Scientific Research Publishing

Theoretical Economics Letters, 2019, 9, 511-529
http://www.scirp.org/journal/tel
ISSN Online: 2162-2086
ISSN Print: 2162-2078

Neural Nets for Stock Indices: Investigating Effect of Change in Hyperparameters

Sonali Agarwal[1], Jwaad Akhtar Khan[2]

[1]Department of Management, Netaji Subhash University of Technology (Formerly Netaji Subhash Institute of Technology) Sector 3, Dwarka, New Delhi, India
[2]School of Management and Business Studies, Jamia Hamdard University, Hamdard Nagar, New Delhi, Delhi, India
Email: sonali1600@yahoo.in, jwaad.a.khan@gmail.com

How to cite this paper: Agarwal, S. and Khan, J.A. (2019) Neural Nets for Stock Indices: Investigating Effect of Change in Hyperparameters. *Theoretical Economics Letters*, **9**, 511-529.
https://doi.org/10.4236/tel.2019.93036

Received: January 27, 2019
Accepted: March 12, 2019
Published: March 15, 2019

Abstract

Artificial neural networks have seen an outburst of interest in past decade. There has been an increasing use of ANNs in prediction based studies owing to their huge performance accuracy. They have been successfully applied across various domains like medicine, geology, finance, physics, engineering etc. The system of neural nets witnesses rise in complexity with increase in number of layers and number of neurons and possesses the capacity to solve intricate problems. The researchers, world over, consider the neural network with three layers (input, hidden and output) a universal approximator of functions as it has given outstanding results in data validation, price forecasting, sales forecasting, customer research etc. over the years. In most of the previous studies, either a standard ANN model has been taken or a default model has been tested using various softwares. But as we understand, a lot of hit and trial should be done by altering the hyperparameters to get the best performance model. In our study we attempt to prove the same point and try to find the best model for our data set wherein we predict the BSE sensex closing price of the next day using previous day data (high price, low price, open price, close price and trade volume). We use deep neural networks with backpropagation and have altered the hyperparameters: number of nodes in hidden layers, the activation function of hidden layers, Number of epochs, the batch size and hence the iterations in each epoch. The model performance was measured with the help of root mean square error on test set of the model. We are able to bring out the differences of tuning of hyperparameter and ultimately find the best predictor model for BSE sensex close value.

Keywords

Deep Neural Networks, Multi-Layer Perceptron, Feed Forward Neural Network, Prediction Models, Hyperparameters, Backpropagation, Activation

Function, Hidden Layer, RBF Neural Network, Epoch, Iteration, Batch Size, Stock Market, BSE Sensex

1. Introduction

Of late, neural networks have been very popular in the prediction based studies. They have been successful across various domain areas like medicine, geology, finance, physics, engineering etc. The system of neural nets becomes more complex with increasing number of neurons and hidden layers. But at the same time, these complex models also enhance the prediction ability of complex problems which are otherwise difficult or unsatisfactory. Till now, a three layered neural network has been christened to be a universal approximator of functions and has been used successfully in data validation, price forecasting, sales forecasting, customer research etc. Neural networks are a relatively new concept that has emerged from the field of Artificial intelligence and is now getting used universally in almost all fields owing to their high performance rates.

One of the decade-long paradox in finance and investment is the "prediction of stock market" and that too with full accuracy. There have been a lot of propositions by various renowned researchers on this topic and a lot of models with various combinations of theories have been introduced, yet the puzzle stands. No model has been found to be fully accurate and reliability of models has been a big concern. Sometimes it has been seen that researchers got satisfied with the model that gave a good result and failed to test alternatives, which if considered could have given better results. This was the biggest gap that we found in previous literature/research. Thus in our study, we try to propose machine learning models with varied alternatives and test them. The effect of change of hyperparameters on the model can drastically affect the model performance. We could not trace any significant previous literature on hyperparameter testing and this itself speaks for the uniqueness of our work. By the end of the study, we were able to conclude the best model with high predictability and very less error. This model if used by investors to predict the future sensex closing value, can help them make good profits or at least prevent the losses from unanticipated rupee depreciation.

1.1. Deep Neural Networks

Feed forward neural networks are the simplest and the basic types of artificial neural networks. Here the connections between nodes do not form a cycle, rather, the information travels only in one direction which is forward from input layer to the output layer. A single layer perceptron uses step function as activation function and uses delta rule for training of neurons. Its drawback is that it cannot learn a XOR function. But an improvement of it, called the multilayer perceptron (MLP) is capable of processing XOR function and computing a con-

tinuous output. Here the activation function is commonly logistic function (also known as a sigmoid function).

$$f(x)=\frac{1}{\left(1+e^{-x}\right)}$$

Since the sigmoid function has a continuous derivative, it allows backpropagation in multilayer perceptrons.

$$f'(x)=f(x)\left(1-f(x)\right)$$

Multilayer perceptron is in fact capable of producing any possible Boolean function. It also satisfies the universal approximation theorem. Thus we also use it in our research. When there are multiple hidden layers, it is called a Deep neural network. More layers mean more processing time but sometimes better results. Large data sets can be churned with efficiency using Deep neural networks (Figure 1).

1.2. Back Propagation

It is a popular method which helps in training of neural networks. Backpropagation optimizes the weights in multiple back passes and helps the network of neurons to correctly map the given inputs to their outputs. For illustration let's consider a neural network with one input layer, one hidden layer and one output layer. Here we are talking of supervised learning, and therefore have the data of inputs and their target outputs with us. The whole process starts with forward feeding of input/output data to the neural network input layer. As the input passes

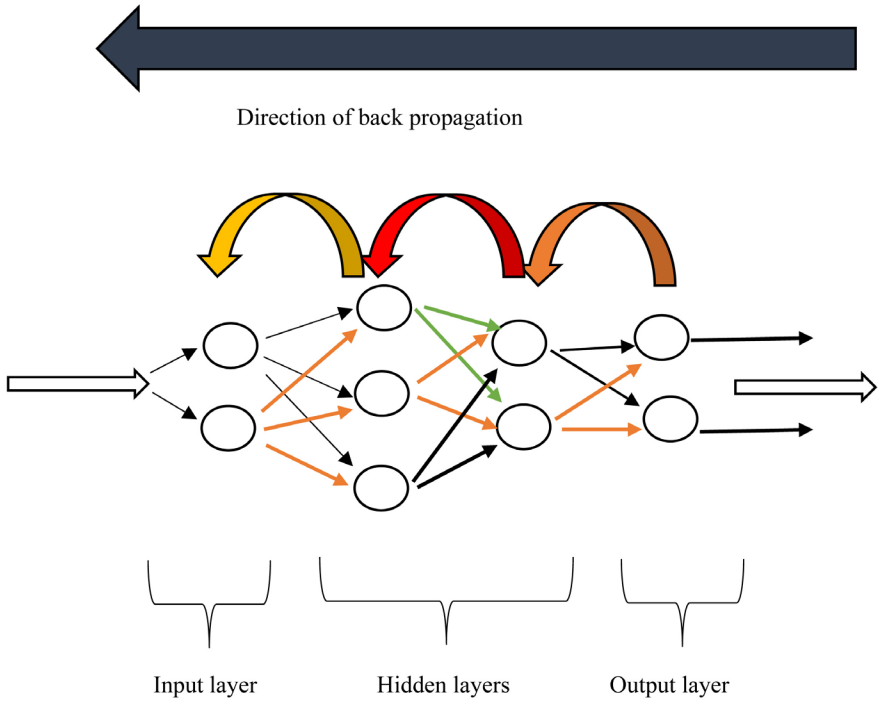

Figure 1. Structure of a deep neural network. (Source: Researcher's own work)

