Journal of Advanced Research Design

Journal homepage:
https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Intelligent Forecasting Based on Long-Short Term Memory

Ahmad Syakirin Shahrunnizam[1], Shahrum Shah Abdullah[1,*]

[1] Department of Mechanical Precision Engineering, Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia

ARTICLE INFO

Article history:
Received 23 March 2022
Received in revised form 7 October 2022
Accepted 19 October 2022
Available online 27 October 2022

Keywords:
Intelligent forecasting; stock markets; long short term memory (LSTM) analysis; machine learning

ABSTRACT

Stock markets are non-linear, and stock market research has become an important topic. Usually, people invest in the stock market based on traditional methods, such as fundamental and technical analysis. In the stock market prediction prices, various tricks and tools are used to minimize the risk, at the same time increase the profits. Forecasting is crucial in the stock market, especially in business which the prediction algorithm can be very complicated and necessitating. Alternatively, the traditional analysis may not guarantee the solid outcome of the stock analysis. For the prediction, Long Short Term Memory (LSTM) analysis is mainly used. This work examines the efficiency of machine learning and predicts stock market prices. This paper also proposes finding the most ideal variables by producing the least Mean Absolute Percentage Error (MAPE).

1. Introduction

The stock market is non-linear, which the output is difficult to predict. Stock market research has become a crucial topic in recent years. In some forecast-based markets, people invest in stocks. To predict the market price of inventory, people look for such methods and tools. Forecasting is very important, especially in the highly complexed inventory process. Methods such as fundamental analysis, technical analysis and regression analyses are primarily used to make predictions.

Accurate forecasting of future stock prices can generate significant returns. The hypothesis of the efficient market assumes that currently, available information is reviewed as stock prices, so price movements that are not recently have discovered information are unpredictable. Those who hold this view are free to use many methods and techniques to enable them to be informed about prices in the future. Because of the underlying structure of the financial sphere and in part due to the combination of known parameters, most closing price and unknown elements, stock prices are believed to be particularly dynamic and subject to rapid adjustments e.g. pandemic outbreak, political instability, etc. An astute trader would forecast the stock price and purchase or sell a stock before it rises or falls in value. Though it is difficult to replace an experienced trader's expertise, an accurate prediction algorithm can directly result in significant profits for investment firms, suggesting

* *Corresponding author.*
E-mail address: shahrum@utm.my

that there is a direct relationship between the quality of the prediction algorithm and the profit made from utilising it.

Every day, the stock market is mentioned in the news when each time it achieves a new high or low. If an effective algorithm could be established to anticipate the short term price of an individual stock, the rate of investment and business prospects in the stock market may increase. According to Svetlana's research [10], using a Simple Moving Average to forecast stock prices is not pragmatic nor acceptable. Forecasting accuracy ranges from 45.63 % on the NASDAQ market to 52.53 % on the SSE market. Nevertheless, in this research, we investigated whether if a model based on Long-Short Term Memory can be developed to predict stock price with a lower proportion of inaccuracy.

2. Literature Review

2.1 Machine Learning

Machine learning is an artificial intelligence field that allows computers to learn and improve without having to be explicitly programmed. Machine learning is the process of developing computer programmes that can access data and learn on their own. Machine learning, like the human brain, relies on input, such as training data or knowledge graphs to grasp entities, domains and their connections. Deep learning can begin once entities have been defined. Observations or data, such as examples, direct experience or instruction are used to start the machine learning process. It searches for patterns in data so that it can draw conclusions based on the examples presented. The basic goal of machine learning is to enable computers to learn on their own, without the need for human interaction and to change their behaviour accordingly.

2.2 Artificial Neural Network (ANN)

With the advancement of modern technology, stock market forecasting is now also moving into the realm of technology. ANN (Figure 1) and genetic algorithms (GA) are most commonly used for stock prediction [3]. Scientists have discovered that bacterial chemotaxis optimization methods may be superior to GA. ANN can be thought of as an approximation of a mathematical formula. The feed-forward networks, which is a common form of ANN are normally used for stock prediction. The weights are updated from the backward error propagation algorithm of the ANN. Other than that, the recurrent neural network (RNN) or a time-delay neural network (TDNN) is better suited for inventory forecasting with time interval compared to ANN structure. The difference is that the RNN uses the feedback connection which is beneficial for stock analysis [1].

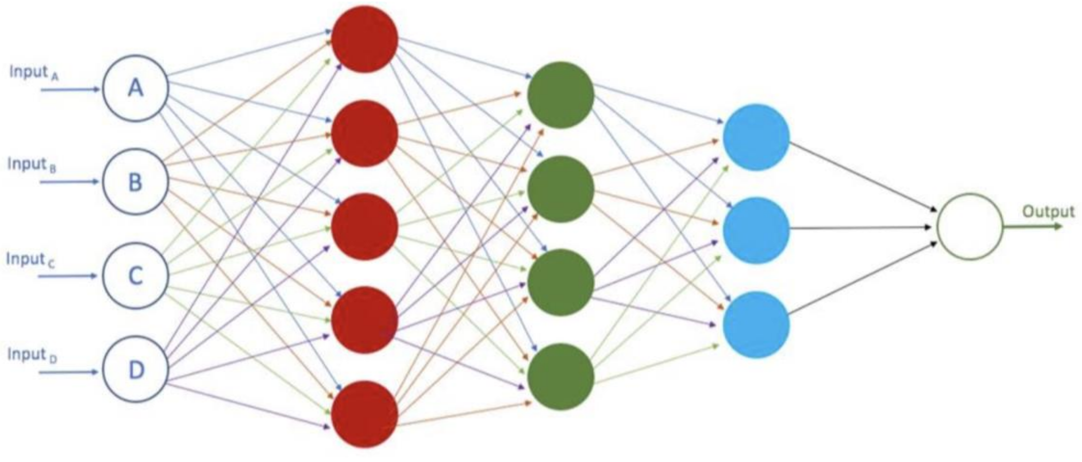


Fig. 1. Artificial neural network

2.3 Recurrent Neural Network (RNN)

A RNN (Figure 2) is a type of ANN in which the connections between nodes generate a directed or undirected graph over time [7,8]. This enables it to behave in a temporal dynamic manner. RNNs, which are derived from feed-forward neural networks, can process variable length sequences of inputs by utilising their internal state (memory). The phrase "RNN" describes a type of network that has an infinite impulse response. Additional stored states are possible in both finite and infinite impulse recurrent networks, and the storage can be controlled directly by the neural network. If another network or graph involves time delays or has feedback loops, it can likewise be used to replace the storage. Gated state or gated memory refers to these controlled states, which are seen in long short-term memory networks (LSTMs). It's also known as a Feedback Neural Network (FNN).

