International Journal for Modern Trends in Science and Technology, 8(01): 259-264, 2022

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0801045
Available online at: http://www.ijmtst.com/vol8issue01.html

Stock Price Prediction : Recurrent Neural Network in Financial Market

Abdul Fayeem | Ajeet Kumar | Rakshita Sagar | Aditi Aggarwal | Dhyanendra Jain

Dr. Akhilish Das Gupta Institute of Technology & Management, Delhi, India.
Corresponding Author Mail Id: kfaheem119@gmail.com

To Cite this Article
Abdul Fayeem, Ajeet Kumar, Rakshita Sagar, Aditi Aggarwal and Dhyanendra Jain. Stock Price Prediction : Recurrent Neural Network in Financial Market. *International Journal for Modern Trends in Science and Technology* 2022, *8* pp. 259-264. https://doi.org/10.46501/IJMTST0801045

Article Info
Received: 09 December 2021; Accepted: 06 January 2022; Published: 13 January 2022.

ABSTRACT

It is difficult to predict the future price of assets due to the abnormally high volatility of the financialmarket.Financialtime seriesdata ismore complexandprone toerrorsthanotherstatisticaldata.Itshowslong term trends and seasonal variations. Developing more realistic models for estimating and extractingmeaningful statistics from it is a great research endeavor. Due to the nature of data nonlinearity, thetraditional statistical models were not very accurate in terms of financial forecasting. Hence, various softcomputing techniques have been developed to make the models more robust and accurate. This paperaims to build a Deep learning model that can predict the future asset values of provided financial datawith the help of Recurrent Neural Network (RNN), especially(LSTM) Long Short-Term Memory topredictionofstock marketdata.

KEYWORDS: Recurrent Neural Network ,Long Short Term Memory ,Stock Prediction

1. INTRODUCTION

First shall we about a feed forward Network (Bebis&Georgiopoulos, 1994) in Deep learning (Akita et al., 2016) (Jiang, 2021) , (Nabipour et al., 2020) that is used for photo classification so if we've skilled this unique feed ahead network for classifying diverse pictures of animals, now in case you see a photo of a cat it'll perceive that photo and could offer a applicable label to that unique photo, in addition in case you feed in an photo of an elephant it'll offer a applicable label to that unique photo as nicely now in case you be aware the brand new output that we've were given this is classifying an elephant has no relation with the preceding output this is of a cat, or you could say that the output at time (t) is impartial of output at time (t-1) as you could take a look at that there's no relation among the brand new output and the preceding output, so we are able to say that during feed forward networks outputs are impartial to every other, now there are few eventualities, wherein we really need the preceding output to get the brand new output allow us to discuss one such scenario, now what occurs while you examine book you'll recognize that book most effective at the information of your preceding phrases very well, so if we use a feed forward Network and attempt to expect the subsequent phrase in a sentence we can not do that. why we are able to't do that? due to the fact my output will virtually rely upon the preceding outputs however withinside the feed forward Network.

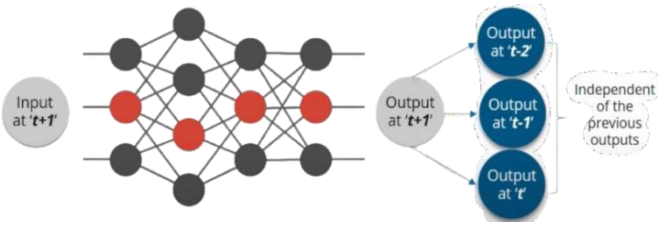

Figure 1 : Representation of feed forward network

Because the new output is unbiased of the preceding output status. output at (t+1) has no relation with output at (t-2) (t-1) and at (t), so essentially we can't use feed-forward networks for predicting the subsequent phrase in a sentence, in addition you could consider many different examples in which we want the preceding output. A few information from the preceding output as a way to infer the new output that is simply one small instance there are numerous different examples that you could consider.In order to resolve this unique hassle we've got input at (t-1) will feed it to our network then we're going to get the output at (t-1) then at the subsequent time stamp this is at time(t), we've got input at time (t) that will be given to a network together with the information from the preceding time step this is (t-1), and that will assist us to get the output at (t). further as output for (t+1) we've got inputs, one is a new input that we provide another facts coming from the preceding timestamps that is (t) with the intention to get the output at time (t+1), further it is able to pass on. so there's a loop in which the information from the preceding timestamp is flowing and that is how we are able to resolve this unique challenge with the help of the idea of recurrent neural networks (Jiang, 2021).

