 to 2020.

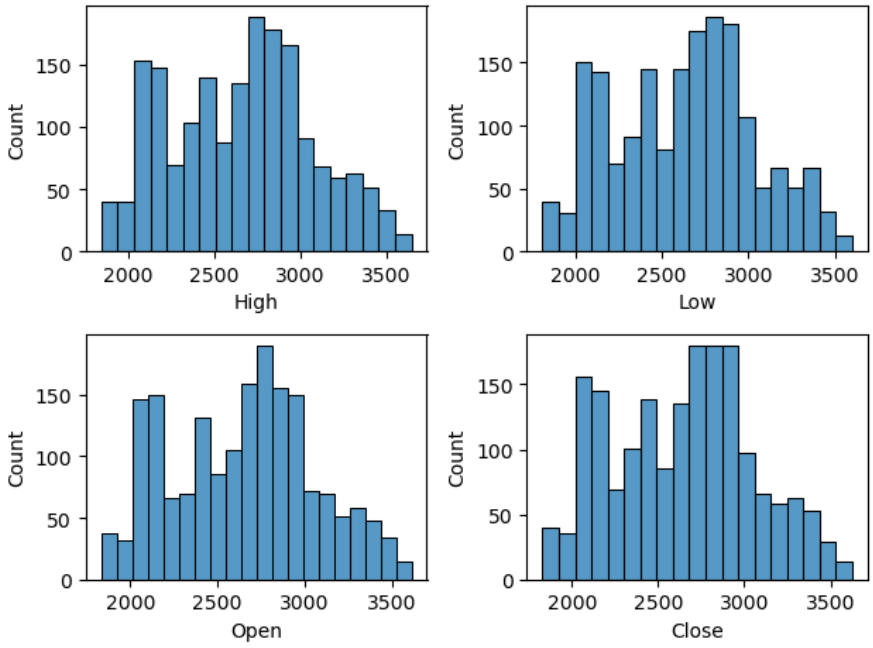


Figure 2 has 4 subplots of the movement of High, Low, Open and Close.

Data Splitting

Data splitting refers to the process of dividing a dataset into two or more subsets, typically for the purpose of training and testing machine learning models. In the context of stock data, data splitting may involve dividing historical stock data into a training set and a testing set.

I have considered high as a feature to predict the future prices of Yahoo. **I have created a data frame of high and then have split the high data frame in 80% training and 20% training dataset.** The training set is used to train a machine learning model, while the testing set is used to evaluate the performance of the model. The goal is to develop a model that can accurately predict future

stock prices based on historical data. This is one of the most crucial steps involved in the prediction process.

Following figure shows the top 5 rows of the high data frame.

```
high_df.head()
```

Date	High
2015-11-23	2095.610107
2015-11-24	2094.120117
2015-11-25	2093.000000
2015-11-26	2093.000000
2015-11-27	2093.290039

Following figure shows the division of data into 80% training and 20% testing.

```
training_size = int(len(high_df) * .8)
```

```
training_data = high_df.iloc[0:training_size]
testing_data = high_df.iloc[training_size:len(high_df)]
```

```
training_data[:5], testing_data[:5]
```

```
(                  High
 Date
 2015-11-23  2095.610107
 2015-11-24  2094.120117
 2015-11-25  2093.000000
 2015-11-26  2093.000000
 2015-11-27  2093.290039,
                   High
 Date
 2019-11-22  3112.870117
 2019-11-23  3112.870117
 2019-11-24  3112.870117
 2019-11-25  3133.830078
 2019-11-26  3142.689941)
```

After splitting the data into testing and training the next important step is to shape the training and testing data. The shaping of the dataset is essential because of the following reasons-

