International Journal of Advanced Engineering and Technology
ISSN: 2456-7655
www.newengineeringjournal.com
Volume 1; Issue 1; March 2017; Page No. 46-51

Design and implementation of artificial neural network system for stock market prediction (A case study of first bank of Nigeria PLC Shares)

[1] Olotu SI, [2] Fasiku AI

[1] Computer Science Department, Federal University of Technology Akure, Ondo State Nigeria
[2] Computer Engineering Department, Ekiti State University Ado - Ekiti, Ekiti State, Nigeria

Abstract
Investing in stock is a reliable way individuals and institutional investors make profit. In the stock market (or capital market) company stocks, bonds, and other securities are traded at an agreed price. At each trading activity in the market, prices of individual stock are normally quoted. Because the stock price fluctuates due to market forces, a sudden downward trend in stock price could yield to a huge loss of capital and discourage a potential investor. Therefore a good prediction of the stock market price is the key to successful trading. Out of several techniques that have been researched to predict the market price, the artificial neural network (ANN) technique gave a more efficient prediction because it allows deeper analysis of large set of data especially those that have the tendency to fluctuate within a short period of time. The First Bank of Nigeria PLC shares dataset is used as case study in this work. In developing the neural network system the research adopts the Multi Layer Feed Forward (MLFF) neural network model. The network is trained using Backpropagation algorithm. The research takes opening price, closing price, number of deals on stock, volume of stock traded and trading day of the week of First Bank of Nigeria PLC shares history as a neural network input. The output of the network model is the predicted price.

Keywords: stock market prediction, backpropagation, feed forward neural network, artificial neural network

1. Introduction

Stocks represent ownership in a company. Investors buy shares to become part owners (shareholders) of a company (Nwaiwu, 2004) [9]. The place where stocks are traded at an agreed price is known as the stock market (or capital market) (Nwaiwu, 2004; Kamich, 2003) [9, 5]. At each trading activity in the market, prices of individual stock are normally quoted. Because the stock price fluctuates due to market forces, a sudden downward trend in stock price could yield to a huge loss of capital and discourage a potential investor. Therefore a good prediction of the stock market price is the key to successful trading. A successful prediction of a stock's future price could yield significant profit. However, some believe that stock price movement are governed by the random walk theory and thus unpredictable. Others disagree and those with this view point possess a myriad of methods and techniques which purportedly allow them to gain future price information. Within the last two decades, a great deal of attention has been focused on the idea of predicting stock prices and price fluctuations (Wojciech, 2007) [12].

Several research efforts have been carried out to predict the market in order to make profit using different techniques ranging from statistical analysis, technical analysis, to fundamental analysis among others, with different results. These techniques cannot provide deeper analysis that is required and therefore not effective in predicting stock market prices. Artificial neural network (ANN) technique is one of data mining techniques that is gaining increasing acceptance in the business area due to its ability to learn and detect relationship among nonlinear variables. Also, it allows deeper analysis of large set of data especially those that have the tendency to fluctuate within a short of period of time. This makes ANN a candidate for stock market prediction. Much research efforts have been made to improve the predictive accuracy and computational efficiency of share values (Wu *et al.*, 2001).

2. Artificial Neural Network Model

The basic processing elements of neural networks are called artificial neurons, or simply neurons or nodes. In a simplified mathematical model of the neuron, the effects of the synapses are represented by connection weights that modulate the effect of the associated input signals, and the nonlinear characteristic exhibited by neurons is represented by a transfer function. The neuron impulse is then computed as the weighted sum of the input signals, transformed by the transfer function. The learning capability of an artificial neuron is achieved by adjusting the weights in accordance to the chosen learning algorithm. The basic architecture consists of three types of neuron layers: input, hidden, and output layers. In feed-forward networks, the signal flow is from input to output units, strictly in a feed-forward direction. The data processing can extend over multiple (layers of) units, but no feedback connections are present (Adeyemo *et al.*, 2012) [2].