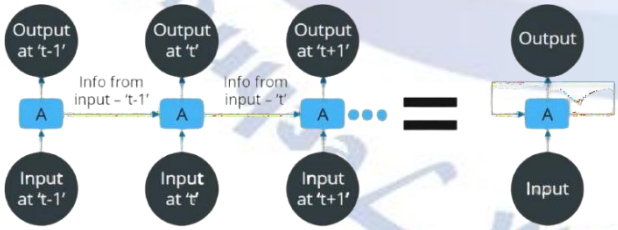

Figure 2 : Representation of Recurrent Neural Network working flow.

1.1.Recurrent Neural Network (Sherstinsky, 2020), additionally called RNNs (Wang, 1993), are a category of neural networks (Huang,et al. 2017), that permit preceding outputs for use as inputs whilst having hidden states (Zaremba,Sutskever&Vinyals, 2014). They are generally as follows:

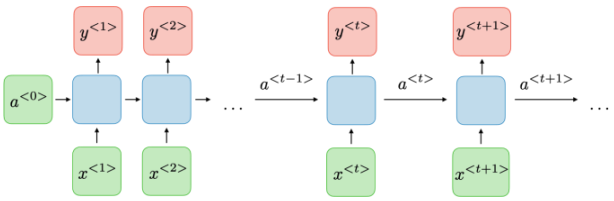

Figure 3 : Recurrent Neural Network (RNNs) Explained.

for on every time stamp (t) the activation function (a <b>) and the output (y <b>) are as follows :

Where Wax, Waa, Wya, ba, are coefficients that proportion temporally and g1, g2 activation functions

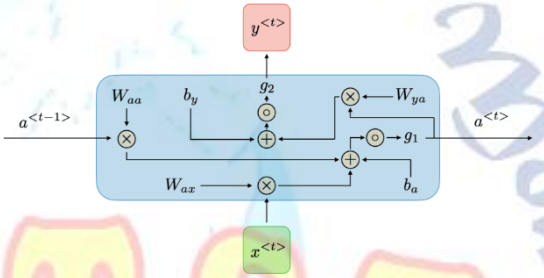

Figure 4 : Block Diagram of RNN working architecture

To train a recurrent neural network, this uses back propagation algorithm for training, but back propagation occurs for each timestamp that is why it's far normally known as back propagation through time ,with back propagation there are convinced issues namely vanishing and exploding gradients.

In handling gradient what takes place whilst you use back propagation (Rumelhart, 1995), you tend to calculate the error (e) that is nothing however the Actual output which you already know and the Model output that you obtain thru a model and the square of that, so that you discern out the error (e). with that error what you do, you tend to find out the change in error(de) with respect to change in weight (dw) or any variable. so change of error (de) with respect to weight multiplied by learning charge(n) will come up with the change in weight (Delta w) then you want to add that change in weight (Delta w) to the old weight (w) to get the new weight. all proper so manifestly what we're seeking to will, we are seeking to lessen the error, so for that we need to discernout what is going to be the change in error (de) , if my variables are modified proper in order that manner we are able to get the

change withinside the variable and upload it to our old variable to get the new variable. now over here what can appear if the value (de/dw) that may be a gradient or you may say the rate of change of error with respect to our variable weight turns into very smaller than one allow it's far like factor zero zerosome thing so in case you multiply that with the learning charge (n) that is actually smaller than one then you get the change of weight (dw) that is negligible.

