IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 8, August 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

Stock Market Prediction using Univariate LSTM model

Srikala Prabhu[1], Prof. Jayshree Kundargi[2]

[1] Department of Electronics and telecommunication, KJ. Somaiya College Of Engineering, Mumbai, Maharashtra, India

[2] Department of Electronics and telecommunication, KJ. Somaiya College Of Engineering, Mumbai, Maharashtra, India

Abstract

In the financial sector, stock market prediction using deep learning models has gained a lot of attention. LSTM is one of the popular models that is used widely for predicting time series. The success of LSTM is largely dependent on the selection of several hyper-parameters, which must be done with care to achieve decent results. There are no defined recommendations for configuring LSTM. In this paper close value of Apple stock price is predicted considering historical data of the previous 10 years. The univariate LSTM model is used for predicting Apple stock prices. Two LSTM variants are used for the prediction of values. RMSE is used as an evaluation metric for predicted values.

Keywords: *LSTM,* ***Stock market, deep learning, forecasting, prediction.***

1. Introduction

Time Series Forecasting has always been a very important area of research in many domains because many different types of data are stored as time series. For example, we can find a lot of time series data in medicine, weather forecasting, biology, supply chain management, and stock prices forecasting. As of 8th August 2021, the worldwide apple stock is 146.21 USD. Because of the high-risk and high-yield nature of the stock market, investors aim to forecast it using a scientific manner. Furthermore, reliable forecasting findings give investors a point of reference to help them reduce investment risk. As a result, finding an adequate model to reliably predict and forecast the price of financial assets is both theoretically and practically important. LSTM is an RNN variation (Recursive Neural Network). It was created to address the issue of RNN quickly dissipating. The LSTM can be thought of as a collection of transition gates. It can skip some units and memorize longer time steps. As a result, LSTM can partially solve the gradient disappearance problem. In this paper, the LSTM model is used for predicting close values of the apple stock market. A univariate model is built,i.e only a single attribute 'Close value' is considered and predicted. Vanilla LSTM and Stacked LSTM models are used for prediction. The accuracy of these models is evaluated using RMSE.

2. Related Work

Time Series Forecasting has always been a very important area of research in many domains because many different types of data are stored as time series. For example, we can find a lot of time series data in medicine, weather forecasting, biology, supply chain management, and stock prices forecasting. New stocks are listed virtually every day as the stock market develops, and there is an increasing number of stocks that can be traded [1]. As a result, deciding on an appropriate stock (investment) becomes a major challenge. As a result, a scientific and efficient approach to replace traditional stock trading methods is required to assist with stock trading guidance. Traditional methods for predicting time series include autoregressive moving average (ARIMA), support vector regression (SVR), and random forest (XGBoot) [2]. However, because stock price data is non-linear and has high noise, these statistical methods can be difficult to make high-precision predictions, and thus do not apply to stock prediction. Studies have shown that most of the time series prediction processing uses the LSTM (Long Short-Term Memory) network [3], which is a time recurrent neural network based on RNN (Recurrent Neural Network). Hochreiter & Schmidhuber proposed, and recently improved and promoted by Alex Graves, LSTM will have selective memory features on long-term scales, suitable for processing and predicting important events with relatively long intervals and delays in time series. In [6] authors have presented univariate and multivariate LSTM models is used for predicting stock values with not much accuracy. Hence a model is required which efficiently predicts with high accuracy.