- Consistent data format: Shaping the data involves converting it into a consistent format that can be easily processed by machine learning algorithms.
- Feature Engineering- Shaping the data is a crucial step in feature engineering, as it involves selecting the relevant features and pre-processing them in a way that makes them suitable for use in the model.
- Model Compatibility- Different machine learning algorithms have different input requirements and may only be able to process data of a specific shape or type. Shaping the data involves ensuring that the input data is compatible with the chosen machine-learning algorithm.

```python
# Create dataset
def dataset(X, y, window=1):
    """
    Parameters
    ----------
    X - X data values
    y - y data values/target
    ------
    Return
    ------
    Numpy array for x and y
    """
    _x,_y = [], []

    for i in range(len(X) - window):
        vals = X.iloc[i: i+window].values # Temporary values
        _x.append(vals)
        _y.append(y.iloc[i+window])

    return np.array(_x), np.array(_y)
```

```python
Xy_train = training_data['High']
Xy_test = testing_data['High']
Xtrain, ytrain = dataset(training_data, Xy_train,10)
Xtest, ytest = dataset(testing_data, Xy_test,10)
```

Data Modelling

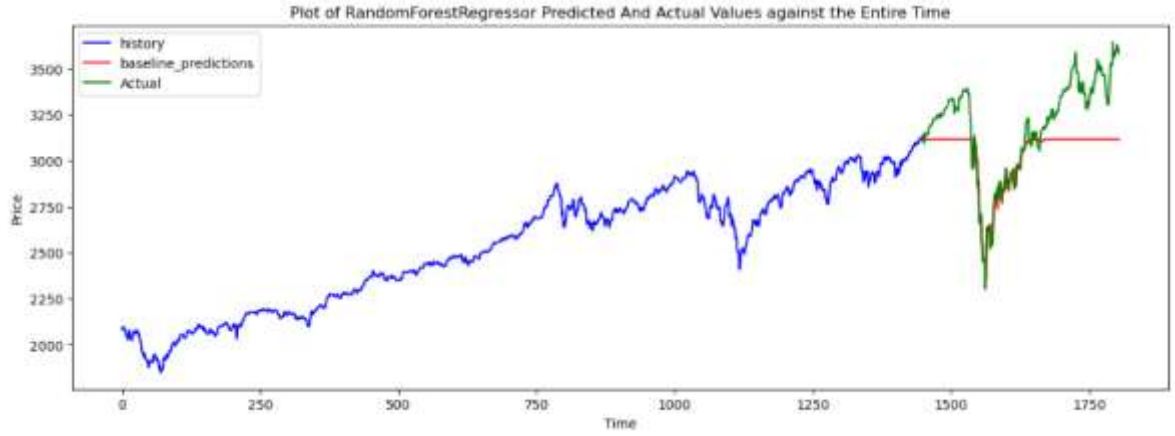
Data modelling refers to the process of creating a conceptual or mathematical representation of a set of data. The goal of data modelling is to create a structure that can be used to organize, understand, and analyse the data.

Random Forest Model

I have considered Random Forest as the first baseline model. In the context of my data, a Random Forest model can be trained on historical high prices to predict future stock prices.

One advantage of the Random Forest model is that it can handle large datasets with many features, which is important for analysing stock data that may contain many variables.

However, it is important to note that no machine-learning model can predict stock prices with 100% accuracy. Stock prices are influenced by a wide range of factors, including economic indicators, company news, and geopolitical events, that are difficult to predict using historical data alone.