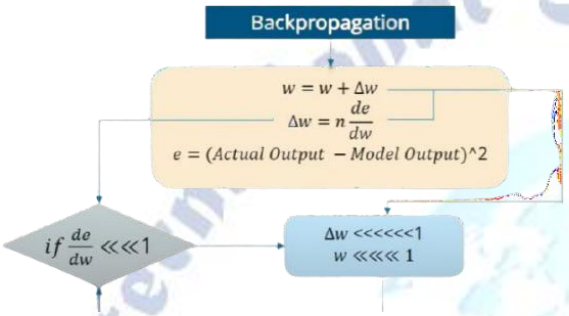

Figure 5 : Flow diagram of Backpropagation.

now bear in mind a state of affairs wherein you want to predict the subsequent phrase in sentence and your sentence is some thing like this "I had been to France" then there are lot of phrases after that few humans talk and then you need to predict what comes after talk, now if I want to try this I need to go back and apprehend the context what's it talking about and this is nothing however your long-term dependencies so what takes place at some stage in long-term dependencies, if this (de/dw) turns into very small then whilst you multiply it with (n) that is again smaller than one you get (dw) which will be very very small on the way to be negligible. so the new manner that you'll get here will be nearly identical to your old weight (w) ,so this new weight will definitely be will always be nearly equal to one old bit there won't be any learning here so this is nothing but your vanishing gradient problem (Hochreiter, 1998).To conquer this vanishing gradient and Exploding Gradients problem, we adopted a technique called (LSTM) **Long Short Term Memory** (Manaswi, 2018).

1.2 Long Short Term Memory (LSTM)

As we recognize how RNN works let us now understand what's LSTM (Greff et al., 2016) or we also can say it as Long Short Term Memory (DiPietro et al., 2020) (Manaswi, 2018). you notice conventional RNNs aren't appropriate at capturing long range dependencies. what i mean to mention right here is that after we have a tendency to work with a completely huge data set and multiple RNN layer, we're on the threat of vanishing gradient problem. while training a very deep neural network gradient or the derivatives decrease exponentially because it propagates down the layer that is referred to as vanishing gradient problem those gradients are actually used to update the weights of a neural network but when the gradients vanish those weights will now no longer get up to date withinside the worst case scenario it'll completely stop the neural network from training. This vanishing gradient problem is a common trouble in very deep neural networks so to conquer this vanishing gradient problem in RNNs Long Short Term Memory was introduced. (Sundermeyer et al., 2012) (Hochreiter&Schmidhuber, 1997), Long Short Term Memory is a amendment to RNNs hidden layer. LSTM is able to remembering RNNs weights and their inputs over a very lengthy time frame in LSTM in addition to the hidden state, cell state is passed right all the way down to the subsequent block the manner LSTM works. It is able to capture long range dependencies this is always it can have memory of previous inputs for extremely prolonged time duration, the manner LSTM cell does that is via way of means of the use of three main gates first one is a Forget gate eliminates the facts this is not beneficial withinside the cell state, then we've Input gate additional information to the cell state is added via way of means of Input gate, and ultimately we've some thing known as as Output gate additional useful information to the cell state is likewise delivered via way of means of an output gate this gating mechanism of LSTM has allowed network to learn the situations for when to forget ignore or maintain information withinside the memory cell.

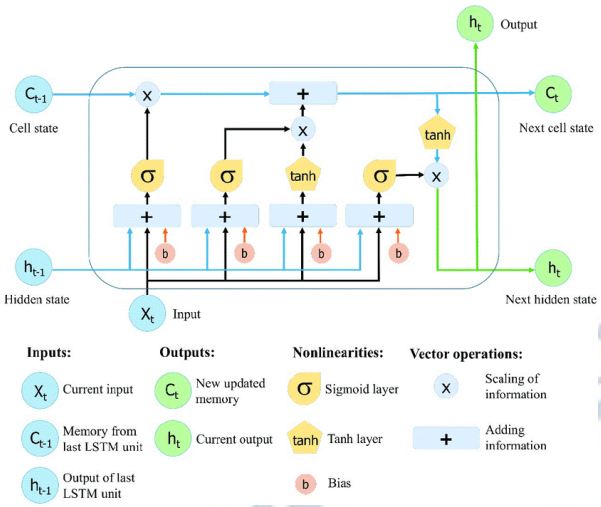

Figure 6 : Flow chart of (LSTM) Long Short Term Memory

2.METHADOLOGY

Lets say 10 days data that I'm taking and I have to predict the data for 11th day. For example (x1, x2, x3, x4, x5, x6, x7, x8, x9, x10) we have taken the 10 days data . Now I'm going to predict the value of 11th day. So we should know that the value of the 11th day is going to be dependent on these previous 10 days values that we are taken. It will not just become 100 or 200 on 11th day ,It will be just dependent on these previous values only. But in this case we are talking about stock market (Chong et al., 2017) (Vargas et al., 2017) (Althelaya et al., 2018), which fluctuate on the basis of external news , reports and performance of companies. We have to take care of external outliers that going to occurs. In this model we are working on same methodology that I described above.