3. Theory

Time Series Forecasting

A time series is an ordered collection of observations recorded over a while. In other words, a time series is a sequential grouping of data based on the time of occurrence. The term "time-series" refers to a relationship between two variables, one of which is time and the other of which might be any quantitative variable. In every Time Series Analysis, time serves as a reference point.
Forecasting data using time-series analysis entails utilizing a significant model to predict future events based on known historical results. Models capable of predicting future values based on previously observed values are known as time-series forecasting models. It's a popular choice for non-stationary data. Non-stationary data are those whose statistical features, such as mean and standard deviation, change with time and are not constant. The non-stationary data is fed into these models as input. The stock market price over time, the price of a property over time, a patient's ECG data over time, and so on are all examples of time series. So, the input is a signal (time-series) that is defined by observations taken sequentially in time
Time series forecasting is approached using a variety of models and methods. Autoregressive (AR), Integrated (I), Moving Average (MA), and additional models that combine these models, such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA), are some of the most used time-series models. :
• Model performance is harmed by missing values, and they are unable to discern complicated patterns in the data.
• They usually perform well for short-term forecasts, but not for long-term forecasts.
Here deep learning models come into the picture, which is designed to overcome these problems. Some of the deep learning models used for time series forecasting are as follows
• Recurrent Neural Networks (RNNs): This is the most common and widely used architecture for Time Series Forecasting problems, although it has several drawbacks that will be described further.
• Long Short-Term Memory (LSTM): These are a type of RNN that was developed to solve the problem of vanishing gradients.
By studying various available works in time series forecasting which are also mentioned in the literature survey section, Long Short-Term Memory (LSTM) models are found to be the most effective in the prediction of time series and handle long structures very well. Before studying LSTM, understanding RNN's limitation is important.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are networks of neuron-like nodes that are grouped into layers and have a similar architecture to regular Neural Networks. Neurons are separated into input, hidden, and output layers, just like normal Neural Networks. Each neuronal connection has a trainable weight associated with it.
RNNs are meant to handle a wide range of complicated computer tasks, including speech recognition, object classification, and so on. Understanding how RNNs work is critical for understanding LSTMs because LSTMs have a similar control flow to RNNs.RNNs are looped networks that allow information to survive[9].

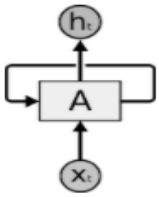

Fig. 1. RNNs having loop

In Fig. 1, A is a hidden layer that examines some input x_t and returns a result h_t. Information can be passed from one network step to the next via a loop. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor, as follows :

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 8, August 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

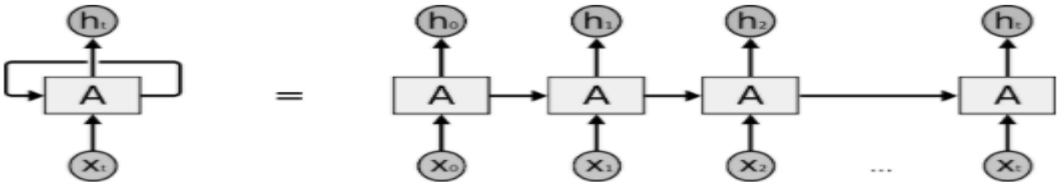

Fig. 2. Chain structured RNNs

In Fig. 2 chain-like structure above reveals that recurrent neural networks are related to sequences and lists and hence are used widely with sequence-related applications.

Sometimes, we only need to look at recent information to perform a certain task. For example, if we want to predict the last word in the sentence “Fruits grow on trees”, we don’t need any other information as it is pretty obvious the next word will be ‘tree’. In such cases, where the gap between the given information and the place that it’s needed is small, RNNs can learn to use the past information.

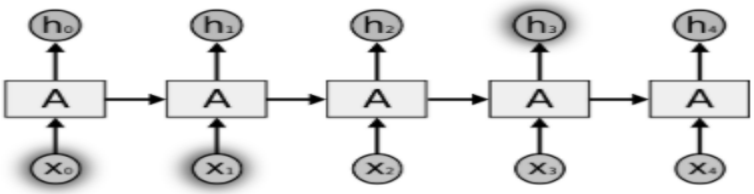

Fig. 3. A small gap in past information

But there are also cases where we need more information. Consider if we want to predict the last word in the text “I live in India… I speak fluent Hindi.” Recent information suggests that the next word is probably the name of a language, but if want to predict which language, we need the context of Hindi, from further back. The gap between the relevant information and the point where it is needed can become very large.

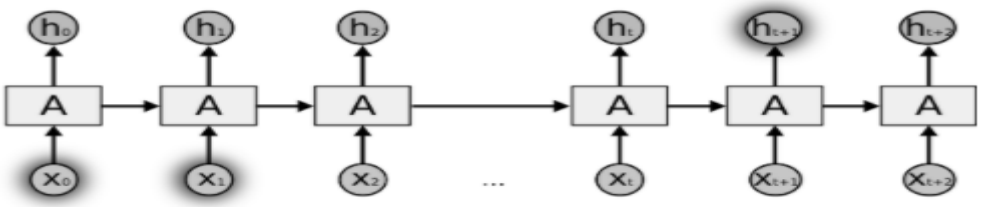

Fig. 4. A large gap in past information

Unfortunately, if the gap widens, RNNs will be unable to learn and connect the data. Though RNNs are capable of handling a series of events that occur in rapid succession, each event's understanding is predicated on knowledge from preceding events. They should ideally be given the deepest RNNs so that they can have a longer memory period and higher capabilities. These could be applied for many real-world use-cases such as stock prediction and enhanced speech detection. However, they are rarely used because of their vanishing gradient problem. This problem was explored in depth by Hochreiter [5].