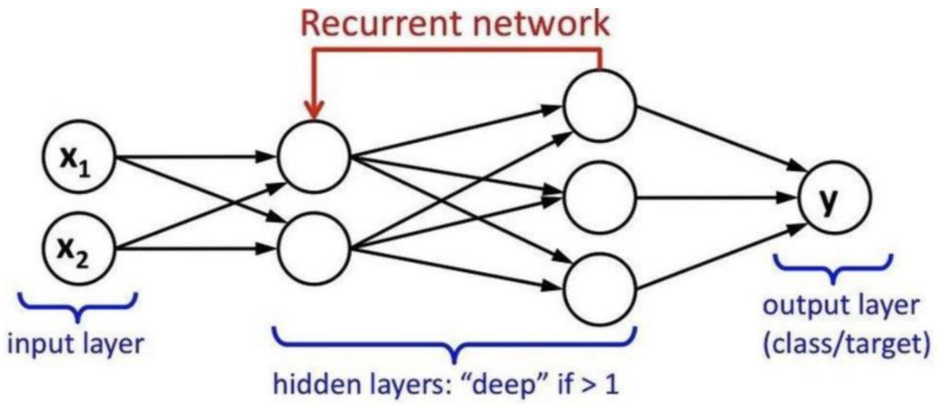


Fig. 2. Recurrent neural network

2.3.1 Long short term memory (LSTM)

LSTM (Figure 3) is similar to RNN structure used in deep learning. Unlike the common neural networks, LSTMs use feedback in the algorithm. LSTM is not limited to processing a single data point, but it can also process the whole data sequence. The LSTM structure consists of different parts with different functions such as a forget gate, an input gate and an output gate. The gates are in charge of the access of the information in and out of the cell. Meanwhile, a random time interval is stored in the cell [2].

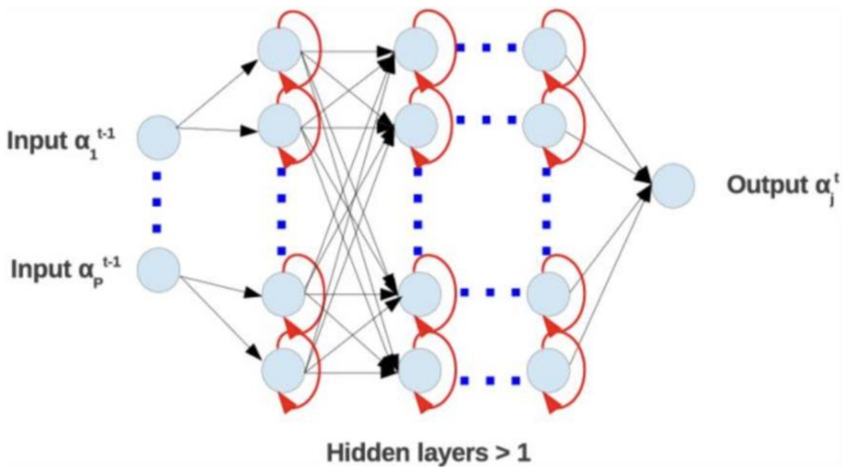


Fig. 3. Long short term memory network

2.4 Convolutional Neural Network (CNN)

CNN (Figure 4) is a deep learning model for processing data with a grid pattern, such as photographs, that is inspired by the organization of animal visual cortex and meant to learn spatial hierarchies of characteristics, from low- to high-level patterns, automatically and adaptively. CNN are made up of three types of layers which are convolution, pooling and fully connected layers. The first two layers, convolution and pooling, extract information while the third, a fully linked layer transfers those features into final output such as categorization. A convolution layer is an important part of CNN which is made up of a stack of mathematical operations like convolution, which is a particular sort of linear operation.

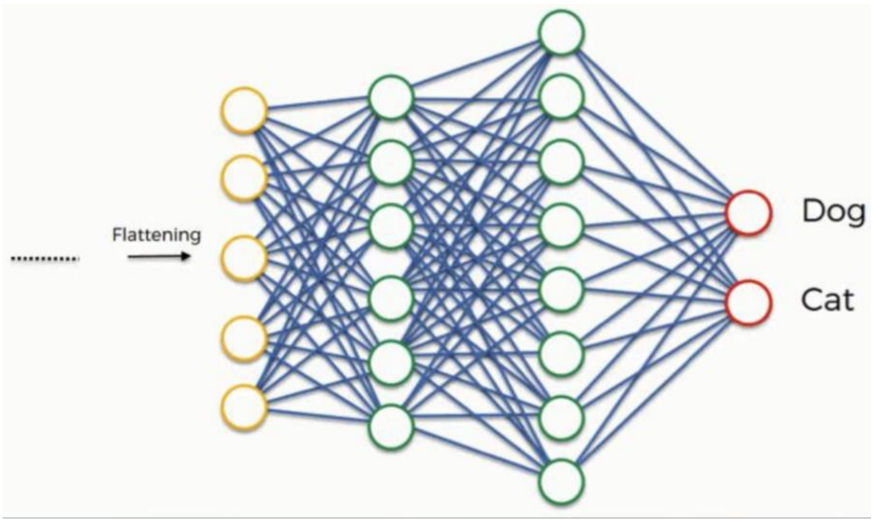


Fig. 4. Convolutional neural network (CNN)

2.5 Previous Works on Stock Prediction

Research conducted by Weng [4] focused on forecasting short-term stock prices using the machine learning method. Five datasets were used for this survey. Open-source application programming interfaces (APIs) and a R package called TTR were used to gather these records. Neural network regression ensemble (NNRE), an unpruned regression tree as a base learner (RFR) and a base learner with a tree as BRT were used in this study. The short-term stock price forecasting was thoroughly examined. With background knowledge, a few indicators were selected and in five datasets.