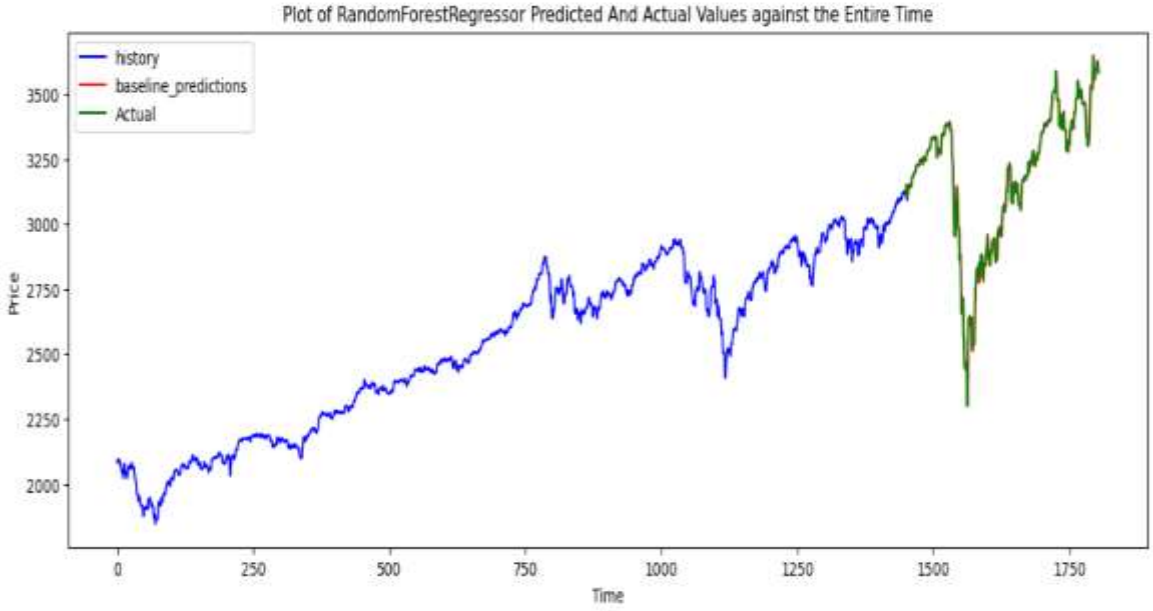


The above figure illustrates the difference between the actual prices of the stock and random-forest predictions. The model is not that accurate because the predicted number varies from the actual number in a very wide range. It is hence essential to create a second baseline model so that the actual and predicted values does not have such a vast difference.

Linear Regression

Linear regression is a statistical technique that can be used to analyse the relationship between two variables. In the context of our data, I have fit the training data firstly into the linear regression model. Based on this training data, the model is going to be trained and finally predict the stock prices. One advantage of using linear regression in stock data analysis is that it is a relatively simple and interpretable method. Linear regression models can provide insights into how changes in one variable may impact the stock price and can be used to identify trends and patterns in the data.

The baseline predictions are in line with the actual values. It means that this model is performing a lot better as compared to the first baseline model. We have calculated the accuracy of every model in the later part of the report, but in lame language after seeing the time-series plot, we can easily say that the model's performance is a lot better as compared to the first baseline model.



Simple Neural Network

Considering random forest and linear regression as the two baseline models, I have designed a simple neural network. For designing the simple neural network, I have used the keras library from tenser flow. The first step for designing a simple neural network is to set the hyperparameters. While designing the simple neural network, shaping the training data, and testing data is a very important feature. In the following figure, I have set the hyperparameters and shaped the input data in a particular way suitable for neural networks-

```
# Hypertext
INPUT_SHAPE = (Xtrain.shape[1], Xtrain.shape[2])
NEURONS = 64
EPOCHS = 25
BATCH_SIZE = 16
```

It is essential while designing the neural network to decide number of hidden layers, output layer and the activation function to be used. I have used "relu" as an activation function. Following is a screenshot of the summary of the simple neural network that I have designed-

```
SimpleNN = Sequential()
SimpleNN.add(Flatten(input_shape=INPUT_SHAPE))
SimpleNN.add(Dense(NEURONS, activation='relu'))
SimpleNN.add(Dense(1))

SimpleNN.compile(optimizer="adam",loss='mean_squared_error',metrics=['accuracy'])

SimpleNN.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 10)                0

 dense (Dense)               (None, 64)                704

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 769
Trainable params: 769
Non-trainable params: 0
_________________________________________________________________
```

