Comparison of HDNN with other Machine Learning Models in Stock Market Prediction

Vaibhav Kumar
Department of Computer Science & Engineering,
DIT University, Dehradun, India

M. L. Garg
Department of Computer Science & Engineering,
DIT University, Dehradun, India

ABSTRACT
A hybrid deep learning model has been developed in this research which is the combination of a deep neural network and the fuzzy inference system. This model is termed in this paper as Hybrid Deep Neural Network (HDNN). In this model, we have integrated the Sugeno fuzzy inference system with the deep neural network. This model has been used in the task of stock market prediction. Through this model, the day's closing price of a stock has been predicted on the basis of certain factors as parameters which affect the price of a stock. We have tested our model in the prediction of seven stocks and compared the result of prediction with popular similar models. It was found that the HDNN model has the best performance in this task. In This paper, we will present the comparison of HDNN model with five other machine learning models- Generalized Linear Model (GLM), Multilayer Perceptron (MLP), Gradient Boost Model (GBM), Random Forest Model (RF) and the Deep Neural Network (DNN).

Keywords
Stock Market Prediction, Machine Learning, Deep Learning.

1. INTRODUCTION
There has been a long research in the field of stock market prediction [1]. The stock market is considered to be very dynamic and complex in nature. An accurate prediction of future prices may lead to higher yield of profit to investors through stock investments. As per the prediction, the investor shall be able to pick the stocks which may give a higher return. Almost every investor today wants to analyze the stock which he or she wishes to pick for investment. There a many investment based companies which invest money of their clients in stock. These companies also follow many analytics techniques.

The early prediction methods in this domain used to take high effort from the analysts and their result were hard to match the fluctuating prices of stocks. Most of them were based on conventional statistical techniques. This was the era when machine learning was not a trend. But the time has changed and many analytics works are done on the basis of machine learning techniques. Various machine learning tools have been used by the researchers to predict the prices of stocks. Various models of the artificial neural network have been used in stock price prediction with their own strength and limitation [2]. After the improvement and advancement in the field of predictive analytics, now deep learning techniques are in trend [3]. They are more capable and have better performance than conventional machine learning techniques. Popular deep learning techniques like variants of deep neural networks have been used in stock market prediction and their result has been much better than the conventional machine learning or statistical techniques [4].

In this research, we have developed a new model of the deep neural network. This model is a hybrid model in which a deep neural network has been integrated with the fuzzy inference system. The resultant model may be termed as the Hybrid Deep Neural Network (HDNN). In this HDNN model both the deep neural network and fuzzy inference system have their own capability and due to these special features we have used this model in our work. There are possibilities that some uncertainties may remain present in the data available for predictive analytics. We need to make predictions on the basis of this data based on machine learning algorithms. But the neural network model cannot handle the uncertainties present in the data. To handle these uncertainties, fuzzy inference system may be applied. If both models are combined together, this new model will have the capability to make predictions by learning from the data with handling the uncertain information.

The performance of the HDNN model is compared with five other machine learning models- Generalized Linear Model (GLM), Multilayer Perceptron (MLP), Gradient Boost Model (GBM), Random Forest Model (RF) and the Deep Neural Network (DNN). These are the popularly used models used in the similar domains. We will present the survey of results and show the percent relative efficiencies of each model.

2. RESEARCH METHODOLOGIES
There are various machine learning methods which are used in predictive analytics. Each model works on an algorithm and trained on the previous datasets. A model after training will be able to predict the output value on the given set of input values. In this section, we will present a brief description of the three above discussed models.

2.1 Hybrid Deep Neural Network

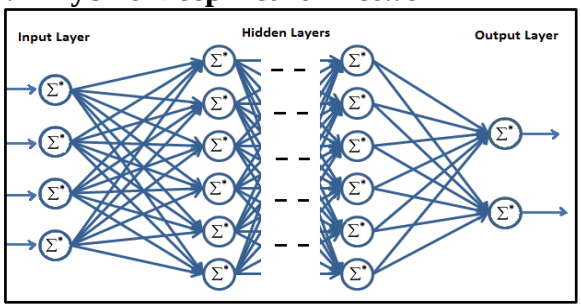

Figure 1: Hybrid Deep Neural Network (HDNN)

We have developed a hybrid model of neural network. This model is the integration of fuzzy inference system with a deep neural network. This model is based on a fuzzy neural network with multiple hidden layers. A fuzzy neural network is a learning model which applies the parameters of fuzzy systems by exploiting the techniques from neural networks. The deep neural network with n number of hidden layers is used in our model and the value of n can be fixed at the time of training. More hidden layers in the network will give the better accuracy in the result but the increase in the number of

hidden layers results in an increase the complexity of the system and hence degrades the performance of training. So deciding the number of hidden layers is also the part of the training process. This HDNN model is represented in figure 1 given below.