On this platform, users do not have to enter their data, but they do need to call the API to get the information directly from the source. Whit that, they assess price forecasts 1 to 10 days ahead, not more than 2 or less than 1 day. The main limitation of their study was to analyse only 20 and to determine if the model could be generalized to other stocks, was required [4].

Besides selection of the Bitcoin characteristics, the LSTM parameters were picked out by the Bayesian optimization. The Bitcoin dataset ranged from the 19th of August 2013 to the 19th of July 2016 and used a couple of the overall deep gaining knowledge technique performance. Prediction of Bitcoin price trend has a few resemblances with inventory marketplace price prediction. Noises embedded and hidden capabilities with inside the price records are an obstacle to this work [5].

Utilising ANN in stock prediction used data from India's National Stock Exchange. A prediction model was created using the Multilayer Perceptron (MLP) Neural Network method. Throughout the year of 2015 to 2017, the MLP prediction model was trained and tested. The conclusions were in direct opposition to those of earlier studies and researches. The MLP neural network performed well with a Median Normalized Error of 0.05995 and a Median Standard Deviation of 6.39825. With positive results, the MLP was used to predict the NSE's LIX15 index [9].

According to Jamous *et al*., [10], as ANNs have been utilised to estimate stock but on the other hand, have a variety of drawbacks that result in reduced prediction accuracy. As a result, they used a combination technique for an effective stock market prediction. This work proposed PSOCoG, a new improved technique for achieving high- in a short period of time [10].

To build a model for predicting future stock, Hamiche was used and in particular the Long–Short Term model. The main goal is to determine how accurate can the forecast be and how many epochs can aid in the improvement of our model. It has been demonstrated that training with substantial fewer data and epochs enhances our testing outcomes while also helping us to acquire better forecast and for a variety of datasets. The goal of their research in the future is to determine what data durations are appropriate for the assets [11].

The daily prices and volumes analyses of the top 10 S and P 500 equities were used by Qui and Zhou [12]. The data covered the years 2004 through 2013. The training, development and testing data were split into the four models: ARIMA, LSTM, Stacked-LSTM and Attention-LSTM. For the model evaluation, MSE was considered. Due to the long-term dependence of time series, Attention-LSTM was revealed to be the most excellent of the four models; the model can predict financial time series considerably better [12].

The S&P 500 Index, Dow Jones and Hang Seng Index were among the stocks that Qiu and colleagues used to create an Attention-based LSTM model to anticipate future stocks. The findings were compared using the Gated Recurrent Unit. From January 2000 to July 2019, data for the S&P 500 Index and the DJIA were collected. The HSI data was also gathered between January 2002 and July 2019. The data contained a total of six variables. The MSE, RMSE, MAE and Coefficient of Determination (R2) were utilised; the smaller the MSE, RMSE, MAE, the closer the actual value and forecast are [13].

Mallikarjuna and Pramod applied the LSTM approach, created a stock data predictor. The model used the LSTM model to anticipate price changes based on a company's past equity share prices. Opening, closing, day low, day high, trading date, turnover and total quantity traded were all included in the historical data. The model was used to forecast the price movement of TATAMOTORS' stock. It successfully obtained a 96 % accuracy using LSTM units and a 50-epoch batch size. The outcomes were unmistakable. The model had a loss rate and moved in lockstep with the real value of the stock [14].

3. Methodology

3.1 Flow Chart

The Figure 5 is the representation of the steps that was applied throughout this research project. The first step was to import the stock market data. The stock data was read using the pandas.read_cvs function. There are few parameters in the stock data, but the one that was used is the closing price, discarding other unnecessary parameters. Next is the input variable. From the stock data, 65 % of the data was used for the training process, while the remaining data was for the test prediction. Three input variables were used in finding the best variables with the best prediction accuracy. The LSTM process trains the dataset from the stock data. The input variables was used in the training process. After that is to plot the chart of the training and the test result and prediction for the next 30 days. In terms of finding the best accuracy, mean absolute percentage error was used.

