FORECASTING OF BSE SENSEX USING NEURAL NETWORKS

N Rama Chandra*; Dr.M.Ramesh; Prof. M. Bhupathi Naidu***; Dr. M.Venkataramanaiah******

*Research Scholar,
Dept. of Statistics, S.V University,
Tirupati, INDIA

**Dept. of Statistics,
S. V. University, Tirupati, INDIA

***Prof. Directorate of Distance Education,
S.V University, Tirupati, INDIA

****Retired Prof.,
S.V University, Tirupati, INDIA
Email id: ramesh14889@gmail.com

DOI: 10.5958/2249-7137.2022.00881.3

ABSTRACT

Machine learning is the underlying tool, which is making all the above mentioned possible and it is only the beginning. Now we want to see how these machine learning algorithms able to help us in forecasting the time series data. There are varieties of machine learning methods like Support Vector machines, Decision Tree, Random forest, Neural Networks and boosting methods. In this analysis, we adopted Neural Networks to forecast the BSE SENSEX data.

By using this method, we forecast the BSE SENSEX Closing point value and then compare the method with the help of RMSE measure.

KEYWORDS: *Neural Networks (NN), Bombay Stock Exchange (BSE) And Root Mean Square Error (RMSE).*

INTRODUCTION

The current study is focused on financial variables and hence the data considered for empirical analysis is drawn from the financial organizations. The data for the study is collected from Bombay Stock Exchange (BSE) which is one of chief trading center where almost all Indian companies are listed out for trading. Even though, there are large number of financial indices are available in this domain, the planned variable considered for the analysis of the study is BSE SENSEX. SENSEX is the short form of sensitive index and it is one of primary indices of Bombay Stock Exchange. Based on SENSEX, several investors plan their financial investments. Some of the organizations look it as a standard index.

1.0 METHODOLOGY

1.1 NEURAL NETWORKS

In Neural Network, we have different types such as convolutional neural networks (CNN), recurrent neural networks (RNN) and multi-layer perceptrons (MLP). The convolutional neural

networks is mostly used in the image recognition, by using these models the machine learning models can recognize an image and they can even classify the images, for example, if there is some number on an image and image is loaded to the model then the model can identify the numbers. The Recurrent Neural Networks is mostly used for sequentially arranged data points like speech identification, where the machine can identify the words spoken by human beings.

1.1.1 BUILDING RECURRENT NEURAL NETWORK (RNN)

In time series forecasting the correlation between current time period and the previous time period(s) plays a major role, we can say that the correlation is a form of memory that carry forward in the data itself. The architecture of Recurrent Neural Network (RNN) perfectly suits for the sequential data, where it can predict the probability of occurrence of a word in a sequence of words and it can identify the sentiment of a given sentence. Therefore, we can use recurrent neural network to forecast the time series values by using the input time series values.

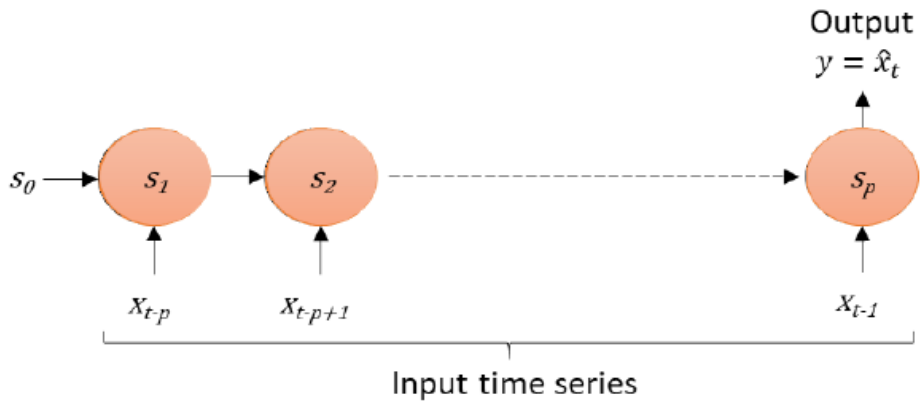

Figure 1.1: Input Time Series for Recurrent Neural Networks

In the figure 1.1, we have x_{t-1}, ...,x_{t-p} time series points and the output from the last point is the forecasted value. The above process is in only one direction, we can create bi-directional which is called Bi-directional recurrent neural networks and there can be multiple directions, which is called deep recurrent neural networks. In figures (1.2 & 1.3), we can see the approaches for both these process.