through the hidden layer, it gets crushed with the activation function of the layer. The compressed output of the hidden layer passes as input into the output layer. Here it gets crushed by the activation function of this layer. The output of this crushing gives the output of the forward pass of inputs through the select neural network. This output is compared to the target output that we want. Depending on the difference between the two, the error is backpassed through the network, starting from the output layer. The weights in the network are updated, so that the output from the neural network gets closer to the target output. The delta rule is used for error backpropagation.

For a neuron "i", with activation function $f(x)$, the delta rule (gradient descent) for i's j^{th} weight w_{ij} is given as follows

$$\Delta w_{ij} = l\left(t_i - y_i\right) f'\left(h_i\right) x_j$$

where l is learning rate

$f(x)$ is the activation function of neuron "i" in question

t_i is the target output

y_i is the produced output

x_j is the j^{th} input

h_i is the weighted sum of neuron's inputs

Once the error reaches the input layer, the weights of the network have got updated. The output from the network is then again compared to the target output. This process goes on till the error is decreased substantially. There can be various ways to terminate training of the network e.g. when the learning rate decreases beyond a predefined limit, or after a predefined number of epochs, or when the relative change in training error falls a defined limit etc.

Different termination ways of the training algorithm and different settings of hyperparameters can give different results differing in accuracy and efficiency.

1.3. Hyperparameter

A hyperparameter is a special class of parameter whose value we set before the beginning of learning process. The value of other parameters is otherwise derived from training.

1.4. Activation Functions

Different activation functions can be used in deep neural networks. Different activation functions can be used in different layers (*i.e.* in hidden layers and output layers).

Unipolar logistic function

$$f\left(x\right) = \frac{1}{1+e^{-l\,x}}$$

This function gives the output between 0 and 1.

Bipolar logistic function

$$f\left(x\right) = \frac{2}{1+e^{-l\,x}} - 1$$

This function gives the output between −1 and 1.

Hyperbolic Tangent function

$$f(x)=\frac{e^{lx}-e^{-lx}}{e^{lx}+e^{-lx}}$$

This function gives the output between −1 and 1.

Radial basis function

$$f(x)=\frac{1}{\sqrt{2\pi}}\frac{e^{-(x-\mu)^2/2\sigma^2}}{\sigma}$$

This function gives the output between 0 and $\frac{1}{\sigma\sqrt{2\pi}}$.

In the algorithm that we used to train our neural networks, we set aside 20% data set for crossvalidation step to measure the error of each iteration and help in gradient descent of error. Test error and the training errors are calculated separately in each constructed neural network.

We calculate SSE *i.e.* sum of square error and RE *i.e.* relative error for both sets. SSE gives an indication of the RMSE (root mean square error) which is by far the most reliable method to measure performance of a neural network. The lesser the error, the better the network.

This research paper has been organized in sections, where Section 1 introduces the title and the subject under consideration in this research, Section 2 talks about previous works done in the similar areas, the need for the study and the research gap are explained in Section 3 under research objectives. Section 4 gives a detailed structure of methodology used in this research followed by Section 5, which explains all the different models with the variations of hyperparameters and the results of all these variations. At the end of Section 5, the best fit model for our research BSE Sensex is described. Finally, Section 6 concludes the paper.

2. Literature Review

Some noteworthy researches in the area of financial modelling using machine learning have been done over past few decades. Few of them are worth mentioning. A pioneering work was done by Kimoto *et al.* [1] wherein they applied modular neural networks to the price indices data of Tokyo stock exchange and developed a prediction system for best time of stock buying and selling and achieved accurate predictions. The simulation showed excellent profits. The research done by Ghiassi and Saidane [2] was commendable since they designed a new model of ANN where in they used the entire data set simulataneously for model parameter estimation. The model was appraised using marketing data set and compared it with traditional feed forward methods. The new model was found to perform better. Ghiassi *et al.* [3] compared the traditional iterative back propagation feed forward method of time series forecasting with the dynamic model of neural network and established the supremacy of the latter method. One of the highly acclaimed research was performed by Chang *et al.* [4]. They

developed an integrated system for stock forecasting in which neural network, case based reasoning and dynamic time windows were combined. The prediction of sell/buy deciding points was found to be better than with any of the three methods used alone. Hamzacebi *et al.* [5] compared two methods (direct and iterative) of artificial neural network for time series forecasting. In iterative method one period value is forecasted from the past period one, and then this value is used to predict the next period value. In the direct method all the values of successive periods can be predicted in one go. The researchers compared the two methods using grey relational analysis and found that direct method was better than the iterative method. An innovative empirical study was attempted by Cheng *et al.* [6] wherein they used fundamental and technical analysis and intergrated them with artificial neural network system and set theory to develop market timing investment strategy model. Forecasting accuracy and returns from investments were used for evaluating the model. Liao and Wang [7] studied the fluctuations of Chinese stock Index and make an improvised forecasting neural network model by introducing stochastic time effective function. They suggest that the closer is the time of the past data is to the current time, the stronger is its effect on the prediction model. The model is also appraised by different parameters of volatility. In another research by Guresen *et al.* [8], the researchers tried to cut through traditional linear and nonlinear approaches to forecast stock market rates and analysed three new models: Multi-layer perceptron (MLP), Dynamic artificial neural network (DAN) and a Hybrid nuerual network. The Mean square error used for appraisal of models showed that the MLP model gave the best predictions when used on the same data set. Moghaddam *et al.* [9] investigated the stock forecasting ability of artificial neural network using NASDAQ stock exchange. Two types of input sets four prior days and nine prior days were used, although both methods were found to be equally meritorious.

In most of the studies, either a standard ANN model has been taken or a default model has been tested using various softwares. But as we understand, a lot of hit and trial should be done by altering the hyperparameters to get the best performance model. In our study we attempt to prove the same point and try to find the best model for our data set.

3. Research Objectives

In this research, we wanted to see the effect of change of some hyperparameters on the model's prediction ability and efficiency. The testing is useful as it points out the effect of taking some hyperparameters as default and getting the results without realizing that the model could be tuned for further better results.

4. Research Methodology

We took the daily data of BSE Sensex from yahoo finance website for a time period 1 January 2014 to 31 December 2018. The daily data contained volume

traded, the high price, low price, close price and open price. Raw data was cleaned for any missing values and standardized using Z scores.

$$\text{Z score} = \frac{\text{Input-mean}}{\text{variance}}$$

A total of 1231 readings (observations) were retrieved for analysis using deep neural networks. Cases with user missing values on factors and categorical dependent variables were excluded during analysis.