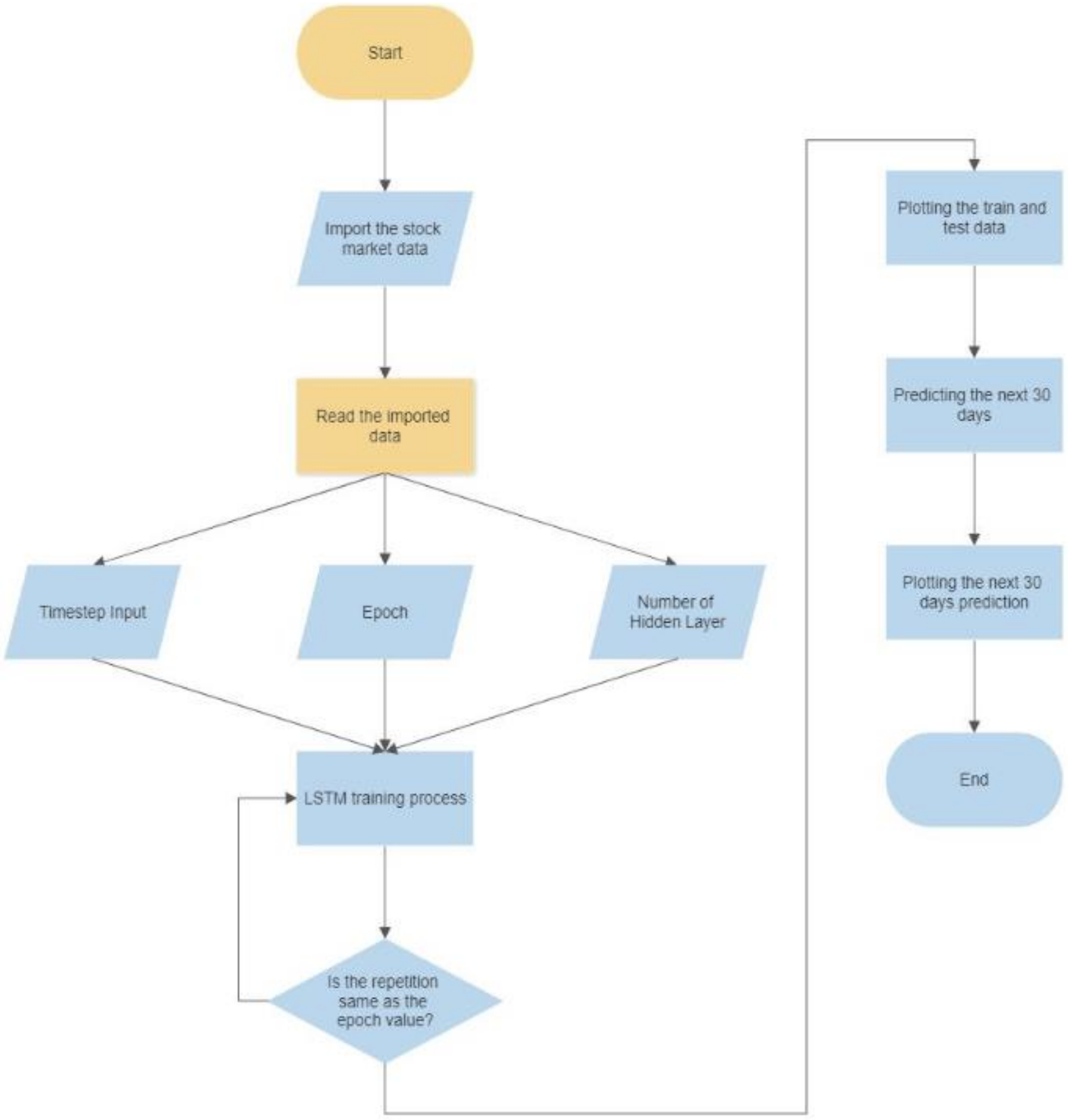

Start
Import the stock market data
Read the imported data
Timestep Input
Epoch
Number of Hidden Layer
LSTM training process
Is the repetition same as the epoch value?
Plotting the train and test data
Predicting the next 30 days
Plotting the next 30 days prediction
End

Fig. 5. Flow chart for the prediction algorithm

3.2 Introduction to Long Short Term Memory (LSTM)

The LSTM networks are mainly used in machine learning, especially for prediction. Therefore LSTM are able to analyse long-term dependencies, which is an advantage for prediction problems. The LSTM uses feedback connection in the algorithms, which enables processing of the entire data sequence. This applies to speech recognition, machine translation, etc [9]. LSTMs are a special type of RNN that exhibits excellent performance for a variety of problems.

3.3 Fundamental Knowledge of LSTM
3.3.1 LSTM design

LSTMs are similar to basic RNN cells. The LSTM consists of three gates. Each gate performs a different function as shown in Figure 6. The first gate is called the Forget Gate, where it chooses whether to keep the information from the previous timestamp or completely erases it if it's not needed. The next gate is called the Input Gate, where the new information is learned from the input to that cell. Finally, in the final gate is the Output Gate, where the current timestamp updated information is passed to the next timestamp [6].

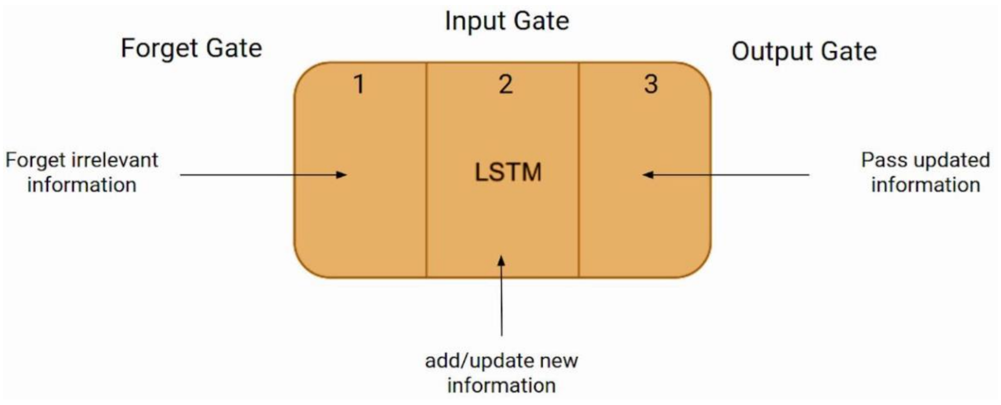


Fig. 6. The gates in LSTM

LSTMs have hidden states. The hidden state for the previous timestamp is represented by H (t-1) and the hidden state of the current timestamp is represented by H (t). In addition, LSTMs have cell states represented by C (t-1) for the previous and C (t) for current timestamps as in Figure 7. Plus, the cell state and the hidden state are called short- term memory and long-term memory, respectively [6].

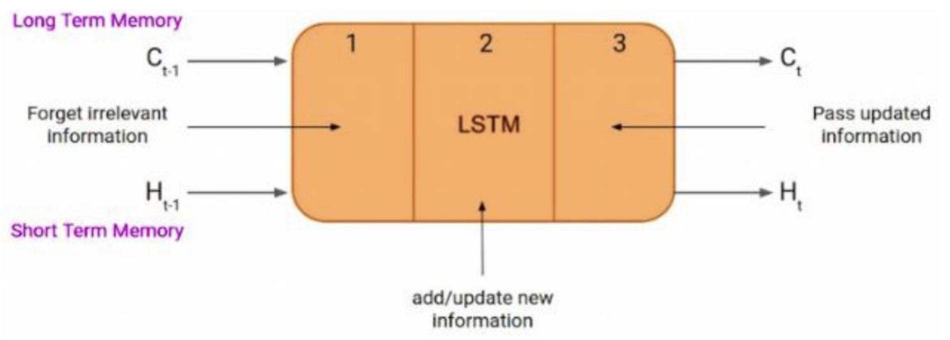


Fig. 7. The Long Term and Short Term Memory in LSTM

Firstly, the LSTM needs to decide whether to keep or forget the information from the previous timestamp. Below is the formula for Forget Gate,

$$f_t = \sigma\ (x_f * U_f + H_{t-1} * W_f) \qquad (1)$$

where the equation functions are [6],

X_t: input to the current timestamp.
U_f: weight associated with the input

H_{t-1}: The hidden state of the previous timestamp
W_f: It is the weight matrix associated with the hidden state

Next, the function is applied to a sigmoid function, resulting in the f_t output, between 0 and 1.