At each node of the network, we have used Sugeno Fuzzy Inference System (FIS). This FIS will map the input to its corresponding output. First, the fuzzy rules Σ^* will be determined for this process which can be followed as:

If (x is A) *AND* (y is B) Then $z = f(x, y)$ (1)

where, $f(x, y)$ is the polynomial of the first order. It gives the first order Sugeno fuzzy model which can be described as

Rule 1: If (x is A_1) *AND* (y is B_1) Then $f1\ (x, y) = p_1x + q_1y + r_1$

Rule 2: If (x is A_2) *AND* (y is B_2) Then $f2\ (x, y) = p_2x + q_2y + r_2$ (2)

where, p, q and r are the fuzzy parameters.

The training of HDNN is performed using a hybrid learning algorithm which was presented by Jang. The least square method is used in this algorithm which identifies the consequent parameters on the fourth layer during the forward pass. These parameters are updated using gradient descent method during backward pass an error is propagated in the backward direction.

In the process of fuzzification, with the Gaussian membership function, we have converted all the input values into corresponding fuzzy values. The Gaussian membership function used in fuzzification is given in equation 3.

$$\mu_{Ai}(x) = \exp\left[-\left(\frac{x-b_i}{2a_i}^2\right)\right] \quad (3)$$

where μ_{Ai} is the degree of membership function for fuzzy set A_i and a and b are the parameters of membership function that can change the shape of the membership function. Once the input values are converted into fuzzy values, these fuzzified inputs will be combined according to the fuzzy rule to establish a rule strength. The consequence of this rule will then be found by combining the rule strength and the output membership function. These consequences are then combined to get an output distribution and this output distribution is finally defuzzified. This defuzzification of output distribution will be done by the Mean of Maximum (MoMax) method which is given in equation 4.

$$z = \sum_{j=1}^{l} z_j \, T \quad (4)$$

where z is the mean of maximum, z_j is the point at which the membership function is maximum and l is the number of times the output distribution reaches the maximum level. After this defuzzification process, the output in crisp form is received at the output layer.

Algorithm

1. For fuzzification, use Gaussian function μ to convert crisp input values into corresponding fuzzy values.
2. Determine fuzzy rule Σ^* according to the input values. Use T-norms for fuzzy operations fuzzy OR and fuzzy AND.
3. Initialize the fuzzy parameters p, q and r.
4. Repeat steps 5 to 11 until training conditions met.
5. Update the fuzzy parameters p, q and r using least square method in forward pass and using gradient descent method in backward pass.
6. Update the number of hidden layers and number of nodes at each layer.
7. Combine fuzzy inputs according to Σ^* to establish rule strength.
8. Find the consequence of the rule by combining Σ^* with μ.
9. Obtain the output distribution by combining the consequences.
10. Obtain the crisp output z by MoMax method.
11. Match the obtained output with target output

Algorithm: Hybrid Deep Neural Network (HDNN)

The steps used in the above algorithm are used in our HDNN model for the prediction of stock closing prices. In the above algorithm, we have proposed to take a random number of hidden layers and a random number of nodes at each layer. This number can be adjusted at the time of training according to the accuracy of the model and training performance. There is no any predefined method to find the number of hidden layers and number of nodes at each layer. So it is decided while training by adjusting the numbers at each epoch of training. The fuzzy parameters discussed in the algorithm can be initialized by random values and these can also be adjusted at the time of training.

This hybrid deep learning model which is Hybrid Deep Neural Network is a concept which integrates fuzzy system and the deep neural network. Both systems have several advantages and disadvantages. The limitations of each system can be addressed after combining with the other system. The fuzzy system gets the feature of learning from observed data duet the presence of neural network together. Similarly, the deep neural network gets the feature of handling the uncertain or imprecise information due to the presence of fuzzy system. They both disappear the limitation of each-other after integration.

2.2 Generalized Linear Model

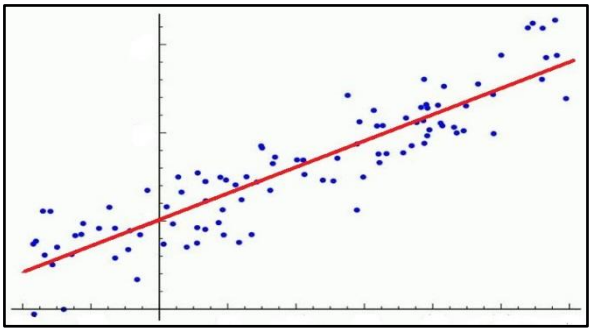

Figure 2: Generalized Linear Model

Generalized linear model, introduced by John Nelder and Robert Wedderburn, is an extension of ordinary linear regression models which is useful in the cases when the dependent variable does not have normal error distribution [51]. This model has a special characteristic that it unifies the statistical models like linear regression, logistic regression, and Poissons regression models [6]. It is represented in figure 2:-

In the generalized linear model, it is assumed that the dependent variable Y follows a distribution in exponential family. In this model, the mean μ depends on the independent variables x_i. This mean is assumed to be a non-linear function of $x_i\beta$ as:-

$$E(Y)= \mu = g^{-1}(x_i\beta) \quad (5)$$

where E(Y) is the expected value of Y, β is an unknown parameter, and *g* is called the link function. Link function in GLM specifies the linking relation between linear model and dependent variable.