2.1 Learning Algorithm

The learning algorithm is based on the generalization of the error-correction learning rule. Specifically, the actual response of the network is subtracted from a desired response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections – hence the name "backpropagation". The weights are adjusted so as to make the actual output of the network move closer to the desired output.

2.2 Backpropagation algorithm

The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods. Properly trained backpropagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input target pairs and get good results without training the network on all possible input/output pairs. The algorithm involves the neuron error gradient and weight update.

a) The Neuron Error Gradient

The weights in the neural network are updated to give the correct output at the output layer. This forms the basis of training the neural network. The back-propagation is used for the weight updates. The input is fed in, the errors calculate and filtered back through the network making changes to the weights to try reduce the error (Bobby, 2008) [4].

The weight changes are calculated by using the gradient descent method. This is achieved by following the steepest path on the error function to try and minimize it. That is, the error at the output neuron is taken (Desired value – actual value) and multiplied by the gradient of the sigmoid function. If the difference is positive there is need to move up the gradient of the activation function and if negative move down the gradient of the activation function (Bobby, 2008) [4]. The formula to calculate the basic error gradient for each output neuron k is:

$$\delta_k = y_k(1 - y_k)(d_k - y_k) \quad (3.12)$$

Where y_k is the value at output neuron k and d_k is the desired value at output neuron k

(Bobby, 2008) [4]

This is a difference between the error gradients at the output and hidden layers. The hidden layer's error gradient is based on the output layer's error gradient (back propagation) so for the hidden layer the error gradient for each hidden neuron is the gradient of the activation function multiplied by the weighted sum of the errors at the output layer originating from that neuron using the formula:

$$\delta_j = y_j(1 - y_j)\sum_{k=1}^{n} w_{jk}\delta_k \quad (3.13)$$

(Bobby, 2008) [4]

b) The Weights Update

The final step in the algorithm is to update the weights, this occurs as follows:

$$w_{ij} = w_{ij} + \Delta w_{ij} \quad (3.14)$$

$$w_{jk} = w_{jk} + \Delta w_{jk} \quad (3.15)$$

$$\Delta w_{ij}(t) = \alpha . inputNeuron_j . \Delta\delta_j \quad (3.16)$$

$$\Delta w_{jk}(t) = \alpha . hiddenNeuron_j . \Delta\delta_k \quad (3.17)$$

Where

α = learning rate

δ = error gradient

The alpha value is the learning rate; this is usually a value between 0 and 1. It affects how large the weight adjustments are and so also affects the learning speed of the network. This value needs to be carefully selected to provide the best results, too low and it will take ages to learn, too high and the adjustments might be too large and the accuracy will suffer as the network will constantly jump over a better solution and generally get stuck at some sub-optimal accuracy (Bobby, 2008) [4].

2.3 Transfer Function

Transfer functions calculate a layer's output from its net input n. Before leaving the node, this number is passed through a nonlinear mathematical function called a sigmoid transfer function. The study uses a particular neural transfer called hyperbolic tangent sigmoid transfer function. The input to the sigmoid is a value between -∞ and +∞, while its output can only be between -1 and 1. The function is defined graphically as:

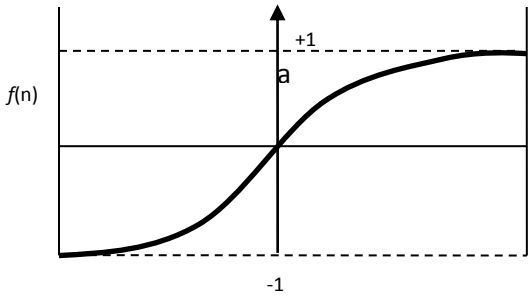

Fig 1: Hyperbolic tangent sigmoid transfer function. Mathematically, it is described by the equation:

$$a = tansig(n) = \frac{2}{(1 - e^{-2n})} \quad (3.2)$$

Where n = input to the node

a = node output

3. Literature Review

Majumder *et al.* (2010) [7] proposed a computational approach for predicting the S&P CNX Nifty 50 Index with an average accuracy of 69.72%. Only the historic prices of the Index values were used for the prediction. Other macro economic factors and other international stock market data as input variables can also be used as input variables in order to improve the accuracy of the model. Adebiyi *et al.* (2012) [1] examined the effect of combination of different market indicators for stock price prediction. The hybrid market indicators consist of technical, fundamental and expert opinion variables as input to artificial neural networks model which is trained with backpropagation algorithm. The prediction model 74% accuracy with Dell stock data and 70% with Nokia stock data both obtained from New York Stock Exchange are used as stock data. Pan *et al.* (2005) [10] presents a computational approach for predicting the Australian stock market index using neural to obtain a prediction correctness of 80%. Kara *et al.* (2010) developed efficient model to predict the direction of movement in the daily Istanbul Stock Exchange (ISE) National 100 Index. Results showed that average performance of ANN model (75.74%). Thenmozhi (2006) [11] used neural

network to predict the daily returns of BSE (Bombay Stock Exchange) Sensex. The prediction accuracy of the training data is 96.3% and that of test data is 96.6%. To produce better prediction of stock prices can be achieved by including other micro and macro-economic variables as inputs. Akintola *et al.* (2011) [3] developed a neural network solution to predict the share values of Intercontinental Bank PLC of the Nigerian Stock Exchange using the error back propagation algorithm. The prediction accuracy of 75%. However, the Accuracy of the forecast can be improved further by training the network with more data. Neenwi *et al.* (2013) [8] analysed stock prices of Access Bank PLC, First Bank PLC, and UBA PLC banks using artificial neural network with back-propagation by using a four day price movement to predict the next market day's price direction. A 30-day stock price was used to predicted price at an accuracy of 50%, 83.33% and 83.33% respectively.

4. System Design

In developing the neural network system the research adopts the Multi Layer Feed Forward (MLFF) neural network. The network consists of an input layer with 5 neurons, 2 hidden layers with 5 neurons each and an output layer with the one neuron (figure 2).