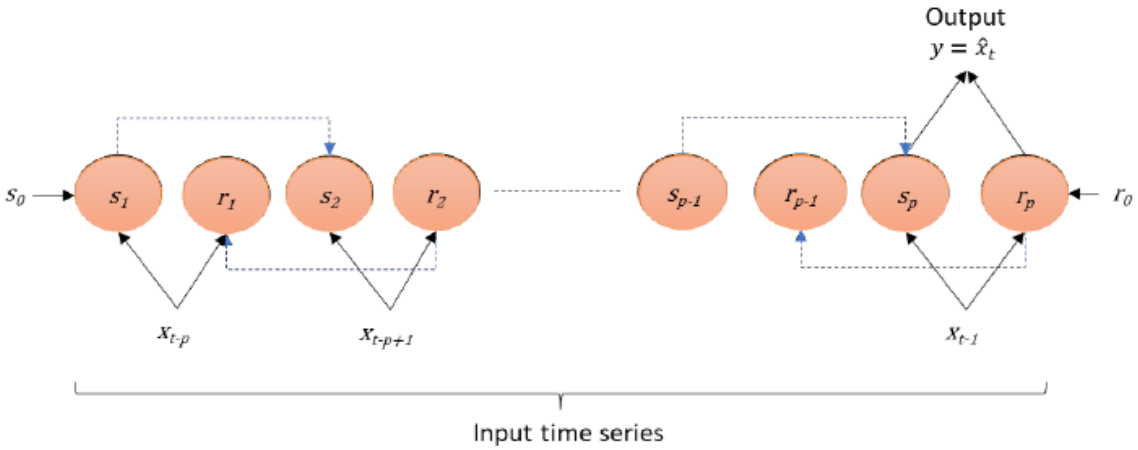

Figure 1.2: Bi-directional Recurrent Neural Networks

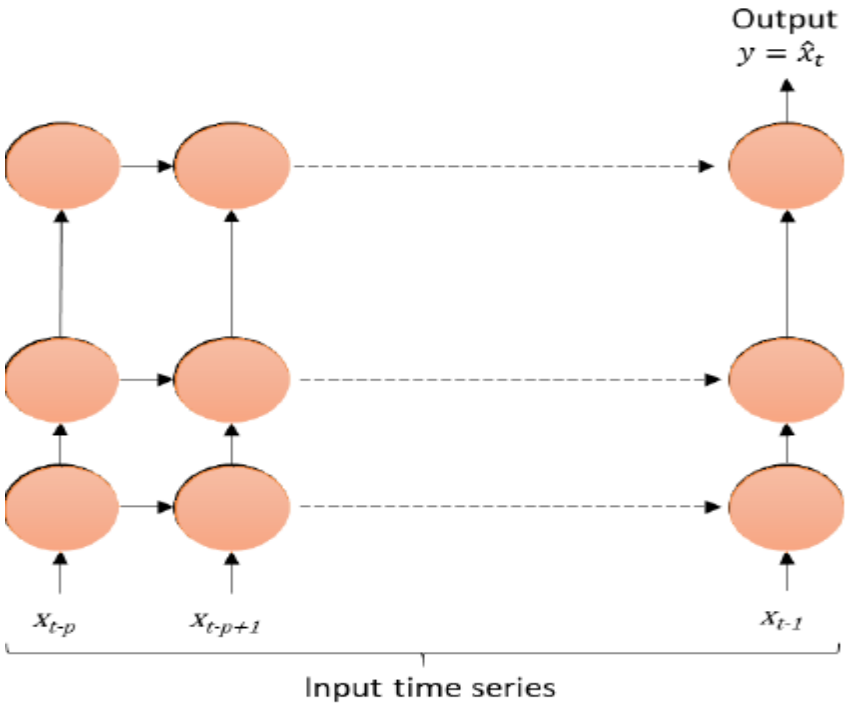

Figure 1.3: Deep Recurrent Neural Network Process

Understanding the computational background of the recurrent neural networks is extremely difficult, in the figure 1.4, we can see the path behind the recurrent neural network.

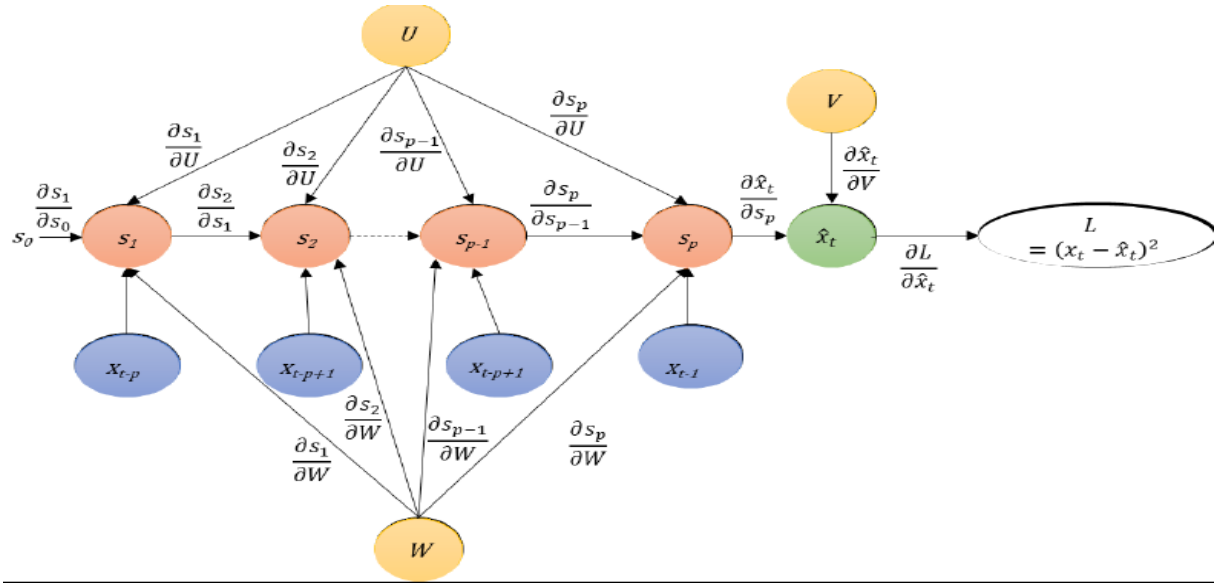

Figure 1.4: Recurrent Neural Network Parameter Estimation Process

Figure 1.4 is showing the computation equation of deep recurrent neural networks.

1.2 RESULTS AND DISCUSSION

Now we will apply the recurrent neural networks for the BSE data and forecast the BSE closing point value. The figure 1.5 shows the movements of BSE closing points from January 2000 to February 2018.