GLM consists if the three elements- Probability distribution, linear predictor and a link function. The probability distribution originates from the exponential family. The linear predictor, denoted by η, combines the information about the independent variables in the model. It is expressed as the linear combination of unknown parameters as $\eta = x_i\beta$. The link function mainly relates this linear predictor with the mean of the distribution function.

There are mainly two fitting methods used in GLM- Maximum Likelihood and Bayesian methods. The estimation process in maximum likelihood method may be found using Newton-Raphson method as:

$$B_{(n+1)}= \beta_{(n)}+\tau^{-1} (\beta_{(n)} u\beta_{(n)}) \quad (6)$$

where $\tau (\beta_{(n)})$ is the observed information matrix and $(u\beta_{(n)})$ is the score function. The Bayesian method is mainly used for approximation of posterior distribution.

GLM models are used in such cases where linear regression models have a limitation in work. Linear models which work on normal distribution may have a limitation in modeling measured proportions. It has limitation to observe data where the variance in data increases with the mean.

2.3 Multilayer Perceptron

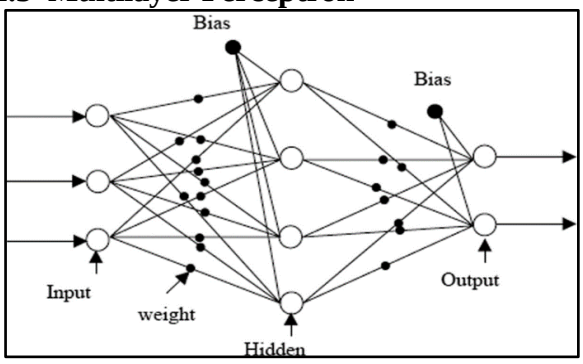

Figure 3: Multilayer Perceptron Model

A multilayer perceptron is a feedforward model of the artificial neural network. It maps a set of input data onto appropriates outputs. This model consists an input layer, an output layer and one or more than one hidden layers between input and output layers. These layers consist of a set of processing neurons. Every neuron of a layer is connected to each neuron of its neighboring layers [7]. The connecting paths are unidirectional in the forward direction- from input layer towards output layer. That is why this model is termed as feedforward model. For the training of multilayer perceptrons, backpropagation learning method is mainly used [8]. The architecture of this model is represented in figure 3.

The net input X_{in} and net output Y_{out} of this network can be obtained as:

$$X_{in} = X \times W \quad (7)$$

and $$Y_{out} = H_{out} \times V \quad (8)$$

where X is the input vector, W is the weight matrix of weights between input and hidden layer, H_{out} is the output from hidden layer and V is weight matrix of weights between the hidden layer and an output layer. When an input vector is applied at the input layer of the network, it produces the corresponding output vector. The process of learning is followed to match the produced output with the desired output. In the process of learning, weights associated with the interconnections between layers are changed at each iteration of training. This change in weight is based on the error calculated in the produced output. This process is followed in backpropagation learning, a supervised learning approach. This is an extension of least square method [9].

If the error in n^{th} output node at i^{th} iteration is denoted by $e_n(i)$, then it can be minimized as:

$$\varepsilon(i)= \frac{1}{2}\sum_n e^2_n(i) \quad (9)$$

and the change in weight can be obtained as:

$$\Delta w_{nj}(i) = - \eta \frac{\partial \varepsilon(i)}{\partial v_n(i)} y_j(i) \quad (10)$$

where y_j is the output of previous j^{th} neural node and η is the learning rate which used for convergence of weights.

The multilayer perceptrons with backpropagation learning have been used very widely in many classification and prediction applications. With the feature of machine learning, they have an advantage over ordinary linear regression models [10].

2.4 Gradient Boost Model

Gradient boost is a machine learning method popularly used in regression and classification to develop predictive models [11]. Gradient boosting consists of three elements- a loss function, a weak learner and an additive model. The loss function is used for optimization and it is used as per the requirement of problem. Weak learners are used for making predictions. Decision trees are mainly used as weak learners in this model. Additive models are used to add weak learners which can minimize the loss function.

Decision trees are generally used as base learners in gradient boosting. A gradient boosting method has been proposed by Friedman which is used to improve the learning with gradient boosting [12]. According to Friedman, a gradient boosting model using decision trees can be formulated as:

$$F_m(x) = F_{m-1}(x)+\gamma_m h_m(x) \quad (11)$$

and $$\gamma_m = \arg\min_{\gamma} \Sigma_{i=1}^{n} L(y_i , F_{m-1}(x_i)+ \gamma\, h_m(x_i)) \quad (12)$$

where F_m is the model at the m^{th} stage of gradient boosting, x is the input value, γ is the coefficient and h_m is the decision tree at m^{th} stage.