$$C_{t-1} * f_t \tag{2}$$

From the equation above, when theft is 0 the output of the multiplication will be 0, therefore the cell will erase the information. However, if f_t equals 1, the network will not erase the information. Next is the Input Gate, where the crucial new data carried by the input data is purposely assessed. The equation is shown below,

$$I_t = \sigma\ (x_t * U_i + H_{t-1} * W_i) \tag{3}$$

The functions are [6],

X_t: input to the current timestamp
U_f: weight associated with the input
H_{t-1}: The hidden state of the previous timestamp
W_f: It is the weight matrix associated with the hidden state

The sigmoid function will be applied to the formula resulting between 0 and 1. With the new information added, the above equation is slightly altered resulting in the below function,

$$N_t = \tanh\ (x_t * + H_{t-1} * W_c) \tag{4}$$

From the new equation, the hidden state function at t_{-1} and input x at t needed cell state to pass the new information. T_{anh} acts as the activation function and because of that, the new data value is between -1 and 1. When the value of N_t is positive, the data is added to the cell state at the current timestamp while if it's negative, the data will be subtracted. Therefore, the cell equation is as below [6],

$$C_t = C_{t-1} * f_t + i_t * N_t \tag{5}$$

In this equation, the latest timestamp only applies to C_{t-1} while the other from previously calculated. Finally, the last gate in the cell is the Output Gate. The general equation for the Output Gate is below,

$$O_t = \sigma\ (x_t * U_o + H_{t-1} * W_o) \tag{6}$$

Because a sigmoid function will be applied to the equation, the value will also be between 1 and 0. As for the current hidden state, O_t and t_{anh} will be used to update the cell state. The equation is as follows,

$$H_t = o_t * \tanh\ (C_t) \tag{7}$$

Based on the function above, the hidden state is a function of current output and the long-term memory, C_t. As for a clearer view, Figure 8 is the image of the LSTM cell.

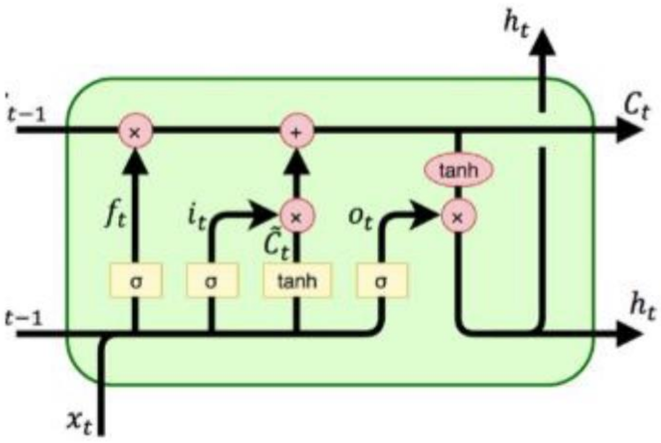


Fig. 8. The clear representation of LSTM

3.4 Tensorflow

Dataflow graph structures that explain how data passes through a graph or a sequence of processing nodes can be created with TensorFlow. Each node in the graph symbolises a mathematical process, and each node-to-node connection or edge is a tensor or multidimensional data array. TensorFlow is a set of workflows which is mainly for developing and training models in Python or JavaScript, and for deploying them in the cloud, on-premise, in the browser or other devices regardless of the language. The Tensorflow data API allows to create complicated input pipelines out of reusable components.

3.5 The Stock Data

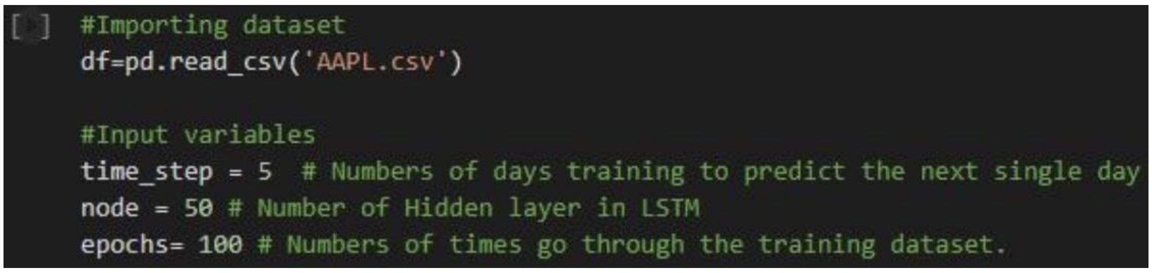
The stock market data that was used in this project is from Apple stock data. This data can be acquired through any historical market and stock data website. For this report, Yahoo Finance was picked to acquire this data due to its accessibility and ease of obtaining. The format of the stock data file was in the .csv type files. This file contained a data list, which is frequently used to transfer data between applications. For the implementation into the prediction algorithm, Pandas library was applied as it contains the .read_csv model to read the stock data. The stock data that was displayed consist of multiple parameters such as the opening price, closing price, the daily high price, the daily low price and etc. For this application to predict the stock price, only the closing price parameter was used, as it is an indicator of the sell price for a specific day.

3.6 The Input Variables

The input variable consists of two types of variables, manipulated and constant variables. The manipulated variables incorporate the uses of the number of days trained and the number of the hidden layers in the LSTM model. The number of days trained, known as timestep, is the amount of days that is required to be trained by the prediction model to predict the next single day. In addition to the present data point, LSTM utilises prior data to make contextual and more accurate predictions. The number of days trained specify how far back in time needed to look to make a prediction, or how many past data points required to incorporate alongside the current data point. The number of the hidden layer is the amount of the hidden layer in between the input and output layer, where a set of weighted inputs is taken by the artificial neurons and uses the activation function to produce the output. Both of these parameters will manipulate, observe and record, studying the effect on the performance and accuracy of the prediction. The constant variables in this study will be the repetition to train the model, known as epoch. Based on the observation, epochs were categorized as constant variables due to its small changes on the performance and the accuracy of the prediction. Plus, this also acts as a standardized variable throughout the datasets.