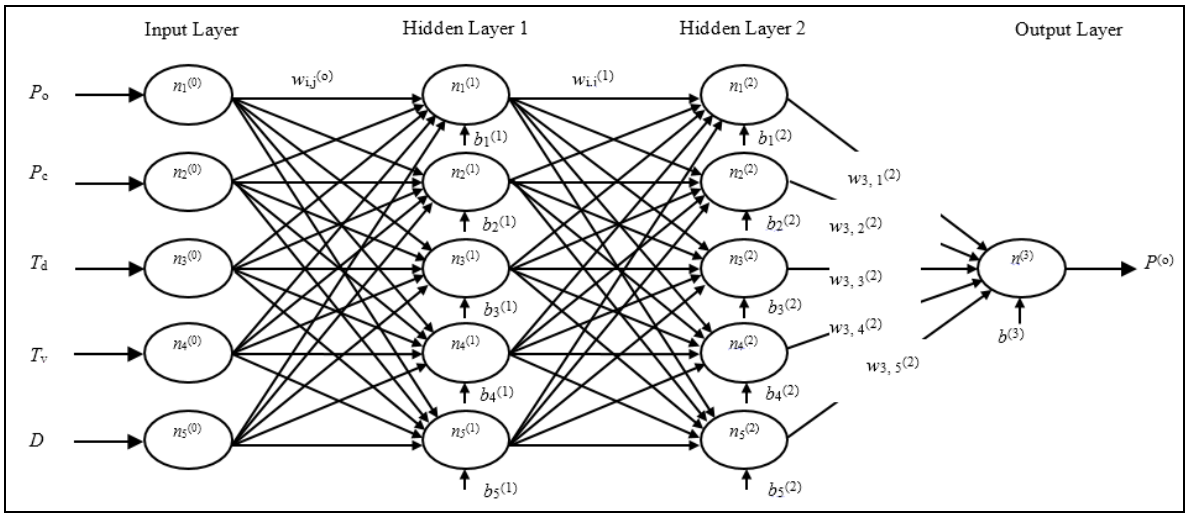

Fig 2: Architecture of the neural network for stock market prediction

The inputs to the network, P_o, P_c, T_d, T_v, D are opening stock price, closing stock price, number of trading deals, volume of shares traded, weekday respectively. Each input is fed in through the input layer nodes $n^{(0)}$ and is multiplied by the weights $w_{i,j}^{(0)}$ associated with the connection with the next layer nodes to obtain the weighted sum $S_i^{(1)}$ as shown in equation (1). An additional bias node b_i with a constant input of 1 is added to make the neural network more flexible.

$$S_i^{(1)} = \sum_{i=0}^{n} \left(n_i^{(0)} \times w_{i,j}^{(1)} \right) + b_i^{(1)}$$
$$(i = 1, \ldots,5; j = 1, \ldots,5; n = 4) \qquad (1)$$

Before leaving the hidden layer 1 node, the weighted sum is passed through sigmoid function in equation (2) to obtain the outputs $n_i^{(1)}$ in equation (3).

$$f(x) = \frac{1}{(1 - e^{-x})} \qquad (2)$$
$$n_i^{(1)} = f\left(S_i^{(1)}\right)(i = 1, \ldots,5) \qquad (3)$$

The hidden layer 1outputs $n_i^{(1)}$ serves as inputs to hidden layer 2 where the weighted sum $S_i^{(2)}$ is calculated and activation function applied to produce the output $n^{(2)}$ as shown in equation (4) and (5) respectively.

$$S_i^{(2)} = \sum_{i=0}^{n} \left(n_i^{(1)} \times w_{i,j}^{(2)} \right) + b_i^{(2)}$$
$$(i = 1, \ldots,5; j = 1, \ldots,5; n = 4) \qquad (4)$$
$$n_i^{(2)} = f\left(S_i^{(2)}\right)(i = 1, \ldots,5) \qquad (5)$$

The hidden layer 2 outputs $n_i^{(2)}$are also fed into the output layer where the final weighted sum $S^{(3)}$ and activated function is derived as shown in equation (6) and (7) respectively.

$$S^{(3)} = \sum_{i=0}^{n} \left(n_i^{(2)} \times w_{i,j}^{(3)} \right) + b_i^{(3)}$$
$$(i = 1, \ldots,5; j = 1, \ldots,5; n = 4) \qquad (6)$$
$$n^{(3)} = f\left(S^{(3)}\right) \qquad (7)$$

The output $n^{(3)}$gives the network output $P^{(o)}$ which is the opening stock price of the next trading day as shown in equation (8).

$$P^{(o)} = n^{(3)} \qquad (8)$$

The back propagation algorithm compares the network outputs $P^{(o)}$ with the actual values $P^{(t)}$ to compute the error value ε as shown in equation (9).

$$\varepsilon = \frac{1}{n} \sum_{i=0}^{n} \left(P_i^{(t)} \times P_i^{(o)} \right)^2$$
$$(i = 1, \ldots,5; n = 175) \qquad (1.9)$$

5. Implementation

The implementation of the neural network prediction was

done using MATLAB 7.7 (R2008), a high-performance language for technical computing. The reason for the choice of the software is the presence of the neural network toolboxes which has comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems.

5.1 Data Collection

The dataset is downloaded online from Capital Asset Ltd, a stockbroking firm with the Nigerian Stock Exchange (NSE) with the URL: www.capitalassetsng.com. The data is stored in a text file with the column arranged in the format number: opening stock price, closing stock price, number of deals, volume of shares sold and the weekday. The data collected is shown in table A-1.