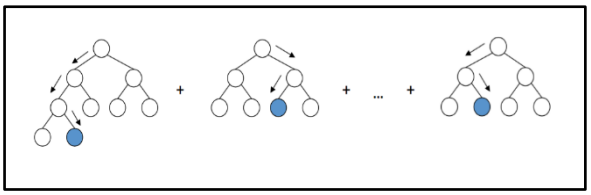

Figure 4: Gradient Boosting Model with Decision trees

There is a parameter called a number of nodes in the tree is used in this model. This parameter allows the level of interaction between variables. It can be adjusted according to the data set used for training. A gradient boosting model with decision trees is represented in figure 4.

2.5 Random Forest

Random forest is a machine learning method belongs to the class of ensemble learning. It is widely used for classification and regression problems [13]. This model is formed with the decision trees and it predicts the output as the mean of the prediction of each individual decision trees in case of regression. In case of classification, it predicts output as the mode of the outputs of each individual decision tree [14]. A typical random forest can be represented as given in figure 5.

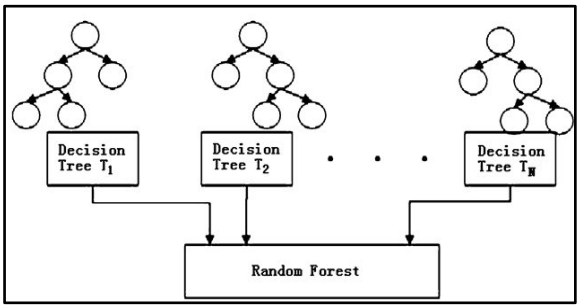

Figure 5: Random Forest with N no. of Decision Trees

When this model is used in regression, leaf nodes of each individual tree predicts the real-valued numbers. In this model, the data is split at some split points for each independent variable based on the homogeneity of the data [15].

A technique known as bagging is used to train the decision trees. Let X is the set of independent variables, Y is the set of dependent variables and b=1, ……, B is the steps of bagging in which in which random samples are selected at each step. Let X_b and Y_b are the training examples to train the decision tree f_b . After the training process, the model can predict for an unknown sample x' as:

$$f' = \frac{1}{B}\Sigma_{b=1}^{B} f_b(x') \qquad (13)$$

2.6 Deep Neural Network

A deep neural network (DNN) is a variety of artificial neural network which has more than one hidden layers [16]. DNNs have a special feature that they can model the complex relationships which are non-linear. DNNs generally have the feedforward architecture. Some recurrent architecture has also been used by researchers in language modeling [17]. Convolutional neural networks have been used as deep neural networks very popularly in the area of image processing [18]. A typical architecture of the deep neural network is represented in figure 6.

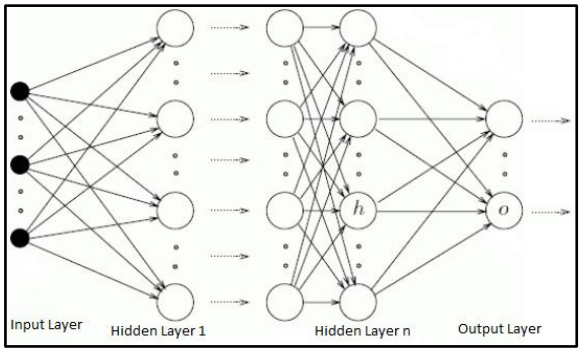

Figure 6: Deep Neural Network

All the processing DNNs are very much similar to the feedforward model of artificial neural networks. Backpropagation learning can be performed to find the matching between produced outputs and desired output. The change in weight in this process can be obtained as:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\frac{\partial C}{\partial w_{ij}} + \xi(t) \qquad (14)$$

where η is the learning rate, C is the cost function and ξ is the stochastic term and w_{ij} is the weight associated with the interconnection between an i^{th} node of one layer and a j^{th} node of next layer.

3. DATA SET

Every predictive model is based on the dataset. If his predictive model is a machine learning based model, then there is a most important role of the data set. This dataset is used for training the machine learning model and later it will be used for testing and validation. In our research, we have used the stock market data. The price of every stock is affected by many parameters. We have included these important parameters as input for the model and on the basis of these input parameters we have predicted the day closing prices. In table 1 given below, we have represented the parameters.

Table 1: Description of Parameters

S. No	Parameter Type	Parameter Name	Parameter Description
1	Input	SENSEX	BSE Sensex Index value
2		NIFTY 50	NSE Nifty 50 Index Value
3		ST_52_High	Stock's 52 weeks highest price
4		ST_52_Low	Stock's 52 weeks lowest price
5		PREV_CLOSE	Previous day closing price of the stock
6		DOLLAR	Current Dollar to INR rate

7		CRUDE	Current crude oil price
8		HANG_SENG	Hang Seng Index
9		DAX	Dow Jones Index
10		SENT_SCR	Market sentiment score
11	Output	DAY_CLOSE	Day closing price of the stock

There are a lot of factors which affects the stock market and the price of each individual stock. Total 10 factors we have taken as an input parameter in our dataset which is the most important factors. Indian stock market is dependent on factors including domestic and global factors. Global factors like Dollar price, crude oil price, performance of US stock market and the opening of Japanese stock market leaves a great impression on the Indian stock market. There is an important reason of including the indices of US and Japanese stock market. First, they both are the big economies and have an important impact on the global economy. Second, US stock market is a very importantly considered stock market which closes in last among all the global markets and gives an impact on the opening of other global markets for next day. Japanese stock market is also an importantly considerable stock market which opens earlier among all the global markets and it also gives an impact on the opening of global stock markets.