3.7 The Algorithm

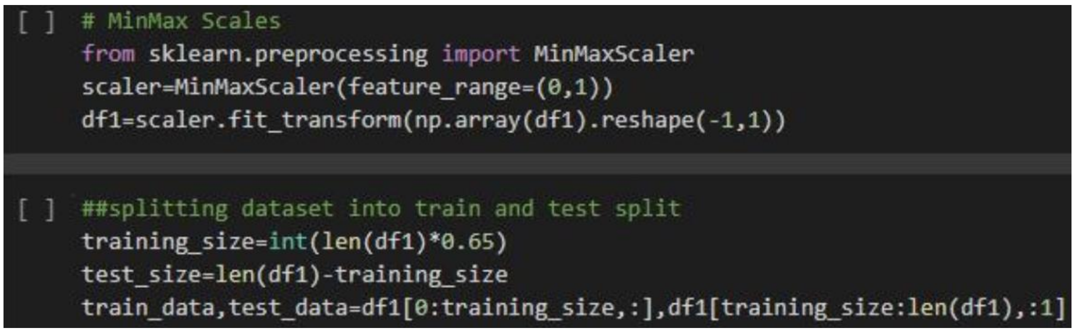
In Figure 9, the pandas read csv command is used because it is a quick and easy way to store large data sets. Next is the input variables, consisting of the time_step, node and epochs. This represents the number of days training to predict the next single day, number of hidden layers in LSTM and the number of repetitions through the training datasets respectively.



```
#Importing dataset
df=pd.read_csv('AAPL.csv')

#Input variables
time_step = 5  # Numbers of days training to predict the next single day
node = 50 # Number of Hidden layer in LSTM
epochs= 100 # Numbers of times go through the training dataset.
```

Fig. 9. Algorithm on reading the stock data and input variables

In Figure 10, MinMax scaler function was utilized. The purpose of this is to convert the big data (closing price) into a small form factor, between 0 and 1. The reason behind this is that the LSTM function can be sensitive towards big data value. Having scales from 0 to 1 will improve the LSTM process. The data frame is split up into two categories, training and testing.

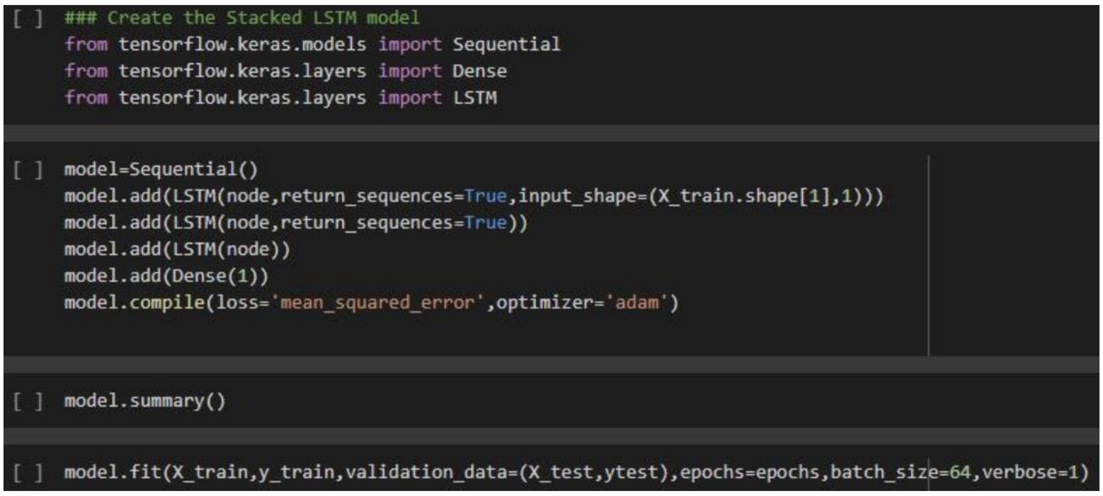


```
# MinMax Scales
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

```
##splitting dataset into train and test split
training_size=int(len(df1)*0.65)
test_size=len(df1)-training_size
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

Fig. 10. MinMax scales and separation of the datasets

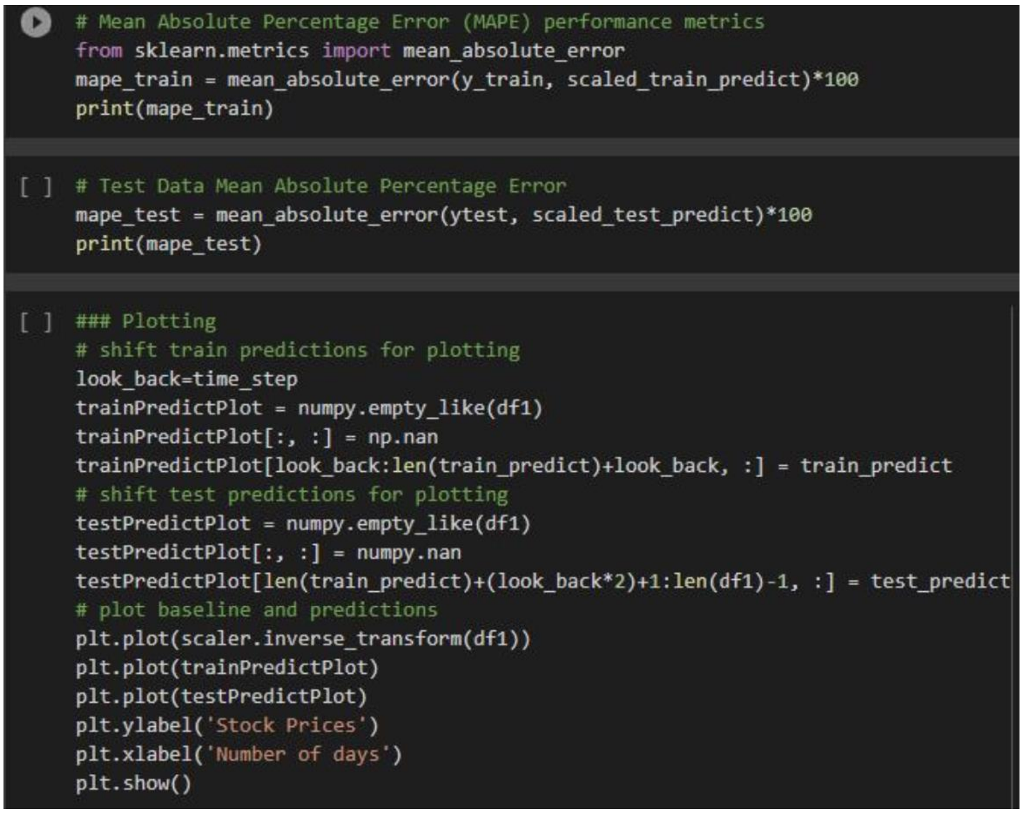
From the code script in Figure 11, the LSTM function consists of the units and return sequences. According to the node value, the units denote the number of hidden layers that will be employed. For each input time step, the return sequences were used to obtain the concealed state output. The training process begins at the model.fit function.



```
[ ] ### Create the Stacked LSTM model
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.layers import LSTM

[ ] model=Sequential()
    model.add(LSTM(node,return_sequences=True,input_shape=(X_train.shape[1],1)))
    model.add(LSTM(node,return_sequences=True))
    model.add(LSTM(node))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error',optimizer='adam')

[ ] model.summary()

[ ] model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=epochs,batch_size=64,verbose=1)
```