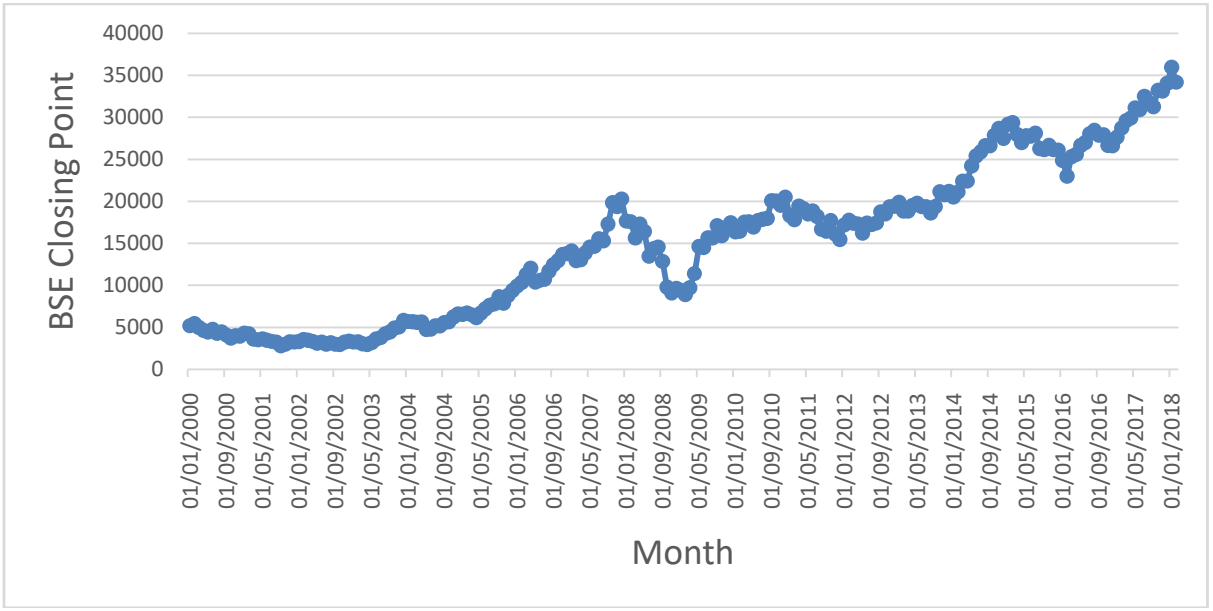

Figure 1.5: Time Series Plot for BSE SENSEX Closing Points

We have to reframe our data to apply the Recurrent Neural Networks for forecasting the BSE SENSEX closing point values; below, we have sample observations from the re-framed data. Most of the machine learning models works better if the underlying data is standardized or within a certain limits. In most cases, researchers use the (-1, 1) limits and some may use (-3, 3) as well. Here we are using the (0, 1) limits. To get the data into these limits we can use the below mentioned formula.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad \ldots\ldots\ldots\ldots \quad (1.1)$$

After changing the scale, the graph obtained is shown in figure 1.6.