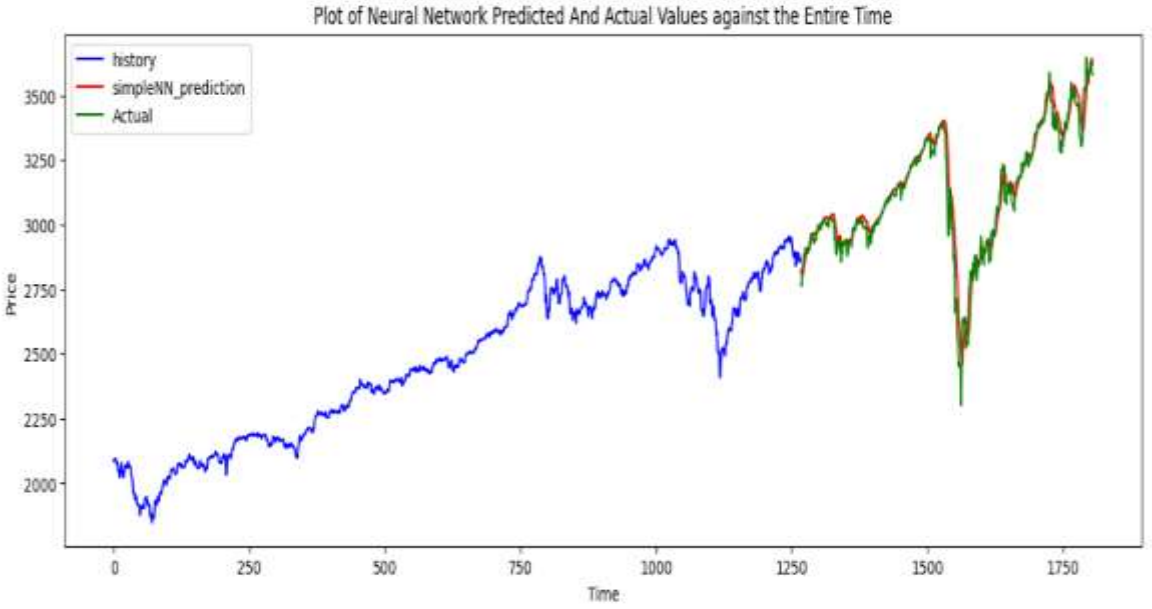
I have fit the training data to the simple neural network with the same hyperparameters as mentioned. Based on these numbers, the simple neural network is going to predict the future prices of the stock-

```
history = SimpleNN.fit(Xtrain,ytrain,epochs=EPOCHS, batch_size=BATCH_SIZE,verbose=0)
```

```
snn_preds = SimpleNN.predict(Xtest).reshape(ytest.shape)
```

```
12/12 [==============================] - 0s 4ms/step
```

Hence, the following figure shows that the performance of my simple neural network model is also good enough because the predicted values are in line with the actual values. It is one of the most prominent applications of the simple neural network model. I have calculated the accuracy of the model in the later part of the report.



Long Short-Term Memory Model

LSTM is the second model that I have designed to predict the future stock prices of yahoo. It is one of the most widely used model to predict the future stock prices. LSTM network use cells that have an internal recurrence + outer recurrence (like RNN). Internal Recurrence uses an additional St, which is used to remember the past. Mathematically, it has three gates-Forget Gate, External Input Gate and Output Gate.

Like SNN, I have set hyperparameters and developed the summary of the model. I will fit the training data to the model and the model will predict the future stock prices.

```
UNITS = 100
```

```
model = Sequential()

model.add(LSTM(NEURONS, activation='relu', input_shape=INPUT_SHAPE))

model.add(Dense(NEURONS/2))

model.add(Dense(1))

model.compile(optimizer='adam',loss='mean_squared_error')
```