The Vanishing Gradient Problem

This is one of the most difficult problems for RNNs to solve. RNNs architecture limits their long-term memory capacity, allowing them to recall only a few sequences at a time. They have a problem with short-term memory. They'll have a hard difficulty transporting information from earlier time steps to later ones if the sequence is long enough. RNNs may forget critical information from the beginning if they are used to forecast the next sequence from a paragraph. As a result, RNN memory is only helpful for short sequences and periods. When using backpropagation, the vanishing gradient problem limits the memory capabilities of typical RNNs, and adding too many time-steps raises the risk of running into a gradient problem and losing information. Gradients are values that are used to update the weights of a neural network. When a gradient reduces as it back propagates through time, this is known as the vanishing gradient problem. As a result, RNN

layers that get a modest gradient update stop learning. These are usually the first layers, so they don't learn. RNNs can forget what they've seen in longer sequences, thus they only have a short-term memory. When it comes to predicting COVID-19 cases, the sequence is lengthier, and its prediction will have a direct impact on the medical resources that will be provided and how the pandemic will be handled. As seen in the literature survey section, LSTMs outperform RNNs.

Long Short-Term Memory (LSTM)

Long Short Term Memory networks "LSTMs" are a special kind of RNN that are capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber in 1997 [5] and were refined and popularized by many people.

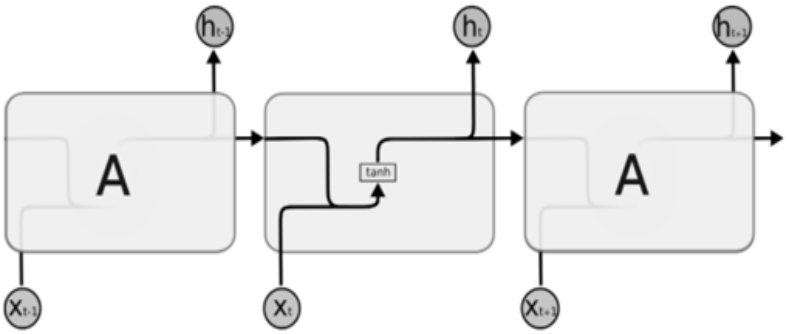

Fig. 5. Single-layer RNN

LSTMs also have this chain-like structure as above [9], but the repeating module has a different structure. Instead of having a single neural network layer like in Fig. 5, there are four, interacting in a very special way which can be seen in Fig. 6.

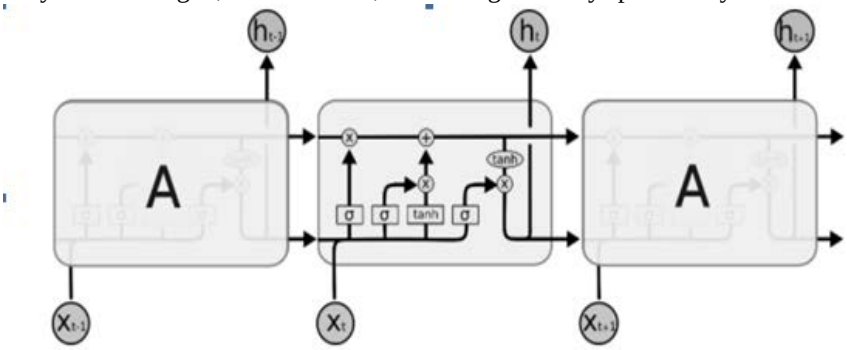

Fig. 6. RNN with four interacting layers

LSTM network is comprised of different memory blocks called cells. There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory are done through three major mechanisms, called **gates.** Each of them is being discussed below.