We used SPSS for analysing the data and constructing different networks. The input or independent variables consisted the high price, low price, open price, close price and the volume traded. The dependent variable or the output value was the closing value of index (labelled closenext here) on the successive day. We set the data partitions as 60% training set, 20% validation set and another 20% as test set. We used batch training for each epoch so as to see the effect of change in iterations on the prediction capacity. The training momentum was set at default value of 0.9. The maximum training time was set to 15 minutes (for the worst scenario if the termination criteria is not reached). The stopping criteria was 6 consecutive training steps with no change in training error or relative change of 0.0001 in training error achieved. Both training and test data were used to compute prediction errors.

The learning rate was set at 0.4 and the lower boundary of learning rate was fixed to 0.001. Gradient descent was used for backpropagation.

The hyperparameters we changed in different models were:

- The number of hidden layers
- Number of nodes in hidden layers
- The activation function of hidden layers
- Number of epochs
- The batch size and hence the iterations in each epoch

The model performance was measured with the help of root mean square error on test set of the model. Both the training and test data sets were used for computing prediction error.

Error Calculation

In the algorithm that we used to train our neural networks, we set aside 20% data set for crossvalidation step to measure the error of each iteration and help in gradient descent of error. Test error and the training errors are calculated separately in each constructed neural network.

We calculate SSE *i.e.* sum of square error and RE *i.e.* relative error for both sets. SSE gives an indication of the RMSE (root mean square error) which is by far the most reliable method to measure performance of a neural network. The lesser the error in the test set, the better the network. As a rule of thumb, if the training error is more, we increase the number of neurons in the hidden layer or the number of hidden layers. If the training error is satisfactory, but test error is more, we presume that the training has led to over-fitting, and therefore we reduce the number of neurons in the hidden layer or the number of hidden layers.

5. Analysis and Results

Case 1. Effect of Change in number of hidden layers and the activation function of hidden layers

In our first variation, we changed the number of hidden layers to 1 and 2. The number of nodes in the single hidden layer were taken as 3 while in the model with two hidden layers, the first layer had 3 nodes and the second had 2 nodes. The number of layers was restricted to 2 as on further increasing, the test error was getting very high. Also, for both the model types (two variations in number of hidden layers) we tested two activation functions for the layers, first one was the sigmoid function and the second one was the hyperbolic tangent function. The performance metrics of the (four total) models is shown in Table 1.

The errors of prediction are elaborated graphically in Figure 10. Figures 2-5 below show the regression results of predicted and actual test set values for

Table 1. Performance metrics of variation in number of hidden layers and its activation functions.

	Number of hidden layers (activation function of hidden layers is sigmoid)		Number of hidden layers (activation function of hidden layers is hyperbolic tangent)	
	1 (hidden nodes = 3)	2 (hidden nodes in first layer =3, hidden nodes in second layer = 2)	1 (hidden nodes = 3)	2 (hidden nodes in first layer =3, hidden nodes in second layer = 2)
Train SSE	1.266	1.964	25.509	1.286
Train RE	0.004	0.005	0.070	0.004
Training time taken	0:00:00.41	0:00:00.58	0:00:00.10	0:00:00.62
Test SSE	0.647	0.761	7.065	1.198
Test RE	0.006	0.007	0.070	0.012
Holdout RE	0.008	0.009	0.090	0.010

SSE = sum of square error, RE = Relative error (Source: Reasearcher's analysis of data).

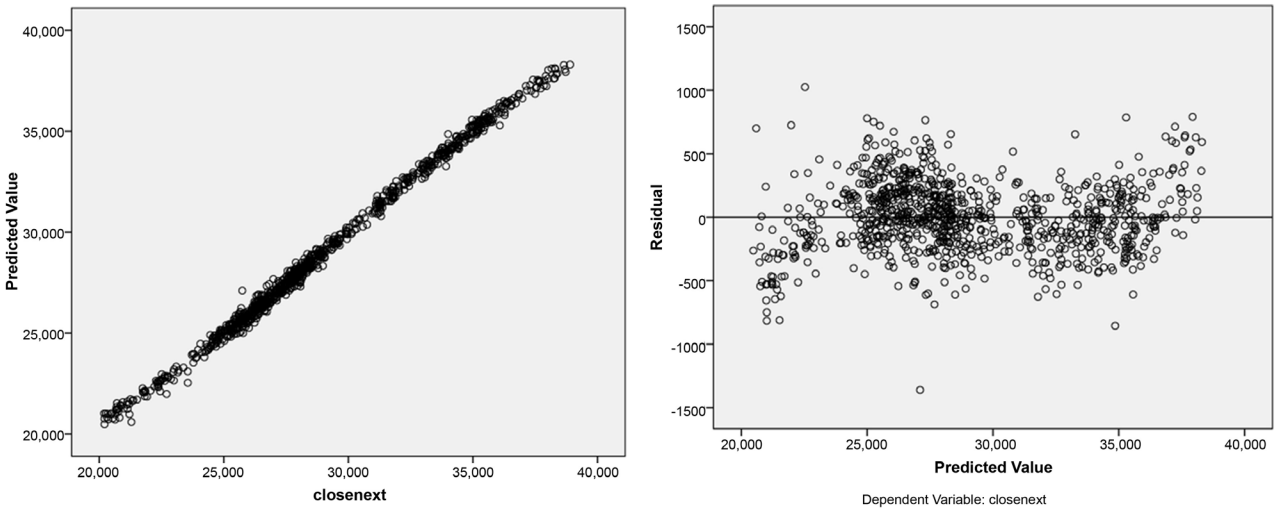

Figure 2. MLP FFN model with Hidden (3) sigmoid: Predicted test value vs. actual test labels. (Source: Reasearcher's analysis of data)

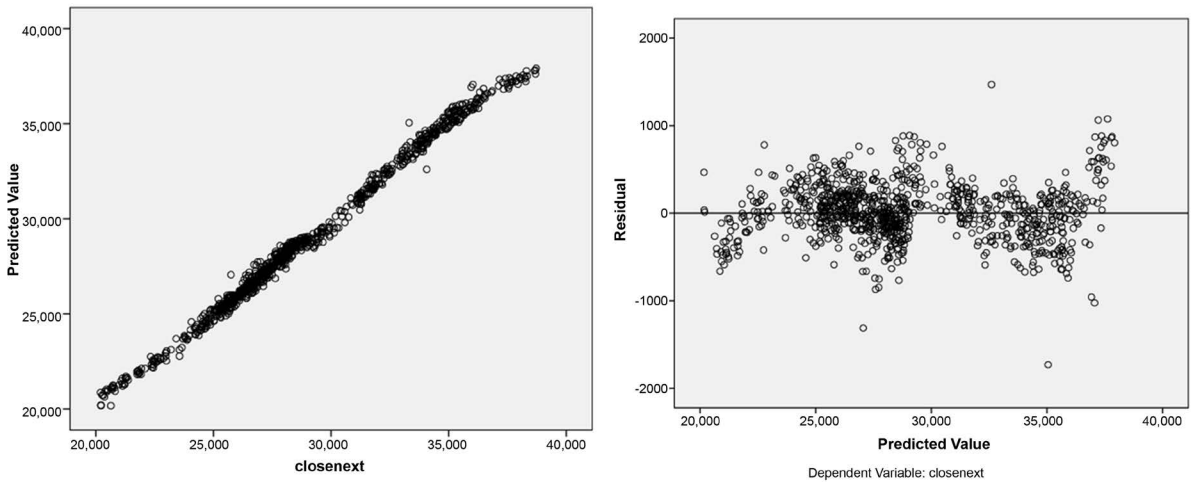

Figure 3. MLP FFN model with Hidden (3, 2) sigmoid: Predicted test value vs. actual test labels. (Source: Reasearcher's analysis of data)