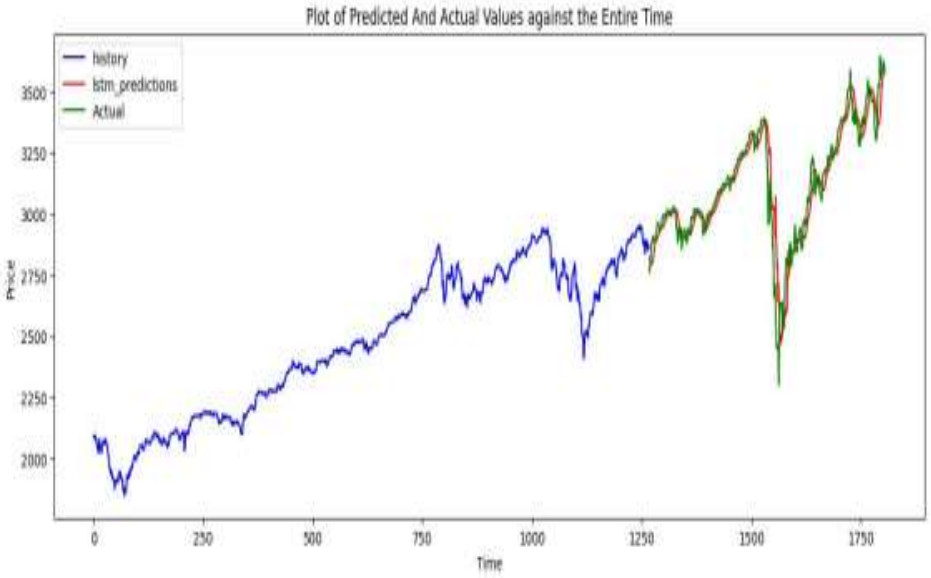
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 19,009
Trainable params: 19,009
Non-trainable params: 0
```

```
history = model.fit(Xtrain, ytrain, epochs=EPOCHS, batch_size=BATCH_SIZE)
```

```
Epoch 1/25
91/91 [==============================] - 3s 6ms/step - loss: 889946.1875
Epoch 2/25
91/91 [==============================] - 0s 5ms/step - loss: 24282.2344
Epoch 3/25
91/91 [==============================] - 1s 6ms/step - loss: 1592.6727
Epoch 4/25
91/91 [==============================] - 0s 5ms/step - loss: 887.0299
Epoch 5/25
91/91 [==============================] - 1s 5ms/step - loss: 854.0067
Epoch 6/25
91/91 [==============================] - 0s 5ms/step - loss: 1028.3512
Epoch 7/25
91/91 [==============================] - 0s 5ms/step - loss: 1034.2979
Epoch 8/25
91/91 [==============================] - 1s 6ms/step - loss: 1007.6666
Epoch 9/25
91/91 [==============================] - 0s 5ms/step - loss: 983.4651
Epoch 10/25
91/91 [==============================] - 0s 5ms/step - loss: 896.4224
Epoch 11/25
91/91 [==============================] - 0s 5ms/step - loss: 840.4919
Epoch 12/25
```

Finally, we have the plot of the predicted values and actual values here. Based on this figure also, we can say that the difference between the predicted and the actual values is not that great. They are close enough. In the next section of the report, I have calculated the accuracy of these models.



Model Accuracy-

In terms of stock prediction data, model accuracy refers to the ability of a machine learning or deep learning model to make accurate predictions of future stock prices or price movements based on historical data.

Evaluating the accuracy of a stock prediction model is a challenging task because stock prices are notoriously volatile and subject to many unpredictable factors such as market sentiment, news events, and geopolitical developments. Moreover, past performance may not always be a reliable indicator of future performance.

R^2 Value of the Model

R-squared (R^2) is a statistical measure that represents the proportion of variance in the dependent variable that is explained by the independent variables in a regression model. In machine learning and deep learning models, R^2 is often used as a metric to evaluate the performance and accuracy of a model. I have chosen to use the R^2 Value as the output is a continuous data.

The main reason R^2 is used is that it provides a simple and interpretable measure of how well the model fits the data. It ranges from 0 to 1, with 1 indicating a perfect fit and 0 indicating no correlation between the independent and dependent variables. A high R^2 value suggests that the model is a good fit for the data, whereas a low R^2 value suggests that the model is not a good fit.

It is important to note that R^2 is not the only metric used to evaluate machine learning and deep learning models, and it should not be used in isolation to make decisions about the suitability of a model for a particular task. Other metrics, such as mean squared error, mean absolute error, and

accuracy, should also be considered depending on the specific task and the type of model being used.

The following figure shows the R2 value of the models. Since, the visual representation of the Random Forest Model gave such a vague result, hence it was not necessary to calculate the R2 value of it. As you can see, the highest accuracy is that of Linear Regression followed by Simple Nueral Network, followed by LSTM.