- **Forget Gate**

Taking the example of a text prediction problem. Let's assume an LSTM is fed in, the following sentence: **"Alex is the police. Mike is a thief "**.As soon as the first full stop after *"police"* is encountered, the forget gate realizes that there may be a change of context in the next sentence[11]. As a result of this, the *subject* of the sentence is *forgotten* and the place for the subject is vacated. And when we start speaking about *"Mike"*, this position of the subject is allocated to *"Mike"*. This process of forgetting the subject is brought about by the forget gate

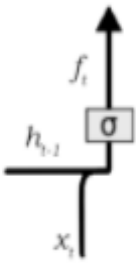

Fig. 7. Forget gate

A forget gate, Fig.7 is responsible for removing information from the cell state. When any information is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network. This gate takes in two inputs; h_{t-1} and $x_{t.}$

The equation of forget gate is as below

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

h_{t-1} is the hidden state from the previous cell or the output of the previous cell and $x_{t.}$ is the input at that particular time step. The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state. The sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely. Similarly, a '1' means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied by the cell state.

- **Input Gate**

For input gate, fig.8 let's take another example where the LSTM is analyzing a sentence: " Alex knows painting. He told me over a call that he went to an art school." Now the important information here is that "Alex" knows swimming and that he went to an art school. This can be added to the cell state, however, the fact that he told all this over the call is less important and can be ignored. This process of adding some new information can be done via the input gate. Here is the structure of the input gate.

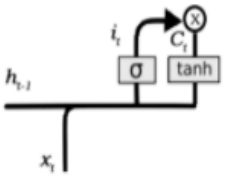

Fig. 8. Input gate

Equation of input gate[14]:

$$i_t = \sigma(W_i \times [h_{t-1}, x_t]) + b_i$$

$$C_t = \tanh(W_c * [h_{t-1}, x_t] + b_c)$$

The input gate is responsible for the addition of information to the cell state. This addition of information is a three-step process as seen from the diagram above.

1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is very similar to the forget gate and acts as a filter for all the information from h_{t-1} and x_t.

2. Creating a vector containing all possible values that can be added (as perceived from h_{t-1} and x_t) to the cell state. This is done using the **tanh** function, which outputs values from -1 to +1.

3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done, we ensure that only that information is added to the cell state is important.

- **Output Gate**

Not all information that runs along with the cell state, is fit for being output at a certain time. We'll visualize this with an example: "Alex painted many paintings over 50 years. He was called as artistic _____." In this phase, there could be several options for the space. But we know that the current input of *'artistic'*, is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Alex could be an appropriate output. This job of selecting useful information from the current cell state and showing it out as output is done via the output gate. Here is its structure:

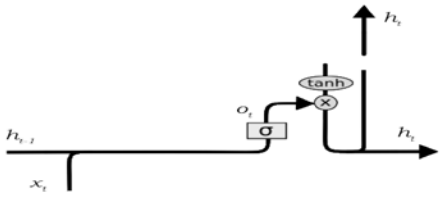

Fig. 9. Output gate

Output gate Equation is as below [10]:

$$O_t = \sigma(W_o[h_{t-1}, x_t]) + b_0$$

$$h_t = o_t * \tanh(C_t)$$

The functioning of an output gate can again be broken down into three steps:

1. Creating a vector after applying the **tanh** function to the cell state, thereby scaling the values to the range -1 to +1.

2. Making a filter using the values of h_{t-1} and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.

3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as output and also to the hidden state of the next cell.

The filter in the above example will make sure that it diminishes all other values but *'Alex'*. Thus the filter needs to be built on the input and hidden state values and be applied to the cell state vector.

There are various types of LSTMs used for time series forecasting stated below[12].

1. Univariate LSTM Models
    - Vanilla LSTM
    - Stacked LSTM
    - Bidirectional LSTM
    - CNN LSTM
    - ConvLSTM
2. Multivariate LSTM Models
    - Multiple Input Series.
    - Multiple Parallel Series.
3. Multi-Step LSTM Models
    - Data Preparation
    - Vector Output Model
    - Encoder decoder Model

From the above-mentioned variants, Vanilla LSTM also known as simple LSTM, and Deep LSTM is also known as stacked LSTM will be used for forecasting stock market data

- **Stacked LSTM**

Stacked LSTM will be used as a model for forecasting in further work along with Vanilla LSTM.
Stacked LSTM can be characterized as an LSTM model that involves multiple LSTM layers as explained. Selecting how deeper the model should be is another aspect of hyperparameter optimization which can generally go from a single layer to three to four-layer deep model architecture where a three-layer architecture is used mostly in complex learning tasks.