2.1.Proposed work :

1.Starting with importing all the required libraries that we are going to use in model.

2.Then we are scrapping the data from yahoo finance website and defining the start and the end point of dataset that we are going to use in our machine learning model.

3.Now we will be doing resetting the index and dropping the columns from dataset, which are not useful for our model analysis. And selecting the CLOSE price column on that we are going to train our machine learning model.

4.This step is optional: In this model we added couple of features which is Moving Averages. This is a indicator being used in Stock market by stock traders,It help to enhance the chart reading experience for traders. In this model we are taking 100 days Moving Averages and 200 days Moving Averages (Kwok et al., 2009).

5.Now coming to the important part of any machine learning model which Data Splitting. We have split the data into Training and Testing part that we usually do for predictions. So we have split a data in such a manner that Training part is 70% of the data and the rest 30% is for Testing part.

6.Then we have to convert our data into a Scaled Data. For this we are importing MinMax Scaler from Sklearn data preprocessing. And defining the features from 0 to 1. That means all the values from the closing price column will scaled down between 0 to 1. That's the way we provide data to our LSTM model.

7.In next step we have to split our data into x train and y train , So that's why we have use a Time Series Analogy (Faraway & Chatfield, 1998). That value for a particular day or we can say the closing price of a particular day will be dependent on a previous days values. In this model we have defined steps as 100, that means the value for the 101th day will be dependent on a previous 100 days. That's why this previous 100 days values become my x train and the 101th day value will be my y train.

8.Then we have to convert our x train and y train into a numpy arrays.

9.Then we have defined a simple LSTM model, So in this model we Defined 4 layers in LSTM model and at last a Dense layer which connects the all layers together.

10. Now heading towards to finalize our machine learning model , we compiled the model with ADAM Optimizer (Zhang, 2018) and kept the losses as Mean Squared error. Which is used in Time Series Analysis. At last compiled our model for 50 epochs.

```
In [34]: model.compile(optimizer = 'adam', loss = 'mean_squared_error')
         model.fit(x_train, y_train, epochs=50)

Epoch 1/50
63/63 [==============================] - 29s 342ms/step - loss: 0.0424
Epoch 2/50
63/63 [==============================] - 21s 336ms/step - loss: 0.0077
Epoch 3/50
63/63 [==============================] - 22s 348ms/step - loss: 0.0077
Epoch 4/50
63/63 [==============================] - 21s 330ms/step - loss: 0.0065
Epoch 5/50
63/63 [==============================] - 19s 294ms/step - loss: 0.0066
Epoch 6/50
63/63 [==============================] - 21s 333ms/step - loss: 0.0062
Epoch 7/50
63/63 [==============================] - 21s 339ms/step - loss: 0.0057
Epoch 8/50
```

Figure 7 : Running Status , Epochs & Error

11. Now let's have a look of our LSTM Model summary.

```
In [33]: model.summary()
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 100, 50)           10400

 dropout_5 (Dropout)         (None, 100, 50)           0

 lstm_7 (LSTM)               (None, 100, 60)           26640

 dropout_6 (Dropout)         (None, 100, 60)           0

 lstm_8 (LSTM)               (None, 100, 80)           45120

 dropout_7 (Dropout)         (None, 100, 80)           0

 lstm_9 (LSTM)               (None, 120)               96480

 dropout_8 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
_________________________________________________________________
```

Figure 8 : Model summary

3. RESULTS

Now the final step to making predictions with our model we are using our Testing data and fitting Testing data in this trained model in the form of x train and y train as same step as training dataset.

For prediction we are used Dataset (Hiransha et al., 2018) ,Apple Stock and we plotted the Predicted and the original price comparison. For this Machine learning model we have created a web application with the help of streamlit library by python which is used to build webapp applications for ML models.