```
print(f"Linear Regression: {round(lr_r2_score * 100, 2)}%")
print(f"Simple Neural Network: {round(snn_r2_score * 100, 2)}%")
print(f"LSTM: {round(lstm_r2_score * 100, 2)}%")
```

```
Linear Regression: 98.2%
Simple Neural Network: 94.69%
LSTM: 88.46%
```

But taking only the R^2 Value is not that good; hence I have calculated the mean squared error of model.

Mean Squared Error

Mean squared error (MSE) is a statistical measure used to evaluate the accuracy of a model's predictions. In the context of the stock market, MSE can be used to assess the accuracy of a stock price prediction model by comparing its predicted values to the actual stock prices.

In the context of the stock market, a lower MSE indicates that the prediction model is more accurate, whereas a higher MSE indicates that the model's predictions are less accurate. MSE can be used to compare the performance of different models or to assess the performance of a single model over time. However, it's worth noting that MSE is just one of many evaluation metrics that can be used to assess the accuracy of a stock price prediction model, and it's important to use multiple metrics in conjunction to get a comprehensive understanding of a model's performance.

```
mse_rf= mse = mean_squared_error(ytest, rfr_preds)
print('MEAN SQUARED ERROR OF RANDOM FOREST MODEL IS :',mse_rf)
```

```
MEAN SQUARED ERROR OF RANDOM FOREST MODEL IS : 44026.34849350825
```

```
mse_lr= mse = mean_squared_error(ytest, y_test_preds)
print('MEAN SQUARED ERROR OF LINEAR REGRESSION MODEL IS :',mse_lr)
```

```
MEAN SQUARED ERROR OF LINEAR REGRESSION MODEL IS : 1407.9879999593788
```

```
mse_snn= mse = mean_squared_error(ytest, snn_preds)
print('MEAN SQUARED ERROR OF SIMPLE NUERAL NETWORK MODEL IS :',mse_snn)
```

```
MEAN SQUARED ERROR OF SIMPLE NUERAL NETWORK MODEL IS : 6662.909553911988
```

```
mse_lstm= mse = mean_squared_error(ytest,y_preds)
print('MEAN SQUARED ERROR OF LSTM IS :',mse_lstm)
```

```
MEAN SQUARED ERROR OF LSTM IS : 7052.091000219802
```

CONCLUSION

From the above two metrics, it is very clear that I will consider Linear Regression is one of the best performing models to predict the future prices.

```
future_prices = model.predict(Xtrain)
```

```
40/40 [==============================] - 0s 3ms/step
```

```
actual_prices = ytrain.tolist()
```

```
future_prices = furure_prices.tolist()
```

```
data = list(zip(actual_prices,future_prices))
future_prices_df = pd.DataFrame(data, columns=['Actual Price', 'Future Prices'])
```