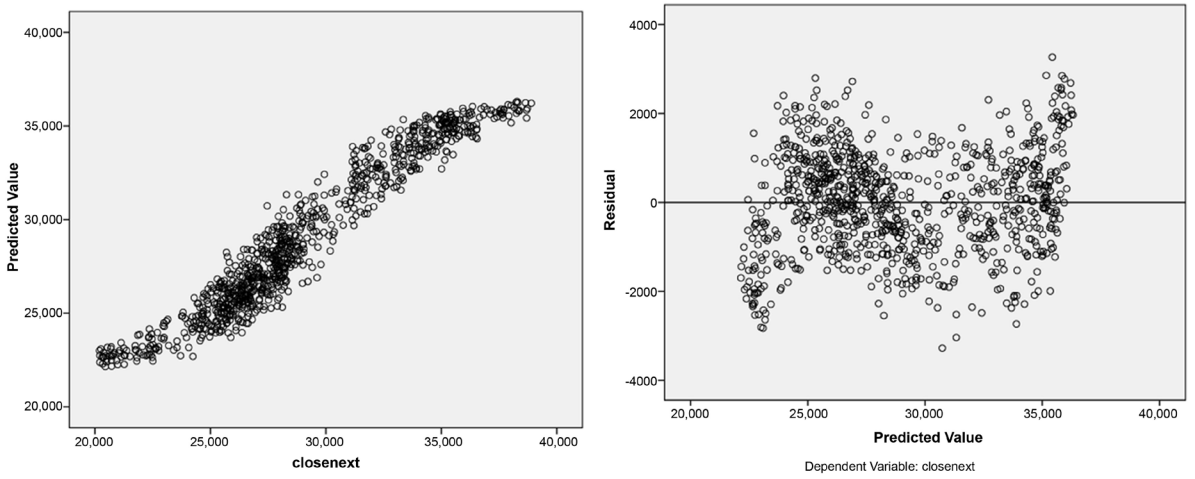

Figure 4. MLP FFN model with Hidden (3) hyperbolic tangent: Predicted test value vs. actual test labels. (Source: Reasearcher's analysis of data)

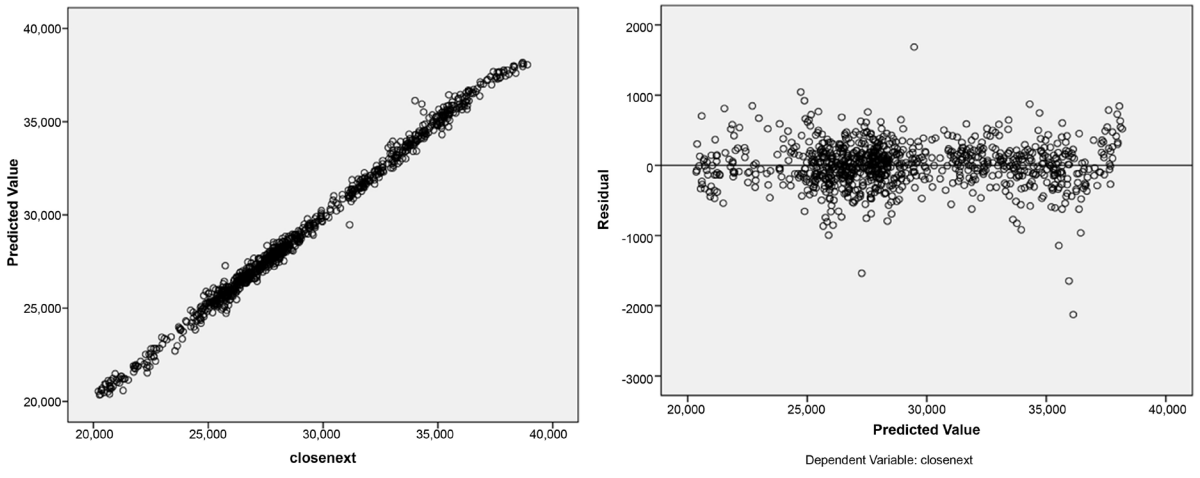

Figure 5. MLP FFN model with Hidden (3, 2) hyperbolic tangent: Predicted test value vs. actual test labels. (Source: Reasearcher's analysis of data)

the four models taken in our case 1. The more clean the regression line, the better the results. Along with this, the figures on the right side show the residual distribution with predicted value. The more centered and closer the residuals, the better is the model.

The graphs shown in Figures 6-9 below show the hits and miss of the actual values and the predicted values for the four models of case 1. The more the visibility of blue line in the graph, the more the misses in the prediction.

From Figure 6, it can be seen that most of the predicted values coincide with the actual values. A few mismatches in between Jan 2014 and March 2014, around August 2018 etc. can be easily identified from the graph plot. There are very few mismatches and outliers. The training time from the data Table 1

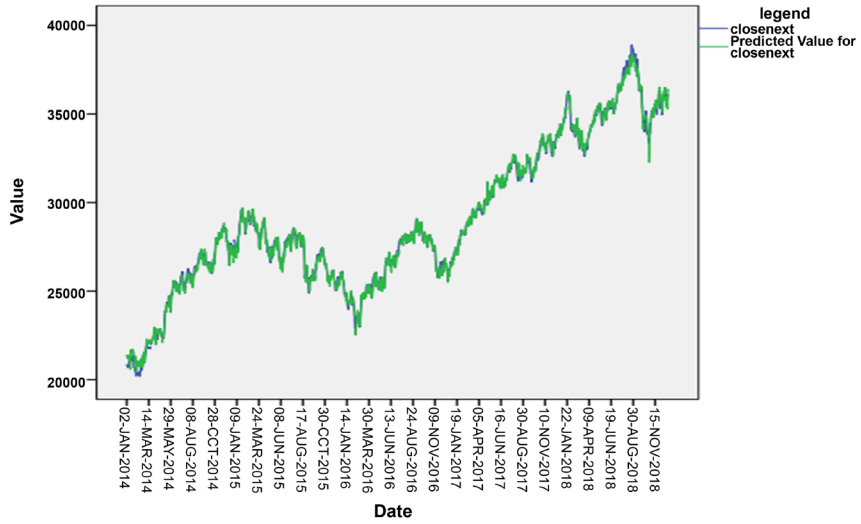

Figure 6. Hidden (3), sigmoid: graph showing actual data values and predicted values. (Source: Reasearcher's analysis of data)