4. Proposed Work

Block diagram of the proposed work

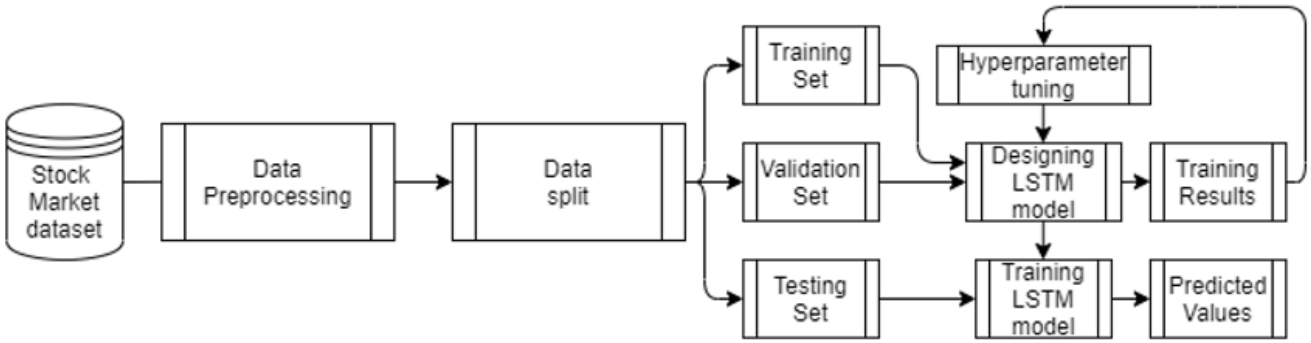

Fig. 10. Block Diagram of Proposed Work

Fig. 10 shows the flow of the proposed work in the form of a block diagram.

Dataset description and preprocessing

The stock market dataset [7] from 01-09-2009 to 31-12-2019, i.e 10 years data is preprocessed and a min-max scaler function is used. As the data used has vast differences in the units, it will be normalized using the MinMaxScalar function so that all the values get converted into values ranging between 0 and 1. The missing values will be handled using the ffill function which propagates the previous value to the cell with an unknown value.

Data Split

The data is split as 50% for training and 20% for validation first. Then after finding optimal hyperparameters, 70% is used for training and 30% for testing.

Model building

In this work, python language is used and Google colab is used for training our LSTM model on its GPUs/TPUs. Here two LSTM models are build, i.e vanilla LSTM and stacked LSTM. Vanilla LSTM has one layer architecture with 50 neurons in that layer. The stacked LSTM model has two layers with 64 and 32 neurons in layers 1 and 2 respectively. Dropout of 0.2 is set at both layers. The model is trained on different epochs to find the minimum error. For evaluating error RMSE is used. The two models are then compared using this RMSE value.
As this model is based on Univariate input. 'Close Value' from the historical stock market data is used for the prediction. The window size is set as 60 which means the previous 60 days will be considered for every predicted value while prediction.

Evaluation metrics
Lots of evaluation metrics can be used to estimate the accuracy of a prediction model. Root Mean Squared Error (RMSE) is one of them. It is a performance metric that is widely used for accurate measurement. Here, the original value, and predicted value are denoted by y_i and $\hat{y}_i$, respectively, and n represents the total amount of data. This error is characterized as below

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 8, August 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

5. Results

Analysis of prediction results

After training our LSTM models, the number of epochs has a significant impact on the result of testing. After prediction, in Fig. 11 it can be seen that the lowest RMSE value for Vanilla LSTM is 0.302 which is observed at 1000 epochs. For stacked LSTM, it can be seen in Fig. 12, 0.012 is the lowest RMSE value obtained at 100 epochs. Fig. 13 and Fig. 14 are the results of prediction for 'Close values' of apple stock.