```
future_prices_df.head()
```

	Actual Price	Future Prices
0	2085.000000	2766.379637
1	2093.840088	2770.231421
2	2093.840088	2761.936151
3	2093.840088	2806.865606
4	2090.419922	2832.581053

THEORETICAL BACKGROUND OF TRADING STRATEGIES

Reference-

https://medium.com/codex/using-python-to-create-an-innovative-trading-strategy-and-achieve-better-results-702dcf4359ce

Based on the above information, I have chosen Linear Regression model as a proficient model to predict the stock prices. In this section of the paper, I have developed some trading strategies. By trading strategies, I mean to say when to buy/sell. I am going to use Stochastic Oscillator and Moving Average Convergence/Divergence (MACD) indicators to determine when to buy and sell.

Stochastic Oscillator

Stochastic oscillator is a popular technical indicator used in trading, particularly in analyzing the momentum of a security or an asset. The stochastic oscillator compares the closing price of a security to its price range over a given period, typically 14 days,

This indicator is momentum-based used for identifying the state of both overbought or oversold. In stock, we refer to market as overbought when market's trend is extremely bullish and bound to consolidate. The stocks are referred to oversold when trends have a tendency to bounce. To identify overbought and oversold, a stochastic oscillator uses %K Line and %D Line.

%K AND %D LINE

%K Line is represented as follows for a period of 14 days:

%K = (Current Close - Lowest Low) / (Highest High - Lowest Low) * 100

where the Current close is the most recent closing price of the stock.
the lowest low is the lowest price in the last 14 days.
the highest high is the highest price in the last 14 days.

The **%D line is a three-day simple moving average of the %K line in the stochastic oscillator.**

Moving Average Convergence/Divergence (MACD)

MACD, or Moving Average Convergence Divergence, is a popular technical analysis indicator that measures the relationship between two exponential moving averages (EMA) of an asset's price. It is designed to identify changes in the strength, direction, momentum, and duration of a trend.

The MACD indicator consists of three components: the MACD line, the signal line, and the histogram.

The MACD line is the difference between the 26-period EMA and the 12-period EMA.
The signal line is the 9-period EMA of the MACD line.
The histogram represents the difference between the MACD line and the signal line.

TRADING STRATEGY

BUY – %K and %D are below 25
SELL- %K and %D are above 70.

BUY- MACD Line and Signal Line < -2
SELL- MACD Line and Signal Line >+2

IMPLEMENTING TRADING STRATEGY

I have taken this function from the reference website mentioned above to calculate the stochastic oscillator.

CALCULATION OF %K AND %D

```
def calculate_stochtick(high, low, close, preriod=14, dl=3):
    l_low = low.rolling(preriod).min()
    h_high = high.rolling(preriod).max()
    k_line = 100 * ((close - l_low) / (h_high - l_low))
    d_line = k_line.rolling(dl).mean()
    return k_line, d_line

yahoo_df['%k'], yahoo_df['%d'] = calculate_stochtick(yahoo_df['High'], yahoo_df['Low'], yahoo_df['Close'])
yahoo_df.tail(10)
```

So basically using the above function, I am intending to calculate %K and %D line. We got the following result upon applying the above function. The columns “%k” and “%d”

Date	High	Low	Open	Close	Volume	Adj Close	%k	%d
2020-11-06	3521.580078	3484.340088	3508.340088	3509.439941	4833950000	3509.439941	93.354986	90.028326
2020-11-07	3521.580078	3484.340088	3508.340088	3509.439941	4833950000	3509.439941	93.354986	93.469069
2020-11-08	3521.580078	3484.340088	3508.340088	3509.439941	4833950000	3509.439941	93.354986	93.354986
2020-11-09	3645.989990	3547.479980	3583.040039	3550.500000	8556610000	3550.500000	76.825633	87.845202
2020-11-10	3557.219971	3511.909912	3543.260010	3545.530029	6024230000	3545.530029	75.619476	81.933365
2020-11-11	3581.159912	3557.000000	3563.219971	3572.659912	4609970000	3572.659912	82.203599	78.216236
2020-11-12	3569.020020	3518.580078	3562.669922	3537.010010	4890120000	3537.010010	73.551761	77.124946
2020-11-13	3593.659912	3552.570068	3552.570068	3585.149902	4709670000	3585.149902	85.234782	80.330048
2020-11-14	3593.659912	3552.570068	3552.570068	3585.149902	4709670000	3585.149902	85.234782	81.340442
2020-11-15	3593.659912	3552.570068	3552.570068	3585.149902	4709670000	3585.149902	83.388372	84.619312

Calculation of MACD Line, Signal Line and Histogram-