We have used a parameter named SENT_SCR which is the market sentiment is given in the range of 0 to 5. A lower score will be given when the market is in bearish form and a higher score will be given when the market is in bullish form. 52 week's highest and lowest prices of a stock are also taken as input parameters because it also has an importance while investing or selling a stock by the investors. When a stock reaches its 52 weeks highest, there are chances that profit booking may start in the stock. If news about the stock I good and it is moving around it 52 weeks lowest then many investors prefer this type of stock for investment.

We have taken 4 years, January 2014 to December 2017, historical prices of 7 stocks- Infosys, SBI, ONGC, Tata Motors, Reliance Industries, Adani Enterprise, and Future Retail. While selecting these stocks, we have taken care to choose a dominating stock from each major sector of the stock market. Apart from the historical prices of these 7 stocks, we have also collected the historical prices of the same duration for all the remaining input parameters. Firstly these parameters are arranged in the data frame for each individual stock for preprocessing. We have applied our HDNN model for day closing price prediction of each individual stock. 70% of the data is used for training purpose and 30% of the data is used for testing purpose. On the successful prediction for one stock, the model will be applied to remaining 6 stocks one by one for predicting their day closing prices. Finally, we will present a consolidated performance and result in an analysis of prediction by the model.

We have fixed our HDNN model as a 10-12*3-1 network. There are 10 nodes at input layer because there are 10 input parameters. There are 3 hidden layers with 12 nodes at each layer. To predict the output, 1 node is taken at the output layer. First, we have applied our model to predict the day closing price of Infosys. It has given 97.2% accuracy in the prediction. Later we have applied our model on remaining 6 stocks for prediction of their day closing prices.

4. RESULTS OF PREDICTION

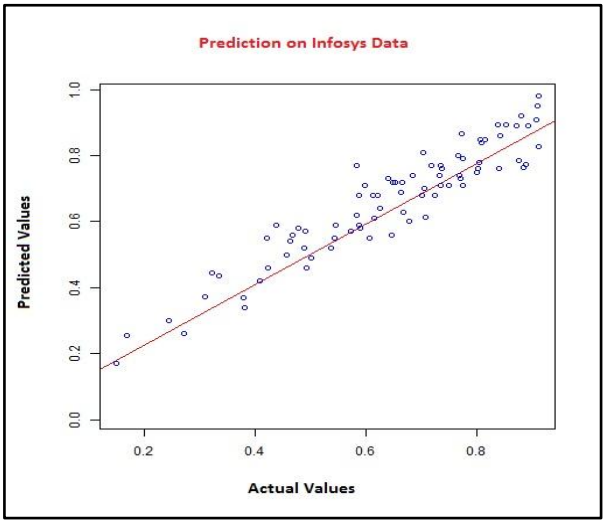

Figure 7: HDNN Prediction on Infosys Data

The comparison of predicted values by HDNN model about day's closing prices and actual day's closing prices of each of the seven stocks is presented below. Out of the whole data set, 30% data has been used in testing and validation.