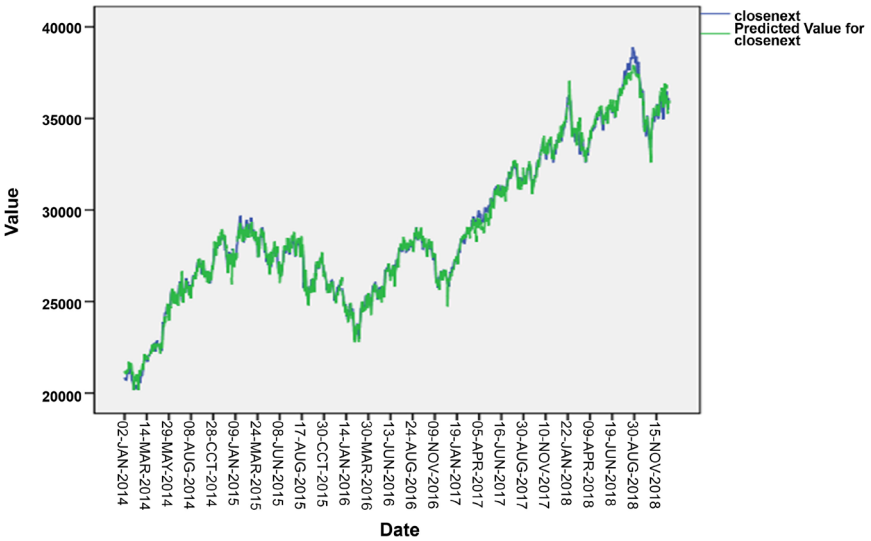

Figure 7. Hidden (3, 2) sigmoid: graph showing actual data values and predicted values. (Source: Reasearcher's analysis of data)

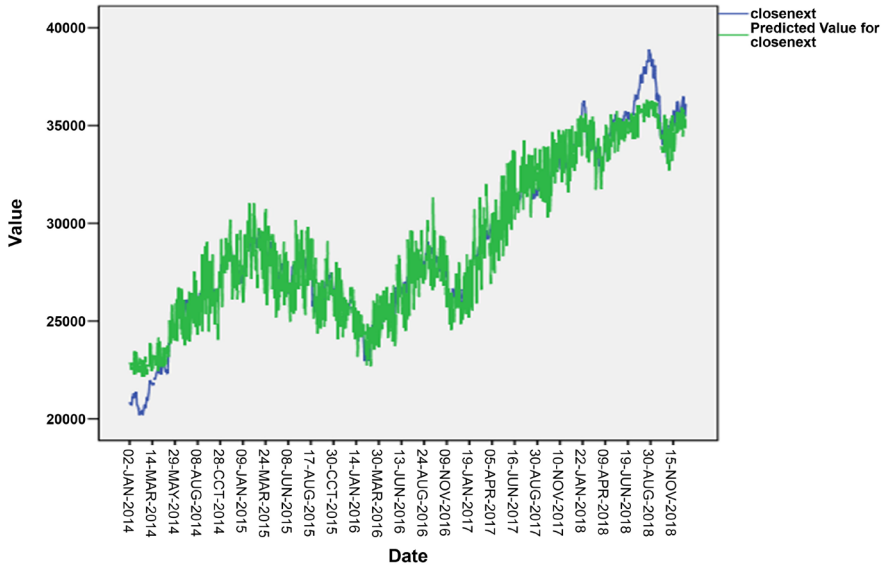

Figure 8. Hidden (3) hyperbolic tangent: graph showing actual data values and predicted values. (Source: Reasearcher's analysis of data)

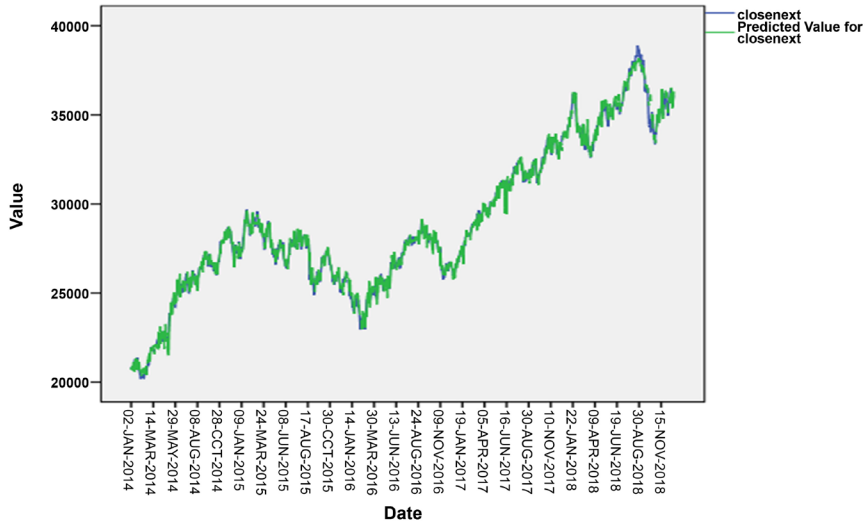

Figure 9. Hidden (3, 2) hyperbolic tangent: graph showing actual data values and predicted values. (Source: Reasearcher's analysis of data)

can also be seen to be low. Overall it is a good model with less error in testing.

It can be seen from Figure 7 that the predicted values and the actual values of closing sensex coincide for most of the study period. Also from Table 1, it can be deduced that both the SSE and RE for both training and testing are more than the previous model. Also, this model has taken more time for training than the previous one with 3 hidden sigmoid neurons. The mismatched areas can be prominently seen in blue colour in the graph (Jan 2014-March 2014, March 2015, April 2017-July 2017, around August 2018 and November 2018).

This model with 3 hidden hyperbolic tangent neurons shows a lot of deviation

between the actual and the predicted closing sensex values. The error rate in both training and testing is high. The disagreement patches are also large and clear (seen in big blue colour lines around January 2014-May 2014, June 2018-Nov 2018 etc.). Also from the graph is messy since the scatter plot shown in Figure 4 shows huge scatter between residuals and predictions. Thus the model is not appropriate for the prediction in current research. Table 1 shows that this model has high SSE in training process. This means that there is requirement to increase the number of hidden layers, or number of hidden neurons or both. To test this we check another model seen in Figure 5 & Figure 9.

When we see Figure 9 along with Figure 8, we can see the difference. There is a lot of improvement in the predictability of model with two hidden hyperbolic tangent layers (3, 2). The training and test error show drastic reduction from the previous model where there was only one hyperbolic tangent layer. Also, the mismatches seen are less (less blue visible lines in above graph). The most prominent deviation seen is from June 2018 to Nov 2018. Though this model is appraised to be good, but the training time taken is more.