Epochs	RMSE	Time(mins)
10	1.108	2
20	0.992	5
50	0.881	10
100	0.613	16
500	0.4736	33
1000	0.302	66

Fig. 11. Vanilla LSTM model RMSE values

Epochs	RMSE	Time(min)
10	0.606	2.5
20	0.1365	3
50	0.0615	8
100	0.0102	25
500	0.1199	38.6
1000	0.2843	88

Fig. 12 Stacked LSTM model RMSE values

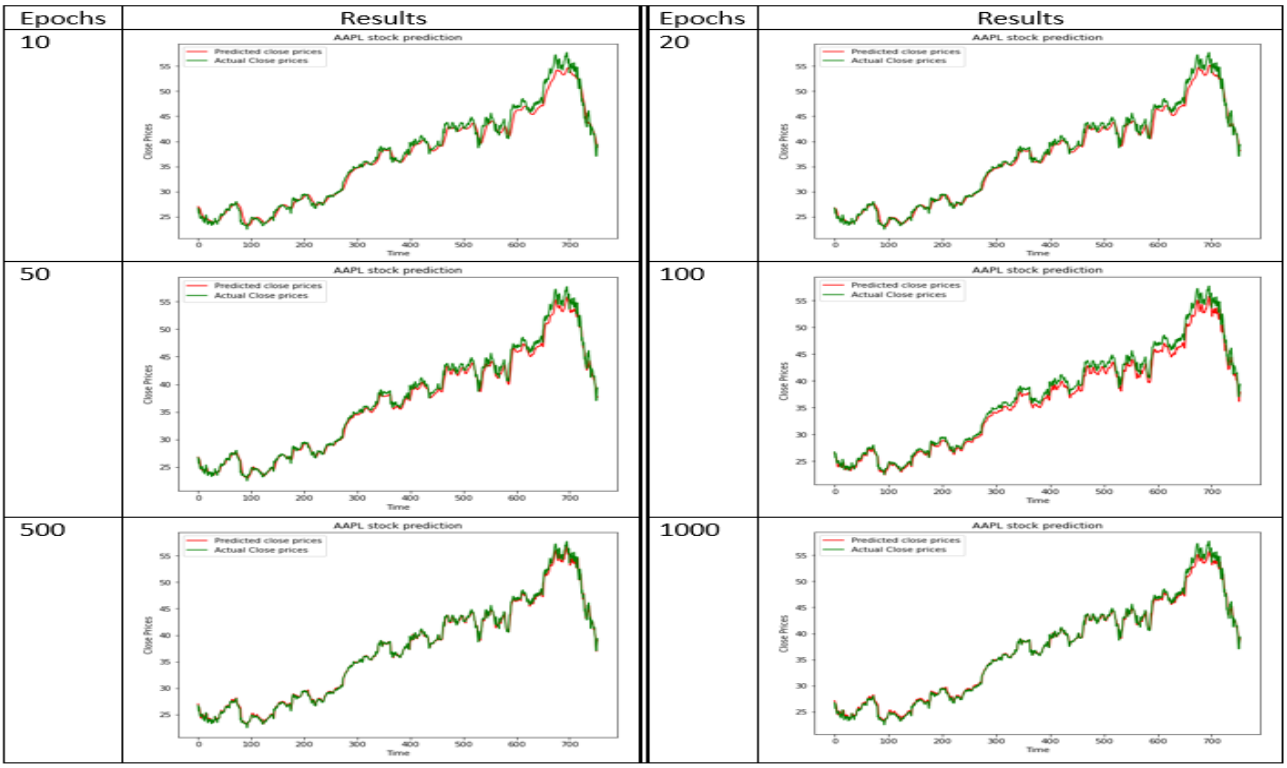

Fig. 13 Prediction results of Vanilla LSTM model

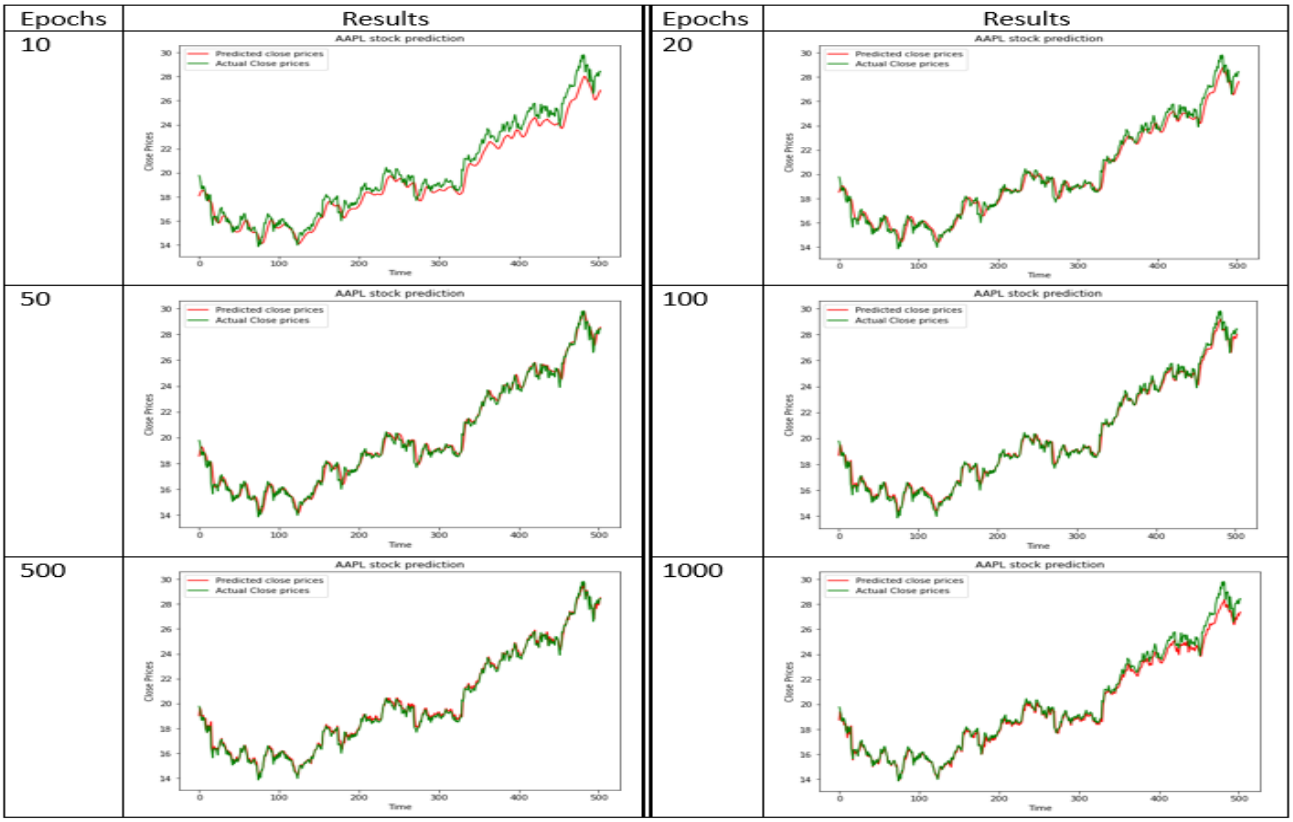

Fig. 14 Prediction Results of Stacked LSTM model

6. Conclusion

Deep Learning algorithms have a considerable influence on modern technology, particularly in the development of distinct time series-based prediction models, as shown in this study. They have the highest level of accuracy when it comes to stock price prediction when compared to other regression models. With correct parameter adjustments, Vanilla LSTM and Stacked LSTM can be employed for stock price prediction. Adjusting these parameters is critical when developing any form of the prediction model, as the accuracy of the prediction is heavily reliant on them. As a result, precise parameter adjustment is also required for both models. In this work, it is seen that stacked LSTM gives more accuracy than vanilla LSTM. This means adding layers to LSTM can give better results when applied to this kind of dataset.