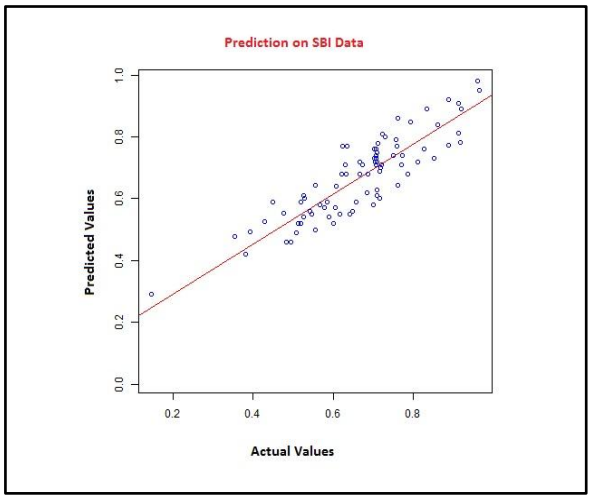

Figure 8: HDNN Prediction on SBI Data

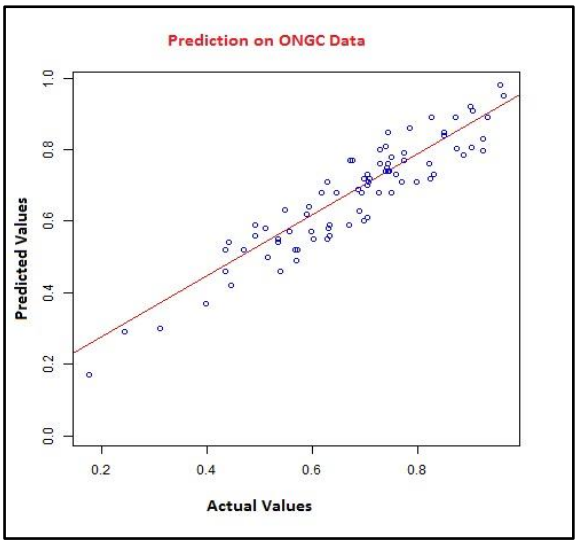

Figure 9: HDNN Prediction on ONGC Data

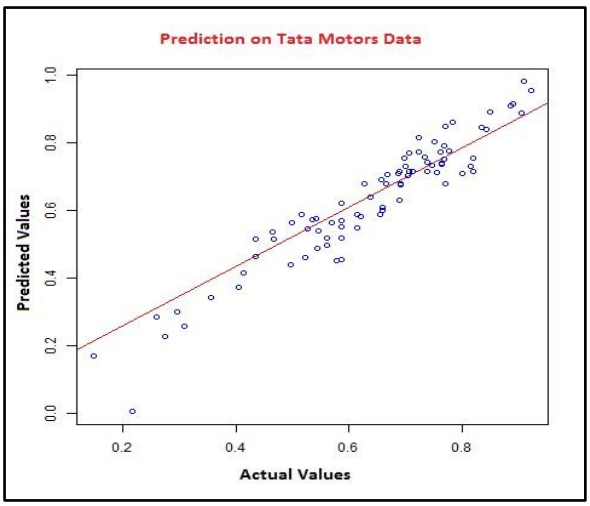

Figure 10: HDNN Prediction on Tata Motors Data

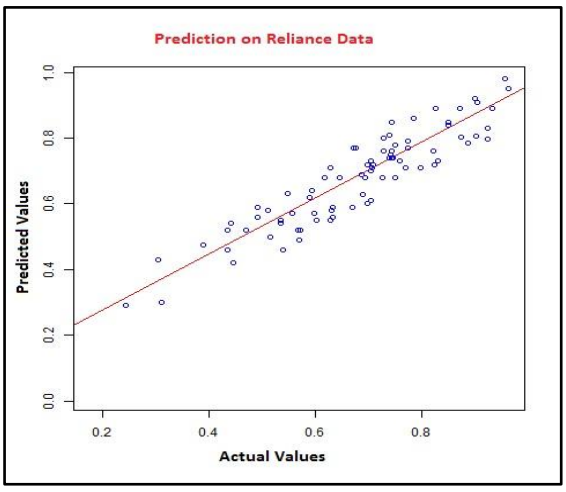

Figure 11: HDNN Prediction on Reliance Data

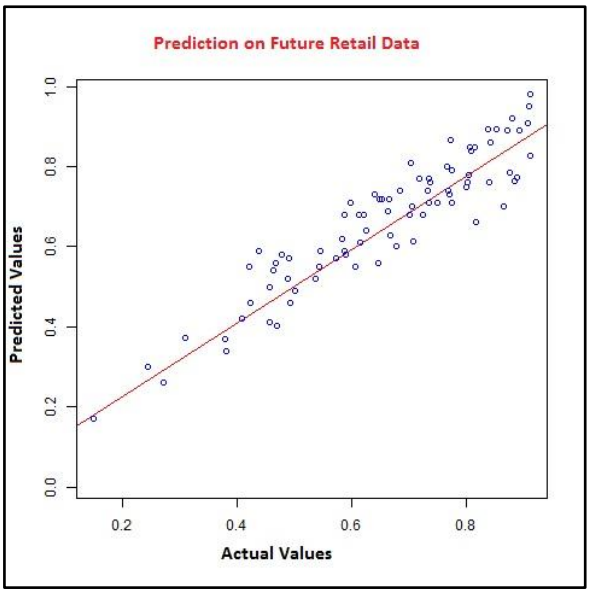

Figure 13: HDNN Prediction on Future Retail Data

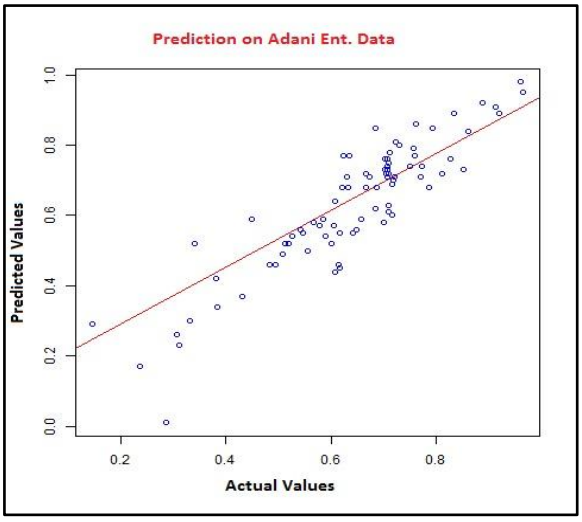

Figure 12: HDNN Prediction on Adani Ent. Data

In the figures from 8.1 to 8.6, the performance of HDNN in predicting the day's closing prices of seven stock has been represented. It can be seen in these figures that the model has a good performance in the task of prediction.