As can be seen from the Figure 10, both the test and train error are low in ANN with sigmoid function activated hidden layer. Also, the model with sigmoid function gives best result with one hidden layer with three hidden neurons. The training time in 3 unit single sigmoid activated layer model has less training time of 0.41 seconds with good performance.

Case 2. Effect of change of training batch size

The batch size used for one iteration was varied to see the effect on model performance. We took the number of records as 10, 20, 30 and 50 for a composition of a batch in successive variations of model. The number of units were 192 and the number of hidden layers was 1 (taken as default setting for case 2 variation). The number of nodes in hidden layer was 3. The activation function used was sigmoid. The output activation function was linear.

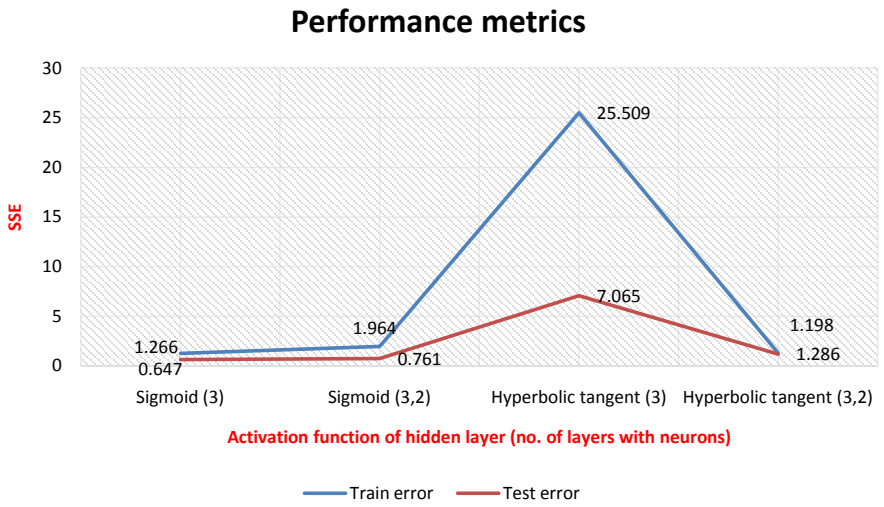

Figure 10. Effect of change in number of hidden layers and change in hidden layer activation function. (Source: Reasearcher's analysis of data)

The results of the performance of models with different batch size iterations is explained through graph in Figure 11.

The batch size was restricted to 50 and no further increase was reported as the error was increasing at exponential rates.

From Figure 11, it can be seen that the least error for both training set and the test set is in 10 records. Also from Table 2, it can be seen that the training time of 0.41 seconds is also comparable with 20 and 30 batch size training time. Hence 10 records is the best size to train the current data set.

Case 3. Effect of change in number of nodes in one hidden layer

In this case, we varied the number of nodes (neurons) in the hidden layer. The number of units were 192 and the number of hidden layers was 1 (taken as default for case 3 variations). The activation function of hidden layer was sigmoid. The output activation was linear. The hidden nodes varied being 3, 10, 20 and 30 in different models tested. On further increase in number of nodes, the error increased immensely.

The performance results of case 3 are explained graphically in Figure 12.

Table 2. Performance metrics of variation in batch size used for training.

	Batch size			
	10	20	30	50
Train SSE	1.266	1.468	2.211	8.118
Train RE	0.004	0.004	0.006	0.022
Training time taken	0:00:00.41	0:00:00.52	0:00:00.32	0:00:00.19
Test SSE	0.647	0.717	0.628	2.343
Test RE	0.006	0.007	0.007	0.022
Holdout RE	0.008	0.007	0.011	0.024

SSE = sum of square error, RE = Relative error (Source: Reasearcher's analysis of data).

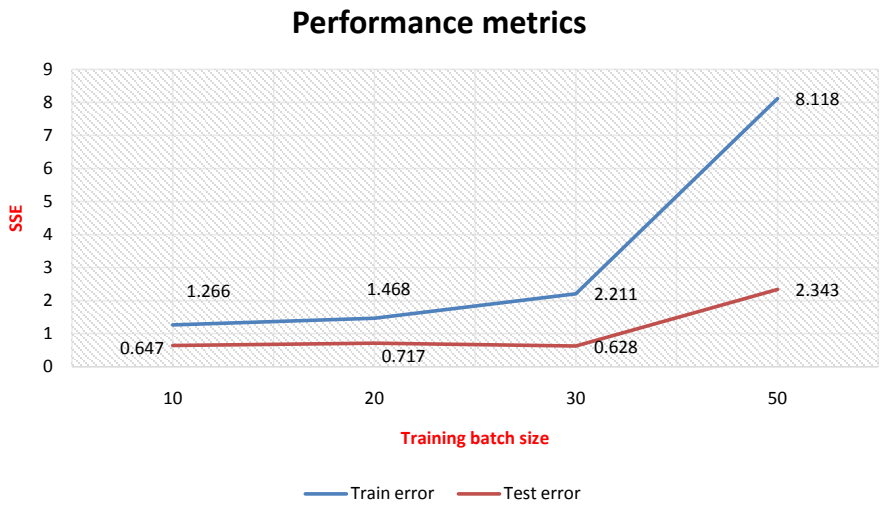

Figure 11. Effect of change of training batch size (no. of records). (Source: Reasearcher's analysis of data)

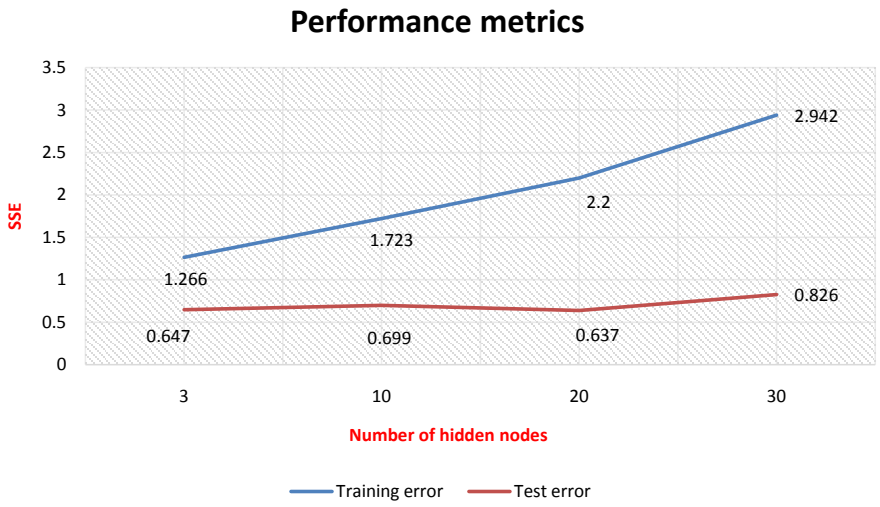

Figure 12. Effect of change in no. of hidden units (nodes) in the single hidden layer. (Source: Reasearcher's analysis of data)

From Figure 12 it can be seen that the training error increases with the number of increasing nodes in the hidden layer. The test set error remains more or less same when the no. of hidden units is 3, 10 and 20. It increase when the no. of nodes is increased to 30. Thus the best model is one with 3 hidden units (nodes) where both the test and train errors are less. Also, from Table 3 it can be seen that the training time is also least when the number of nodes is 3.