REFERENCES

[1] Deng Fengxin, Wang Hongliang, " Application of LSTM Neural Network in Stock Price Trend Forecast-Based on the Research of Stock Market Data in US and Hong Kong Stock Markets" [J],Financial Economy,2018(14):96-98.

[2] Ma Chaoqun, Wang Xiaofeng, "Forecast of Vegetable Sales Based on LSTM Network Model" [J],Modern Computer (Professional Edition), 2018 (23): 26-30.

[3] Bao W, Yue J, Rao Y, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory" [J], Plos One, 2017,12(7).

[4] Zachary C L, John B, Charles E, " A Critical Review of Recurrent Neural Networks for Sequence Learning", Computer Science,2015.

[5] Hochreiter S , Schmidhuber J.Long short-term memory[J].Neural Computation,1997.

[6] Jingyi Du ,Qingli Liu ,Kang Chen ,Jiacheng Wang, "Forecasting stock prices in two ways based on LSTM neural network", 3rd Information Technology,Networking,Electronic and Automation Control Conference (ITNEC 2019), IEEE, 2019.

[7] Dataset- https://finance.yahoo.com/quote/AAPL/history/

[8] https://towardsdatascience.com/time-series-forecasting-with-deeplearning-andattention-mechanism-2d001fc871fc

[9] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

[11] https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

[12] https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714