Fig. 11. The LSTM function

In Figure 12, the accuracy measurement for the prediction is by using the Mean Absolute Percentage Error (MAPE). It is determined as the average absolute percent inaccuracy for each time period minus actual values divided by real values, and it evaluates this accuracy as a percentage. The units of the variable are scaled to percentage units, making it easier to grasp.



```
# Mean Absolute Percentage Error (MAPE) performance metrics
from sklearn.metrics import mean_absolute_error
mape_train = mean_absolute_error(y_train, scaled_train_predict)*100
print(mape_train)

[ ] # Test Data Mean Absolute Percentage Error
    mape_test = mean_absolute_error(ytest, scaled_test_predict)*100
    print(mape_test)

[ ] ### Plotting
    # shift train predictions for plotting
    look_back=time_step
    trainPredictPlot = numpy.empty_like(df1)
    trainPredictPlot[:, :] = np.nan
    trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
    # shift test predictions for plotting
    testPredictPlot = numpy.empty_like(df1)
    testPredictPlot[:, :] = numpy.nan
    testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict
    # plot baseline and predictions
    plt.plot(scaler.inverse_transform(df1))
    plt.plot(trainPredictPlot)
    plt.plot(testPredictPlot)
    plt.ylabel('Stock Prices')
    plt.xlabel('Number of days')
    plt.show()
```

Fig. 12. MAPE and the train and test data plotting

Next is the plotting of the train and to test the predicted chart. Because of the time_step, the initial time_step data frame was not plotted as it is used for the training and prediction. The chart starts the training and test plotting after the initial time_step data frame.

4. Results and Discussion

4.1 Mean Absolute Percentage Error (MAPE)

Table 1 indicates two MAPE, both for train data and test data. The main MAPE for choosing the best number of days to train for the prediction is the MAPE on Test Data value. The purpose of having both MAPE is to record the LSTM output whether it is over-fitted or under-fitted. The over-fitting happens when the test data RMSE value is lower than the train data RMSE. From the result given, there are no over-fitted outputs.

The variable used in this first approach in finding the best value for predicting the stock value is the time_step. The number of days train of 5 is the best variable for the prediction, as the MAPE value for the test data, at 11.60 % is the lowest compared to other training days. These are only the first step in finding the best value for the prediction. Next table will show the accurate result, affected by various numbers of hidden layers in the neural network.

Table 1
MAPE result from different number of days train

Number of Hidden Layers = 10, Training Repetition (Constant) = 100		
Number of days train	MAPE on Train Data (%)	MAPE on Test Data (%)
5	8.27	11.60
10	10.17	14.57
15	10.83	18.19
2	11.52	17.21
25	14.55	20.06
30	18.20	21.43
35	18.02	21.54
40	17.92	20.13
45	16.99	19.14
50	16.65	18.08

The variable used in this second approach in finding the best value for predicting the stock value is the number of the hidden layers (Table 2) for the LSTM algorithm. The number of hidden layers of 50 is the best amount of the hidden layer for the prediction, as the MAPE value for the test data, at 11.21 % is the lowest compared to previous 10 hidden layers at 11.60 % and the rest of the hidden layer.

Table 2
MAPE result from different number of hidden layers

Number of Hidden Layers	MAPE on Train Data (%)	MAPE on Test Data (%)
5	8.79	12.10
10	8.27	11.60
15	8.40	11.58
20	8.22	11.54
25	7.58	11.44
30	7.11	11.34
35	6.82	11.31
40	6.47	11.23
45	6.35	11.22
50	6.35	11.21

Note: Number of day train = 5, Training repetition (Constant) = 100

Based on the result given, the number of hidden layers does affect the accuracy of the prediction. Below is the graph plot for the train and the test data alongside with the original chart.

4.2 Chart Diagram

From Figure 13, we can see the train and the test data chart is neither over-fitted nor under-fitted. The train data and the test data set uses 5 days time_step for training and predicting the output of the next single day respectively. Therefore the first 5 days used in the train data and the 5 days after the day 163 in the chart for the test data were not plotted.

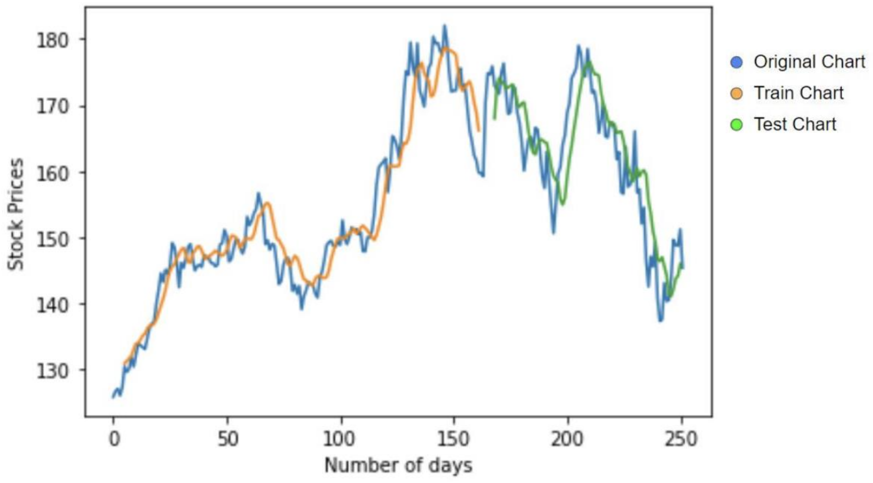


Fig. 13. Graph for the train and test data

Based on the below 30 days graph in Figure 14, the previous 5 days are used to predict the next, day 6 and later. Therefore the initial day for the next 30 predictions starts at day 6.