Case 4. Effect of change in number of training epochs

In case 4, we attempted to study the effect of change in number of training epochs. One epoch is defined as one pass of the complete data set through the neural network. The number of units were 192 and the number of hidden layers was 1 (taken as default for case 4 variation analysis). The number of nodes in hidden layer was 3. The activation function used was sigmoid. The output activation was linear. We tested a varied number of epochs (starting from 10, 20, … upto 100). There was change in error at each change and also the training time varied across all cases (Table 4).

The performance of case 4 variations is explained further through graph in Figure 13.

From Figure 13 it can be seen that both training and test error are less when the number of epochs used for training are 10, 75 and 80. But the least error is when the number of epochs used is 10. Also the time taken for training is less when seen along with the error rate. Thus the best model for our data set in this research is the one using 10 epochs for training. This is suggested keeping in mind both the model error and the training time taken.

Case 5. Using radial basis function

We tried to use radial basis function network in place of multilayer perceptron feed forward model to check if it gives a better result. In radial basis function, the number of hidden layers is always 1. Hidden layer's number of nodes is varied. The activation function of hidden layer is softmax function. We varied the

Table 3. Performance metrics of variation in number of hidden layer nodes.

	Number of nodes			
	3	10	20	30
Train SSE	1.266	1.723	2.200	2.942
Train RE	0.004	0.005	0.006	0.008
Training time taken	0:00:00.41	0:00:00.76	0:00:01.49	0:00:03.06
Test SSE	0.647	0.699	0.637	0.826
Test RE	0.006	0.007	0.008	0.007
Holdout RE	0.008	0.008	0.006	0.007

SSE = sum of square error, RE = Relative error (Source: Reasearcher's analysis of data).

Table 4. Performance metrics of variation in number of training epochs.

	Number of epochs							
	10	20	50	75	76	77	80	100
Train SSE	1.554	8.641	10.912	1.547	4.298	6.491	1.725	5.042
Train RE	0.004	0.023	0.030	0.004	0.012	0.018	0.005	0.014
Training time taken	0:00:00.37	0:00:00.16	0:00:00.15	0:00:00.56	0:00:00.30	0:00:00.22	0:00:00.60	0:00:00.17
Test SSE	0.718	2.238	3.444	0.788	1.679	2.667	0.920	2.314
Test RE	0.007	0.023	0.036	0.007	0.017	0.024	0.009	0.021
Holdout RE	0.007	0.022	0.029	0.009	0.014	0.026	0.011	0.020

SSE = sum of square error, RE = Relative error (Source: Reasearcher's analysis of data).

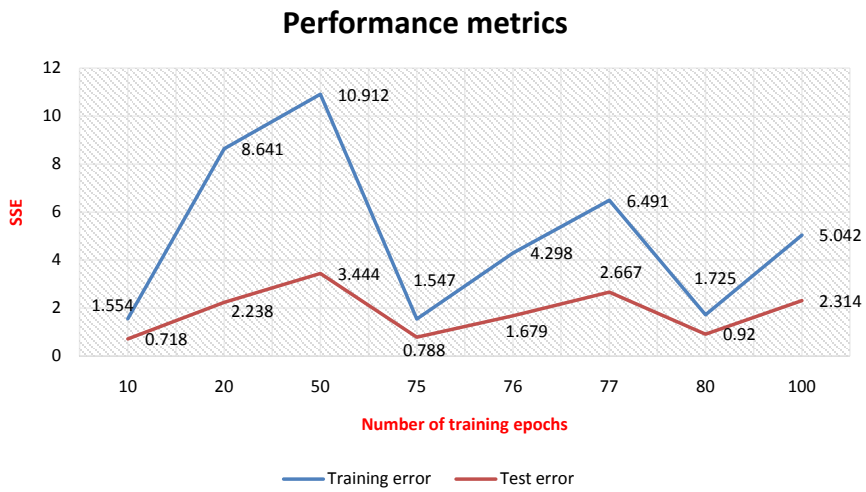

Figure 13. Effect of change in no. of training epochs. (Source: Reasearcher's analysis of data)

number of hidden units (nodes) in the single hidden layer from 10, 20, up to 50. Further increase was giving high errors. The performance statistics are reported in Table 5.

The results of variations done in case 5 are explained through graph in Figure 14.

Table 5. Performance metrics of variation in number of nodes in hidden layer when RBF function model is used.

	Number of hidden units in the hidden layer							
	10	20	29	30	31	32	35	50
Train SSE	6.906	2.262	2.401	2.030	2.489	2.354	2.498	2.555
Train RE	0.019	0.006	0.007	0.006	0.007	0.006	0.007	0.007
Test SSE	1.868	0.587	0.817	0.494	0.746	0.803	0.623	0.842
Test RE	0.019	0.005	0.007	0.006	0.007	0.007	0.005	0.008
Holdout RE	0.016	0.005	0.005	0.008	0.007	0.006	0.006	0.008

SSE = sum of square error, RE = Relative error (Source: Reasearcher's analysis of data).

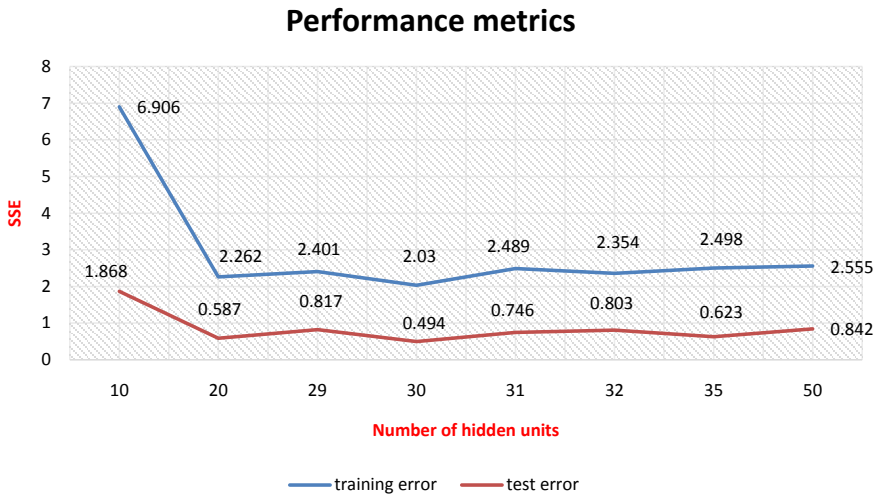

Figure 14. Effect of changing no. of hidden units in radial basis function model. (Source: Reasearcher's analysis of data)

As we can see from Figure 14, the training and testing error co-vary as the no. of hidden units is increased from 20 to 50. The highest error is when the number of units is 10. If we compare these results with our best fit MLP FFN (one hidden layer, with three nodes, sigmoid activation function, 10 record iteration, 10 epochs for training), the training error is high in all cases of RBF network. The test error is although comparable in few of the RBF network cases, overall we can say that a MLP FFN is best suited for BSE Sensex prediction with both training and test errors at their minimum. They show the best performance.