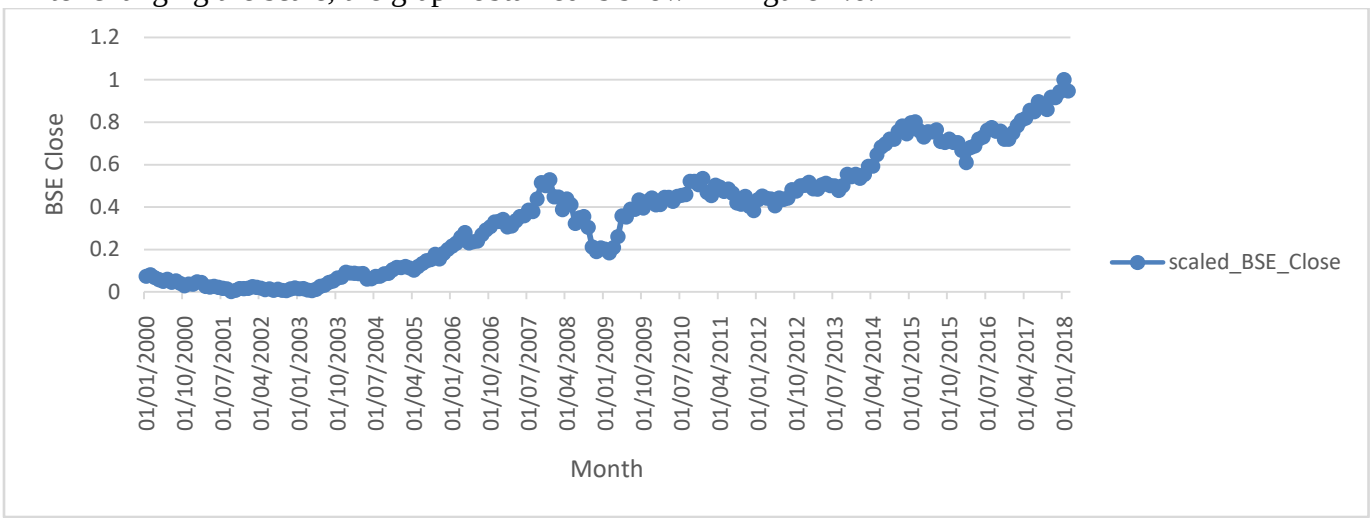

Figure 1.6: BSE SENSEX Closing Points with Change of Scaled Values

In figure 1.6, if we take a close look at Y-axis the scale is only between 0 and 1.While developing the machine learning or any other statistical models, it is a good practice to divide the data into three parts, viz., training data, validation of data and out of time data. In this process we

divide the time of study into two, the first period is being the time of the study and the other, out of study time period. Further we divide the time of the study period into two random samples, training data and validation of data (Usually 70:30), we develop the model on the training data and we check the accuracy of the model on the validation of data.

Usually the machine learning model captures most of the patterns of the training data and in most of the cases the patterns hold true for the entire study period time as our validation of data is also from the same time period the model can predict well on the validation of time period as well. In the analytical world we usually call it as problem of over fitting, to check the over fitting problem we need to check the accuracy of the model, we need to apply the model on the data where it has no idea, which is our out of time data.

So, in our time series data, to overcome the problem of over fitting we divided the data into two parts one is training period and the other is forecasted time period.

In order to build the recurrent neural network we need to reshape the data, in traditional time series methods the moving average or the auto regressive method automatically collects data from the previous periods. However, in the machine learning model, especially the neural network methods, we need to reshape the data as per the neural network frame work. In the below table (4.1), we can see the output of the reshaped data. Using the python software, we reshaped the data as below;

```
Shape of train arrays : (196, 8) (196, 1)
Shape of Test Arrays : (14, 8) (14,1)
```

As we can see, we have two arrays in the data, one with 196 rows and 8 columns, another with 196 observations. Here it is nothing but the time series data with 8 periods lag. Here we have data from January, 2000 to December, 2016 total 204 observations as we take 8 as the lag period the training will start from September, 2000 take all the previous 8 months as the lag period. In the table 4.1, we can see that to predict the value for the September, 2000 the machine learning model will consider the previous 8 periods as base which is from January, 2000 to August, 2000 and to predict the value for October, 2000 the method will consider the values from February, 2000 to October, 2000.

TABLE 1.1: REAL DATA BEFORE CHANGING THE SCALE

Lag Periods								Current period
1/1/2000	2/1/2000	3/1/2000	4/1/2000	5/1/2000	6/1/2000	7/1/2000	8/1/2000	9/1/2000
2/1/2000	3/1/2000	4/1/2000	5/1/2000	6/1/2000	7/1/2000	8/1/2000	9/1/2000	10/1/2000
3/1/2000	4/1/2000	5/1/2000	6/1/2000	7/1/2000	8/1/2000	9/1/2000	10/1/2000	11/1/2000

The real data, after changing the scale as per the formula (1.1) is shown in table 1.1 (a).