Performance Evaluation

The following parameters have been used to evaluate the performance of the HDNN model.

(a). Mean Squared Error (MSE): The MSE measures the quality of a predictor. If Y is a vector of *n* predictions, and X is the vector of observed values of the variable being predicted, then the MSE can be calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2 \qquad (18)$$

(b). Root Mean Square Error (RMSE): The RMSE is used as a measure of differences between values predicted by a predictor and actual values. It is the square root of MSE. It can be calculated as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{n}} \qquad (19)$$

where Y is a vector of *n* predictions, and X is the vector of observed values of the variable being predicted.

(c). Mean Absolute Error (MAE): The MAE is the measure of differences between two continuous variables. It can be calculated as:

$$\text{MAE} = \frac{\sum_{i=1}^{n}|Y_i - X_i|}{n} \qquad (20)$$

where Y is a vector of *n* predictions, and X is the vector of observed values of the variable being predicted.

The performance of HDNN model during training is given in table 2.

The performance of HDNN during testing and validation is given in table 3.

It can be seen in the table 2 and 3 that the HDNN model has a better performance in testing and validation as compared to the performance of the training. This shows that the model is best fitted and it has not an issue of overfitting. A model which as better performance in training than that in testing and validation is said to be overfitted.

The accuracy of the HDNN model in prediction in five runs is given in table 4.

It can be seen in the above table that the average accuracy of HDNN is more than 93% in all the runs. Now the performance of HDNN model will be compared with the existing similar important models on the basis of important performance measuring parameters. We will compare the performance of our HDNN model with Generalized Linear Model (GLM), Multilayer Perceptron (MLP), Gradient Boost Model (GBM), Random Forest Model (RFM), and Deep Neural Network (DNN). We will compare the performances of each of the model in terms of accuracy in prediction which is given in table 5.

Percent Relative Efficiency

To find out the performance of a model in comparison to other models can be obtained by calculating the percent relative efficiency (PRE). It will show how efficient a model is in front of the rest of the similar models. To calculate the PRE, mean squared error of the new model and the compared old model is used. It can be calculated as:

$$\text{PRE (new)} = \frac{MSE\ (old) * 100}{MSE\ (new)} \qquad (21)$$

Table 2: HDNN Performance During Training

	Infosys	SBI	ONGC	Tata Motors	Reliance Ind	Adani Ent	Future Retail
MSE	0.082	0.074	0.09	0.070	0.087	0.10	0.099
RMSE	0.07	0.085	0.074	0.065	0.054	0.99	0.088
MAE	0.058	0.043	0.078	0.095	0.1	0.08	0.074

Table 3: HDNN Performance During Testing and Validation

	Infosys	SBI	ONGC	Tata Motors	Reliance Ind	Adani Ent	Future Retail
MSE	0.009	0.008	0.07	0.005	0.009	0.01	0.018
RMSE	0.055	0.070	0.061	0.048	0.040	0.020	0.058
MAE	0.037	0.021	0.053	0.071	0.080	0.062	0.071

Table 4: Accuracy in Prediction by HDNN

	Run 1	Run 2	Run 3	Run 4	Run 5	
	Correct (%)	Correct (%)	Correct (%)	Correct (%)	Correct (%)	Average Accuracy
Infosys	96.90	97.10	97.50	96.80	97.80	97.08
SBI	97.20	95.90	94.50	96.00	96.90	95.90
ONGC	96.50	96.20	93.00	95.10	96.60	95.20
Tata Motors	96.10	96.80	96.90	97.20	96.85	96.75
Reliance Ind	95.40	96.40	96.10	97.85	98.10	96.44
Adani Ent.	94.20	94.40	96.95	97.55	93.80	95.78
Future Retail	93.30	92.10	95.90	93.00	95.95	93.58

Table 5: Comparison of HDNN with Other Models							
Data/Model	**Infosys**	**SBI**	**ONGC**	**Tata Motors**	**Reliance Ind**	**Adani Ent**	**Future Retail**
HDNN	97.08	95.90	95.20	96.75	96.44	95.78	93.58
GLM	90.25	90.10	89.99	90.00	89.85	89.00	88.75
MLP	91.33	92.66	90.45	93.00	93.00	90.33	90.66
GBM	95.65	94.35	93.55	92.50	94.10	91.40	91.00
RFM	94.26	93.20	91.22	95.50	93.50	91.99	90.50
DNN	95.80	94.90	94.30	96.00	95.00	92.50	92.85

Table 6: Percent Relative Efficiency (in %)						
	HDNN	**GLM**	**MLP**	**GBM**	**RFM**	**DNN**
HDNN	100.00	327.20	225.00	175.50	150.00	120.00
GLM	30.56	100.00	75.52	40.50	40.40	30.95
MLP	44.40	132.41	100.00	66.66	70.55	60.20
GBM	57.14	248.44	250.00	100.00	150.00	110.00
RFM	66.66	250.00	141.84	66.66	100.00	88.00
DNN	83.33	327.86	166.11	90.90	113.63	100.00

The percent relative efficiency of each of the models is given in table 6.