```
def calulate_macd(df, slow, fast, smooth):
    macd = ta.trend.MACD(df['Close'], window_slow=slow, window_fast=fast, window_sign=smooth)
    df['MACD_Line'] = macd.macd()
    df['Signal_Line'] = macd.macd_signal()
    df['Histogram'] = macd.macd_diff()
    new_df = df.dropna()
    return new_df
```

From the above function, I calculated the three components of MACD:

- MACD Line
- Signal Line
- Histogram

Date	High	Low	Open	Close	Volume	Adj Close	%k	%d	MACD_Line	Signal_Line	Histogram
2015-12-26	2067.360107	2058.729980	2063.520020	2060.989990	1411860000	2060.989990	81.152663	81.152663	-4.428167	-10.675039	6.246871
2015-12-27	2067.360107	2058.729980	2063.520020	2060.989990	1411860000	2060.989990	81.152663	81.152663	-2.900093	-9.120049	6.219956
2015-12-28	2057.770020	2044.199951	2057.770020	2056.500000	2492510000	2056.500000	71.676752	77.994026	-2.028011	-7.701642	5.673631
2015-12-29	2081.560059	2060.540039	2060.540039	2078.360107	2542000000	2078.360107	95.802247	82.877221	0.422182	-6.076877	6.499059
2015-12-30	2077.340088	2061.969971	2077.340088	2063.360107	2367430000	2063.360107	76.124981	81.201327	1.140457	-4.633410	5.773867

After calculating the Stochastic Oscillator and MACD Components, now I am finally implementing a function where we are selling/buying a stock with respect to our trading strategy as explained above.

```
def stock_strategy(prices, kl, dl, macd, macd_signal):
    bp = [] #buying_price
    sp = [] #selling_price
    sm_signal = [] #stoch_macd_signal
    signal = 0

    for i in range(len(prices)):
        # ((%K & %D) < 25) & ((MACD LINE & SIGNAL LINE) < -2)
        if kl[i] < 25 and dl[i] < 25 and macd[i] < -2 and macd_signal[i] < -2:
            if signal != 1:
                bp.append(prices[i])
                sp.append(np.nan)
                signal = 1
                sm_signal.append(signal)
            else:
                bp.append(np.nan)
                sp.append(np.nan)
                sm_signal.append(0)
        # ((%K & %D) > 75) & ((MACD LINE & SIGNAL LINE) > 2)
        elif kl[i] > 75 and dl[i] > 75 and macd[i] > 2 and macd_signal[i] > 2:
            if signal != -1 and signal != 0:
                bp.append(np.nan)
                sp.append(prices[i])
                signal = -1
                sm_signal.append(signal)
            else:
                bp.append(np.nan)
                sp.append(np.nan)
                sm_signal.append(0)
```

```
        else:
            bp.append(np.nan)
            sp.append(np.nan)
            sm_signal.append(0)

    return bp, sp, sm_signal
```

Finally, I am implementing Linear Regression model to predict the future high prices. I implemented linear regression on the test data. There ae 536 datapoints in the test data but these are the first 5 rows that show the value of “High Predicted” -

Date	High	Low	Open	Close	Volume	Adj Close	%k	%d	MACD_Line	Signal_Line	Histogram	High Predicted
2019-05-30	2799.000000	2776.739990	2786.939941	2788.860107	3273790000	2788.860107	19.092333	11.145720	-24.378031	-19.343782	-5.034249	2787.877708
2019-05-31	2768.979980	2750.520020	2766.149902	2752.060059	3981020000	2752.060059	1.141108	11.228045	-28.123209	-21.099668	-7.023541	2801.089665
2019-06-01	2768.979980	2750.520020	2766.149902	2752.060059	3981020000	2752.060059	1.141108	7.124850	-30.736972	-23.027129	-7.709844	2766.135608
2019-06-02	2768.979980	2750.520020	2766.149902	2752.060059	3981020000	2752.060059	1.301150	1.194455	-32.434514	-24.908606	-7.525908	2766.187889
2019-06-03	2763.070068	2728.810059	2751.530029	2744.449951	3943810000	2744.449951	11.165783	4.536014	-34.001947	-26.727274	-7.274673	2770.450534

Now based on the above result and predicted high value, an investor can decide whether to buy/sell a stock.