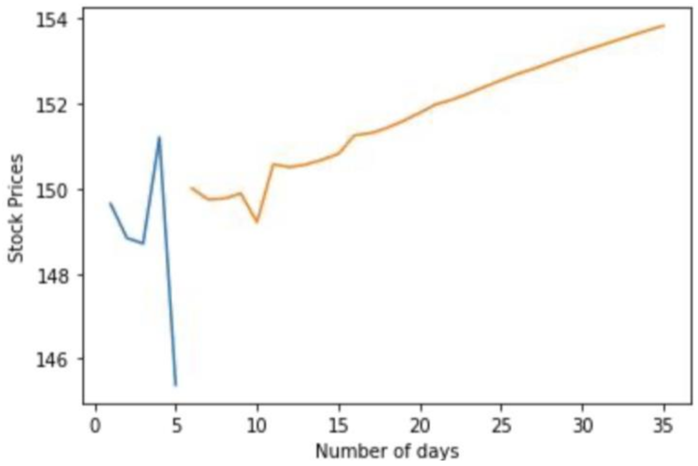


Fig. 14. Graph for the next 30 days prediction

As for the results as in Figure 15, the time_step (days needed to predict the next single day) and the number of hidden layers to be used in this model does affect the output prediction accuracy. The number of days train and the number of hidden layers of 5 and 50 respectively, are the most ideal variables in producing the most accurate prediction, by producing the least mean absolute percentage error at 11.21 %. Based on the predicted graph, the stock price is on an upward trend.

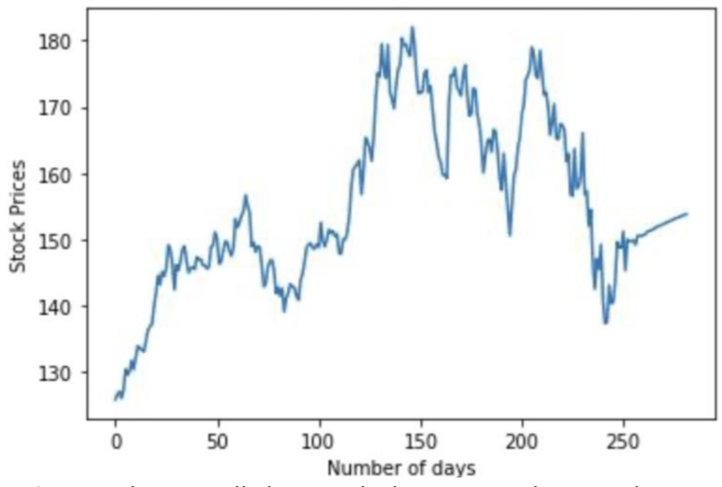


Fig. 15. The overall chart with the next 30 days prediction

5. Conclusion

The success in designing the stock prediction algorithm by using the tensorflow LSTM model does give encouragement in pursuing machine learning. Other than that, the achievement in finding the ideal days of training and number of hidden layers for the most efficient prediction performance based on the value of the MAPE is a total success. The accuracy of the prediction was meant to be calculated through the Root Mean Squared Error, however due to its non-percentage form, the accuracy representation was difficult to understand. The algorithm was meant for research purposes, and it is not advisable to use it as a reference in real life stock prediction. There are many other variables which are not included in this research, which might bring a major effect on the prediction accuracy.

References

[1] Eugene, Fama. "Efficient capital markets: A review of theory and empirical work." *Journal of finance* 25 (1970): 383-417. https://doi.org/10.1111/j.1540-6261.1970.tb00518.x

[2] Liu, Shuanglong, Chao Zhang, and Jinwen Ma. "CNN-LSTM neural network model for quantitative strategy analysis in stock markets." In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*, pp. 198-206. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-70096-0_21

[3] Kim, Kyoung-jae, and Ingoo Han. "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index." *Expert systems with Applications* 19, no. 2 (2000): 125-132. https://doi.org/10.1016/S0957-4174(00)00027-0

[4] Weng, Bin, Lin Lu, Xing Wang, Fadel M. Megahed, and Waldyn Martinez. "Predicting short-term stock prices using ensemble methods and online data sources." *Expert Systems with Applications* 112 (2018): 258-273. https://doi.org/10.1016/j.eswa.2018.06.016

[5] McNally, Sean, Jason Roche, and Simon Caton. "Predicting the price of bitcoin using machine learning." In *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)*, pp. 339-343. IEEE, 2018. https://doi.org/10.1109/PDP2018.2018.00060

[6] *Saxena, Shipra. "Introduction to long short term memory (lstm)." Analytics Vidhya (2021).*

[7] Dupond, Samuel. "A thorough review on the current advance of neural network structures." *Annual Reviews in Control* 14, no. 14 (2019): 200-230.

[8] Shahvaroughi Farahani, Milad, and Seyed Hossein Razavi Hajiagha. "Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models." *Soft computing* 25, no. 13 (2021): 8483-8513. https://doi.org/10.1007/s00500-021-05775-5

[9] Dzikevicius, Audrius, Svetlana Saranda, and Aleksandra Kravcionok. "The accuracy of simple trading rules in stock markets." *Economics and Management* 15 (2010): 910-916.

[10] Jamous, Razan, Hosam ALRahhal, and Mohamed El-Darieby. "A new ann-particle swarm optimization with center of gravity (ann-psocog) prediction model for the stock market under the effect of covid-19." *Scientific Programming* 2021 (2021): 1-17. https://doi.org/10.1155/2021/6656150

[11] Moghar, Adil, and Mhamed Hamiche. "Stock market prediction using LSTM recurrent neural network." *Procedia Computer Science* 170 (2020): 1168-1173. https://doi.org/10.1016/j.procs.2020.03.049

[12] Zou, Zhichao, and Zihao Qu. "Using LSTM in stock prediction and quantitative trading." *CS230: Deep learning, winter* (2020): 1-6.

[13] Qiu, Jiayu, Bin Wang, and Changjun Zhou. "Forecasting stock prices with long-short term memory neural network based on attention mechanism." *PloS one* 15, no. 1 (2020): e0227222. https://doi.org/10.1371/journal.pone.0227222

[14] Pramod, B. S. "Mallikarjuna Shastry P." *M,"Stock Price Prediction Using LSTM," Test Eng. Manag* 83 (2020).