Table 1.1(a): Real Data after changing the scale

Lag Period 8	Lag Period 7	Lag Period 6	Lag Period 5	Lag Period 4	Lag Period 3	Lag Period 2	Lag Period 1	Current Period
0.072200394	0.079490442	0.066046881	0.055679022	0.048924364	0.058430473	0.044286834	0.050242479	0.038571586
0.079490442	0.066046881	0.055679022	0.048924364	0.058430473	0.044286834	0.050242479	0.038571586	0.027129026
0.066046881	0.055679022	0.048924364	0.058430473	0.044286834	0.050242479	0.038571586	0.027129026	0.035784845
0.055679022	0.048924364	0.058430473	0.044286834	0.050242479	0.038571586	0.027129026	0.035784845	0.035004533
0.048924364	0.058430473	0.044286834	0.050242479	0.038571586	0.027129026	0.035784845	0.035004533	0.045700263
0.058430473	0.044286834	0.050242479	0.038571586	0.027129026	0.035784845	0.035004533	0.045700263	0.043296891
0.044286834	0.050242479	0.038571586	0.027129026	0.035784845	0.035004533	0.045700263	0.043296891	0.023912465
0.050242479	0.038571586	0.027129026	0.035784845	0.035004533	0.045700263	0.043296891	0.023912465	0.021341991
0.038571586	0.027129026	0.035784845	0.035004533	0.045700263	0.043296891	0.023912465	0.021341991	0.024742847
0.027129026	0.035784845	0.035004533	0.045700263	0.043296891	0.023912465	0.021341991	0.024742847	0.019460436

In neural network we have input layer, output layer and a bunch of hidden layers, the forecasting power of the model may increase with the number of hidden layers but it will also cause the problem of over fitting, we can use the option of dropout to reduce this problem. In this neural network, the parameter estimation will work on the Mean square error with the linear activation function. In the following python code, we can see that we have one input layer, 3 hidden layers, and one output layer. Here we used dense as 32 for the first hidden layer which will allow the input layer to create 32 neurons.

```
input_layer = Input(shape=(8,), dtype='float32')
```

```
dense1 = Dense(32, activation='linear')(input_layer)
dense2 = Dense(16, activation='linear')(dense1)
dense3 = Dense(16, activation='linear')(dense2)
```

```
dropout_layer = Dropout(0.2)(dense3)
```

```
output_layer = Dense(1, activation='linear')(dropout_layer)
```

```
ts_model = Model(inputs=input_layer, outputs=output_layer)
ts_model.compile(loss='mean_squared_error', optimizer='adam')
ts_model.summary()
```

In the table 1.2, we can observe the total number of parameters that the model generates at each step, in total the model has to estimate 1,105 parameters, which should minimize the mean squared error.

TABLE 1.2: RECURRENT NEURAL NETWORKS PARAMETER ESTIMATION

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 8)	0
dense_1 (Dense)	(None, 32)	288
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 16)	272
dropout_1 (Dropout)	(None, 16)	0
dense_4 (Dense)	(None, 1)	17

Total params: 1,105
Trainable params: 1,105
Non-trainable params: 0

To obtain the accurate forecasts, we need to use the appropriate number for lag period. Since there is no method to identify the lag period, we carried out the analysis on different numbers for lag period we mentioned two of them with its RMSE values in table 4.3

TABLE 1.3: RMSE VALUES FOR DIFFERENT LAG PERIODS

Lag Period	RMSE Value
6	1530.212232
8	937.1616

In figure 1.7, we can find the actual verses forecasted values when we consider lag period as 6. Its RMSE value is 1530.212232

Figure 1.7: Actual values Vs Forecasted values of BSE SENSEX Closing Points by using RNN with lag period 6

We tried with another lag period, which is 8, its RMSE value is 937.1616, and this lag period 8 gave us minimum RMSE value compared to the other lag period. In figure 4.8, we have the comparison between the actual values and forecasted values for all time periods when we consider lag period as 8.