The best fit model for BSE Sensex prediction

Based on the above results from Cases 1 - 5 (represented in Tables 1-5 and Figures 2-15), we can easily say that the best performance is given by MLP FFN model for prediction of BSE sensex and the network with one hidden layer having 3 neurons gave the best results. We also suggest that the hidden layer should be activated with sigmoid function and the output layer by linear function. We get best results with batch training having 10 records per iteration and 10 epochs for whole training. The detailed summary of the model is given below in Table 6.

From the above statistics (Table 7), we can see that closing value of the previous

Table 6. Case processing summary.

		N	Percent
Sample	Training	707	60.3%
	Testing	239	20.4%
	Holdout	227	19.4%
Valid		1173	100.0%
Excluded		58	
Total		1231	

(Source: Reasearcher's analysis of data)

Table 7. Independent variable importance.

	Importance	Normalized Importance
Volume	0.077	20.4%
Open	0.155	41.2%
High	0.293	77.8%
Low	0.098	26.0%
Close	0.377	100.0%

(Source: Reasearcher's analysis of data)

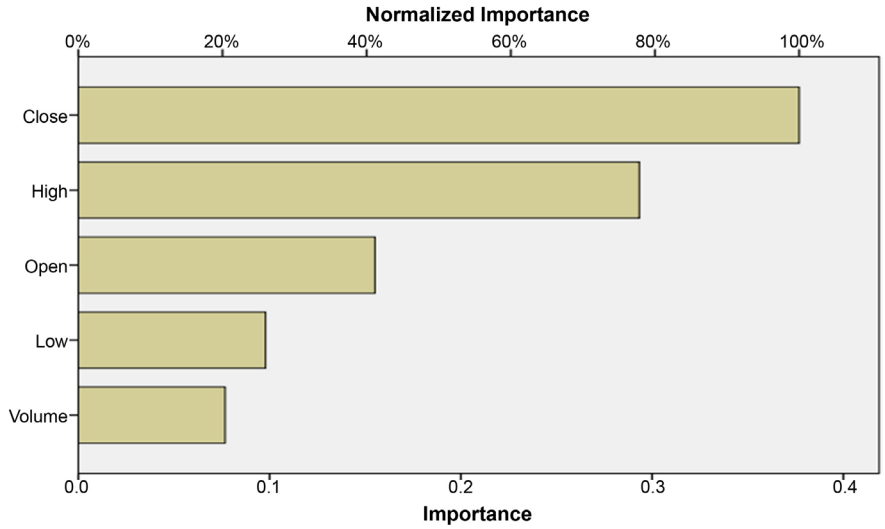

Figure 15. Graph showing variable importance in prediction of closing value of BSE Sensex on the next day. (Source: Reasearcher's analysis of data)

day has the highest effect on the closing value of the successive day. Highest previous day values have a little less importance.

6. Conclusion

In our attempt to investigate the effects of change in hyperparameters in the performance of neural network model for BSE Sensex prediction, we discuss 5 different cases. In our first case, we saw the effect of change in number of hidden

layers and their activation functions. In the second case we changed the batch size of each iteration. Next we probed the effects of change in number of units in hidden layer. Case 4 assessed the change in number of training epochs used in the model. Lastly we changed the multilayer perceptron feed forward network to radial bases function network and tapped its performance and predictive ability. From all the above experiments, we conclude that the best model for prediction of BSE sensex next day close value is a multilayer perceptron feed forward network with gradient descent based back propagation. MLP FFN with one hidden layer, having three nodes and sigmoid activation function gave the best prediction, with least performance error and best trade-off training time. The network activation should use sigmoid function for hidden layers and linear function for output layers. 10 records per iteration and 10 epochs of training gave the most accurate results with feasible training time. Also from our study we conclude that the most important variable for prediction of next day sensex closing value is the closing value of the previous trading day. This result is in line with the previously established theories where it is said that the stock markets follow a random walk process and the trading strategy is a martingale process. Today's price is the best prediction for tomorrow's price. Thus using the best fit neural network model discovered in our research, we can use today's close price to predict tomorrow's best value.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M. (1990) Stock Market Prediction System with Modular Neural Networks. 1990 *IJCNN International Joint Conference on Neural Networks*, San Diego, 17-21 June 1990. https://doi.org/10.1109/IJCNN.1990.137535

[2] Ghiassi, M. and Saidane, H. (2005) A Dynamic Architecture for Artificial Neural Networks. *Neurocomputing*, **63**, 397-413. https://doi.org/10.1016/j.neucom.2004.03.014

[3] Ghiassi, M., Saidane, H. and Zimbra, D.K. (2005) A Dynamic Artificial Neural Network Model for Forecasting Time Series Events. *International Journal of Forecasting*, **21**, 341-362. https://doi.org/10.1016/j.ijforecast.2004.10.008

[4] Chang, P.-C., Liu, C.-H., Lin, J.-L., Fan, C.-Y. and Ng, C.S.P. (2009) A Neural Network with a Case Based Dynamic Window for Stock Trading Prediction. *Expert Systems with Applications*, **36**, 6889-6898. https://doi.org/10.1016/j.eswa.2008.08.077

[5] Hamzacebi, C., Akay, D. and Kutay, F. (2009) Comparison of Direct and Iterative Artificial Neural Network Forecast Approaches in Multi-Periodic Time Series Forecasting. *Expert Systems with Applications*, **36**, 3839-3844. https://doi.org/10.1016/j.eswa.2008.02.042

[6] Cheng, J., Chen, H. and Lin, Y. (2010) A Hybrid Forecast Marketing Timing Model

Based on Probabilistic Neural Network, Rough Set and C4. 5. *Expert Systems with Applications*, **37**, 1814-1820. https://doi.org/10.1016/j.eswa.2009.07.019

[7] Liao, Z. and Wang, J. (2010) Forecasting Model of Global Stock Index by Stochastic Time Effective Neural Network. *Expert Systems with Applications*, **37**, 834-841. https://doi.org/10.1016/j.eswa.2009.05.086

[8] Guresen, E., Kayakutlu, G. and Daim, T.U. (2011) Using Artificial Neural Network Models in Stock Market Index Prediction. *Expert Systems with Applications*, **38**, 10389-10397. https://doi.org/10.1016/j.eswa.2011.02.068

[9] Moghaddam, A.H., Moghaddam, M.H. and Esfandyari, M. (2016) Stock Market Index Prediction Using Artificial Neural Network. *Journal of Economics, Finance and Administrative Science*, **21**, 89-93. https://doi.org/10.1016/j.jefas.2016.07.002