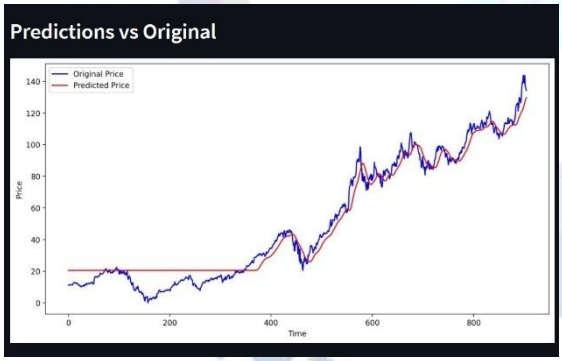

Figure 9 : Final Predicted chart of Nifty 50 index.

4. CONCLUSION

In this model, we proposed the financial data which is collected from different markets and applying algorithms in order to make some useful insights from provided financial data, especially for stocks data. Although the Recurrent Neural Network is seems to be best for Time Series Analysis but in back propagation a vanishing and exploding gradient problem can lead to a bad result. To overcome this gradient problems we introduced LSTM a long short term model which did it best to overcome this gradient problem and lead to better prediction.

REFERENCES

[1] Hochreiter, S., &Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[2] Zaremba, W., Sutskever, I., &Vinyals, O. (2014). Recurrent neural network regularization. arXiv preprint arXiv:1409.2329.

[3] Rumelhart, D. E., Durbin, R., Golden, R., & Chauvin, Y. (1995). Backpropagation: The basic theory. Backpropagation: Theory, architectures and applications, 1-34.

[4] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116.

[5] Bebis, G., &Georgiopoulos, M. (1994). Feed-forward neural networks. IEEE Potentials, 13(4), 27-31.

[6] Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016, June). Deep learning for stock prediction using numerical and textual information. In 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS) (pp. 1-6). IEEE.

[7] Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications, 83, 187-205.

[8] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., &Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

[9] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short- term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[10] Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). NSE stock market prediction using deep-learning models. Procedia computer science, 132, 1351-1362.

[11] Jiang, W. (2021). Applications of deep learning in stock market prediction: recent progress. Expert Systems with Applications, 184, 115537.

[12] Manaswi, N. K. (2018). Rnn and lstm. In Deep Learning with Applications Using Python (pp. 115-126). Apress, Berkeley, CA.

[13] Kwok, N. M., Fang, G., & Ha, Q. P. (2009, November). Moving average-based stock trading rules from particle swarm optimization. In 2009 International Conference on Artificial Intelligence and Computational Intelligence (Vol. 1, pp. 149-153). IEEE.

[14] Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., &Salwana, E. (2020). Deep learning for stock market prediction. Entropy, 22(8), 840.

[15] Jiang, W. (2021). Applications of deep learning in stock market prediction: recent progress. Expert Systems with Applications, 184, 115537.

[16] Vargas, M. R., De Lima, B. S., &Evsukoff, A. G. (2017, June). Deep learning for stock market prediction from financial news

articles. In 2017 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA) (pp. 60-65). IEEE.

[17] Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In Thirteenth annual conference of the international speech communication association.

[18] DiPietro, R., & Hager, G. D. (2020). Deep learning: RNNs and LSTM. In Handbook of medical imagecomputing and computer assisted intervention (pp. 503-519). Academic Press.

[19] Althelaya, K. A., El-Alfy, E. S. M., & Mohammed, S. (2018, April). Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In 2018 9th international conference on information and communication systems (ICICS) (pp. 151-156). IEEE.

[20] Manaswi, N. K. (2018). Rnn and lstm. In Deep Learning with Applications Using Python (pp. 115-126). Apress, Berkeley, CA.

[21] Wang, J. (1993). Analysis and design of a recurrent neural network for linear programming. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40(9), 613-618.

[22] Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS) (pp. 1-2). IEEE.

[23] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[24] Faraway, J., & Chatfield, C. (1998). Time series forecasting with neural networks: a comparative study using the air line data. Journal of the Royal Statistical Society: Series C (Applied Statistics), 47(2), 231-250.