Figure 1.8: Actual values Vs Forecasted values of BSE SENSEX Closing Points by using RNN with lag period 8

Figure 1.9 shows the actual values and forecasted values only for the forecasted period

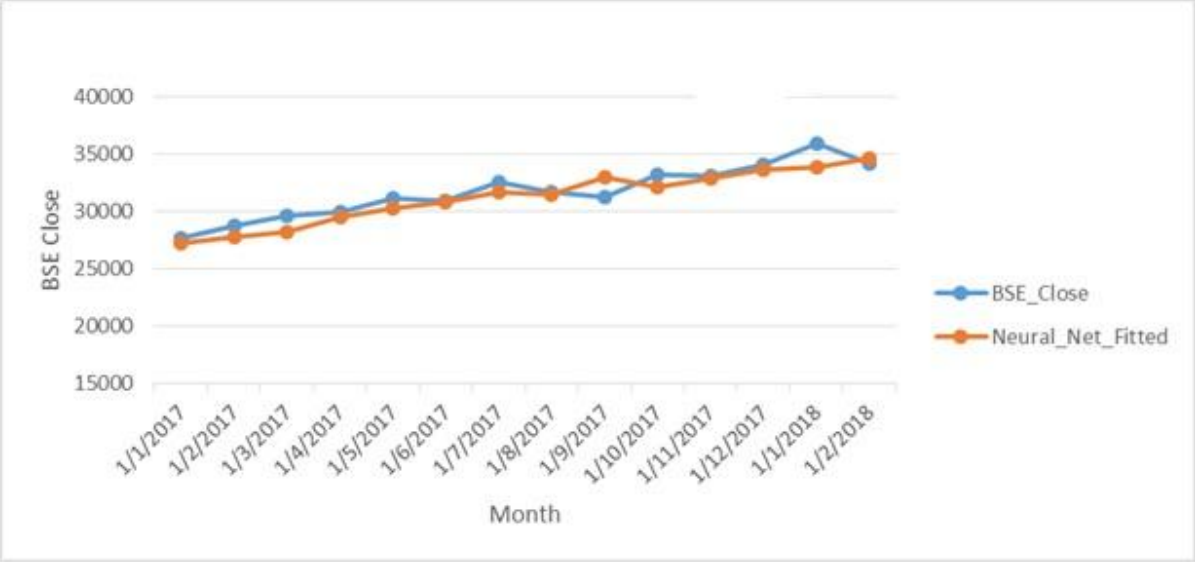

Figure 1.9: Actual Vs Forecasted values of BSE SENSEX Closing Points only for forecasted period by using RNN with lag period 8

In the beginning we changed the scale of the original values and here we rescaled the values into the original values and presented in the table 1.4 along with their forecasted values.

TABLE 1.4: RESHAPED SCALED DATA INTO ORIGINAL DATA AND THEIR FORECASTED VALUES

Month	BSE Close	Forecasted Values
1/1/2017	27655.96	28896.146
1/2/2017	28743.32	28845.771
1/3/2017	29620.5	29258.129
1/4/2017	29918.4	29136.092
1/5/2017	31145.8	29844.55
1/6/2017	30921.61	30706.77
1/7/2017	32514.94	31305.54
1/8/2017	31730.49	32726.578
1/9/2017	31283.72	32482.123
1/10/2017	33213.13	33179.062
1/11/2017	33149.35	33394.195
1/12/2017	34056.83	34206.89
1/1/2018	35965.02	34245.695
1/2/2018	34184.04	35338.945

The overall comparison between the traditional methods and neural networks basing on RMSE values can be observed in the table 1.5.

TABLE 1.5: THE RMSE VALUES FOR DIFFERENT FORECASTING METHODS

Method	RMSE
SES Method	5,758
Holt's Additive	1,831
Holt's Multiplicative	1,326
ARIMA	1,103
RNN with single Time Series Variable	937

1.3 CONCLUSION

In this Paper, we consider neural network method for forecasting of BSE SENSEX Closing Points. In Neural Networks we have variety of forecasting methods like forward feeding, recurrent neural network, conventional neural network etc., we used recurrent neural network method to forecast the BSE closing point, with single time series variable. We forecasted the BSE closing point values for the period from January 2017 to February 2018 and got the RMSE value of 937, which is less than RMSE Values of the considered traditional forecasting methods. So, we can select this method to forecast the future values of BSE SENSEX Closing Point.