It can be seen in the table that the Hybrid Deep Neural Network is most efficient among all the similar machine learning models in the task of stock market prediction. After observing the results in figures and tables, it is clear that HDNN is the best model among all the existing similar models in the task of predicting the day's closing prices of the stocks.

5. CONCLUSIONS AND FUTURE SCOPE

The HDNN model used in this research has been applied to the task of stock market prediction. It has been observed that this model has the better performance in testing and validation as compared to the performance in training This indicates that the model is best fitted it does not has the issue of overfitting which common in many neural network systems. We have tested our model in predicting the day's closing prices of seven stocks. We have performed this process five times to find the average accuracy in prediction by the model. It has been observed that the model has given more than 93% accuracy in all the runs. However, it has given around 97% accurate result in predicting the day's closing price of Infosys. The performance of prediction by HDNN has also been compared with some existing popularly used similar machine learning models. Deep neural network and the gradient boost model have also given better results but they have been lagging behind the HDNN model. To find out the strength of each of these models in front of other models in the task of predicting the day's closing prices of stocks, we have calculated the percent relative efficiency. It can be seen that the HDNN has been proved as a most efficient model in this task of stock prediction. It should also be noticed that the HDNN is 1.2 times efficient that deep neural network in this task. We have seen that this model has given more than 93% accurate results in prediction. It has given a better result as compared to other important models. So it has been proved that this model can be applied in stock market predictions.

This HDNN model can be used in the works of prediction other than the stock market. But this can be used in the work which uses the numerical data. It can be used in many predictive tasks like predicting the results of students in a subject based on the internal marks, predicting the outcome of a match based on the previous information about important factors. So, there is a big scope of using this model in the tasks of predictive analytics. There are many financial organizations which use this concept for financial investments. Every prediction in the organization or company is based on certain input values. So this model will be very much useful for these organizations which are related to predictive analytics. Not only the organizations who do financial investments but the organizations which want to predict the sale value of its products based on certain factors can also use this model. There is a scope to create new features in these model so that they can be applied in many domains with better performance. The new techniques may be integrated to exploit the opportunities of the model in prediction. Parameter tuning can also help to improve the performance of these models. Parameter tuning can also help to improve the performance of these models. So it can be said that there is a very wide opportunity and open scope for this model. Similarly, this research opens the scope of development of models based on the HDNN.

6. REFERENCES

[1]. J W Lee, 2001, "Stock price prediction using reinforcement learning", Proceedings of the IEEE International Symposium on Industrial Electronics, Pusan, South Korea.

[2]. L Pastor, P Veronesi, 2009, "Technological Revolutions and Stock Prices", American Economic Review, Vol-99, Issue-4, Pages- 1451-83.

[3]. A Yoshihara, K Fujikawa, K Seki, K Uehara, 2014, "Predicting Stock Market Trends by Recurrent Deep Neural Networks", Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Pages- 759-769.

[4]. K Chen, Y Zhou, F Dai, 2015, "A LSTM-based method for stock returns prediction: A case study of China stock market", Proceedings of the IEEE International Conference on Big Data, CA, USA.

[5]. Shoichi Eguchi, "Model Comparison for generalized linear models with dependent observations" (2017), Econometrics and Statistics, Vol-59.

[6]. John Nelder, Robert Wedderburn, (1972), "Generalized Linear Models", Journal of Royal Statistical Society.

[7]. Frank Rosenblatt, (1961), "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", Spartan Books.

[8]. David Rumelhart, Geoffrey Hinton, R. J. Williams, (1986), "Learning Internal Representations by Error Propagation", Parallel Distribute Processing: Explorations in the microstructure of cognition, Vol-1.

[9]. Varun Kumar Ojha, Ajith Abraham, Vaclav Snasel, (2017), "Metaheuristic design of feedforward neural networks: A review of two decades of research", Engineering Applications of Artificial Intelligence, Vol-60.

[10]. Simon Haykin, (2012), "Neural Networks: A comprehensive Foundation", 2nd Edition, Prentice Hall.

[11]. Ji Kan, (2017) , "Evaluation of Mining Engineering technology innovation ability and application based on BP neural network", International Conference on Industrial Technology and Management (ICITM).

[12]. Yanru Zhang, Ali Haghani, (2015), "A gradient boosting method to improve travel time prediction", Transportation Research, Part C.

[13]. J. H. Friedman, (1999), "Greedy Function Approximation: A Gradient Boosting Machine".

[14]. Tin Kam Ho, (1995), "Random Decision Forests", 3rd International Conference on Document Analysis and Recognition, Montreal.

[15]. Tin Kam Ho, (1998), "The Random Subspace Method for Constructing Decision Forests", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol-20.

[16]. Tin Kam Ho, (2002), "A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructions", Pattern Analysis and Applications.

[17]. J. Schmidhuber, "Deep Learning in Neural Networks", Technical Report IDSIA-03-14 arXiv: 1404.7828.

[18]. J. Schmidhuber, (2001), "LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages", IEEE Transaction on Neural Networks, Vol-12.