REFERENCES

1. Abraham, B. and Ledolter, A. (1983): Statistical Methods for Forecasting, New York: Wiley.

2. Azoff. E. M. (1994): Neural Network Time Series Forecasting of Financial Market, John Wiley & Sons Ltd.

3. Bagherifard, K., Nilashi, M., Ibrahim, O., and Janahmadi, N., (2012): Comparative Study of Artificial Neural Network and ARIMA Models in Predicting Exchange Rate, Research Journal of Applied Sciences, Engineering and Technology, Vol.4, pp. 4397-4403.

4. Brockwell, Peter J. and Davis, Richard A. (2002): Introduction to Time Series and Forecasting, 2nd Edition, Springer-Verlag.

5. Chavan, P.S., and Patil, S.T. (2013): Parameters for Stock Market Prediction, International Journal of Computer Technology & Applications, Vol. 4 Issue .2, pp. 337-340.

6. De Livera, A.M., Hyndman, R.J. and Snyder, R.D. (2011): Forecasting Time Series with Complex Seasonal Patterns Using Exponential Smoothing. Journal of the American Statistical Association, Vol. 106, pp. 1513-1527.

7. Holt, Charles C., 2004: Forecasting seasonal and trends by exponentially weighted moving averages," International Journal of Forecasting, Elsevier, vol. 20(1), pp. 5-10.

8. Jin, R., Wang, S., Yan, F., and J. Zhu (2015): The Application of ARIMA Model in 2014 Shanghai Composite Stock Price Index, Science Journal of Applied Mathematics and Statistics, Vol.3, Issue.4, pp. 199-203.

9. Junior, P. R., Salomon, F. L. R., de Oliveira Pamplona, E. (2014): Arima: An applied time series forecasting model for the bovespa stock index. Applied Mathematics, Vol. 5, pp. 3383-3391.

10. Keith Ord (2004): Charles Holt's report on exponential weighted moving average: An introduction and appreciations, International Journal of Forecasting, February 2004, Vol.20, Issue.1, pp.1– 3.

11. Konarasinghe, W. G. S., Abeynayake, N. R., and L. H. P. Gunaratne (2015): ARIMA Models on Forecasting Sri Lankan Share Market Returns International Journal of Novel Research in Physics Chemistry & Mathematics, Vol.2, Issue.1, pp. 6-12.

12. Krishna Reddy, M. and Kalyani, D. (2005): Applications of Neural Networks in Time Series Forecasting, International Journal of Management and Systems, Vol.21, pp. 53-64.

13. Kuo –Cheng, O Joung K, T Jung LC (2012): Time Series and Neural Network Forecast of Daily Stock Prices, Investment Management & Financial Innovations, Vol.9, pp. 32-54.

14. M. Ramesh, K. Radhaiah and Prof. Venkataramanaiah (2016): Forecasting of gold prices using Holt's exponential smoothing method, An International Multidisciplinary Research Journal, Vol.6, Issue.7, pp. 136-143.

15. M. Ramesh, M. Sunitha and Prof. Venkataramanaiah (2016): Holt's exponential smoothing method for Forecasting Silver prices, South Asian Journal of Marketing &

Management Research, Vol.6, Issue.8, pp. 1-12.

16. Panichkitosolkul, W. (2006): A Comparison of Forecasting Method of Daily Jewellery Gold Prices: Holt's Forecast Method, Box- Jenkins Method and Combined Forecast Method, Naresuan University Journal, Vol.14, Issue.2, pp. 9-16.

17. Saini, S., Singh, N.P., and R.R. Laxmi (2006): Application of ARIMA Models in Forecasting Stock Prices, International Journal of Mathematics and Computer Applications Reaserach, Vol.6, Issue.6, pp. 1-10.

18. Zhang, Z., Li, M., Bai, R. (2012): An Integrated Model for Stock Price Prediction based on SVM and ARMA, Journal of Theoretical and Applied Information Technology, Vol. 44, Issue.1, pp. 51-54.