5.2 Data Preprocessing

An important task in preparing data sets is data normalization. The importance of data scaling is to avoid attributes to greater numerical ranges dominate values in smaller numerical ranges and also to avoid numerical difficulties during the calculation. The recommended range of data for neural network is to be scaled in between [-1, +1] or [1, 0]. The data is first transformed into a standard form for the network to learn easily. In this study, the inputs are normalized to value between -1 and +1. This can be done by a number of normalization techniques but one of the popular techniques used expresses the data in terms of the maximum and minimum of the data set. All the values are normalized by using the following equation:

$$Y = \frac{(2 \times X) - (Max + Min)}{(Max + Min)} \qquad (3.1)$$

Where
Y = normalized value
X = present value

5.3 Data Partitioning

The data needs to be partitioned for training and testing purposes. The system allows the ratio of the train data to test data to be set at 50:50, 60:40 or 70:30. For this study the 70:30 ratio was picked. Since there are 575 sets of data in all, the training data will be 403 while the test data 172.

5.4 Network Parameters

At creation, the network weights and biases are initialized. The best prediction training parameters needs to be set. These include the number of iterations for training (epoch), learning rate and momentum. Upon several trials and best values were set at 2000 and 0.5 for epochs and learning rate respectively. The momentum parameter is not applicable to gradient descent backpropagation as so set at 0.

5.5 Network Training and Validation

After the network parameters are set, the network is ready for training. 400 data points is used to train the network which runs for 2000 epochs (iterations). On completion the system gives a training completed alert. In order to test for the efficiency of the system, the network is validated by simulating the trained network to obtain its response to the test inputs data. The network is then evaluated using 174 data points.

6. System Evaluation

The accuracy of the estimated output is improved by an interactive learning process in which the outputs for various input vectors are compared with targets and an average error is computed (Akinwale *et al.* 2009). As each input is applied to the network, the network output is compared to the target. The performance of the models was evaluated using five metrics which include mean square error. The error is calculated as the difference between the target output and the network output. The goal is to minimize the average of the sum of these errors. It is necessary to get a measure of the spread of the y values around the average. To do this the root-mean-square (rms) error is used. To construct the rms error, the residuals are determined. The residual are the difference between the actual values and the predicted values denoted by t_i - a_i, where a_i is the observed value for the i^{th} observation and t_i is the predicted value.

Table 1: Prediction result of 20 test datasets

	Actual (t)	Predicted (a)	Error (t-a)	Error^2 (t-a)^2	% Error (%)	Accuracy (%)
1	10.66	10.76	-0.10	0.01	0.96	99.04
2	10.65	10.74	-0.09	0.01	0.86	99.14
3	10.50	10.74	-0.24	0.06	2.30	97.70
4	10.50	10.65	-0.15	0.02	1.46	98.54
5	10.30	10.52	-0.22	0.05	2.14	97.86
6	10.01	10.29	-0.28	0.08	2.79	97.21
7	10.25	10.52	-0.27	0.07	2.64	97.36
8	10.01	10.46	-0.45	0.20	4.48	95.52
9	10.09	10.18	-0.09	0.01	0.94	99.06
10	10.00	10.24	-0.24	0.06	2.38	97.62
11	9.76	10.06	-0.30	0.09	3.10	96.90
12	9.25	9.46	-0.21	0.04	2.27	97.73
13	8.48	8.93	-0.45	0.21	5.34	94.66
14	8.52	8.54	-0.02	0.00	0.21	99.79
15	9.30	9.07	0.23	0.05	2.52	97.48
16	9.82	9.56	0.26	0.07	2.61	97.39
17	9.77	10.05	-0.28	0.08	2.85	97.15
18	10.00	10.03	-0.03	0.00	0.28	99.72
19	9.65	10.02	-0.37	0.14	3.85	96.15
20	9.00	9.48	-0.48	0.24	5.39	94.61

Table 1 shows prediction results for 20 observations. The complete prediction is shown in Figure A-2. The typical performance function that is used for training feedforward neural networks is the mean sum of squares of the network errors given by:

$$\text{mse} = \frac{1}{n}\sum_{i=0}^{n}(t_i - a_i)^2 \qquad (5.1)$$

where t is the target output and a is the actual output n is the number of data points of test data

The mean squared error of the network where n=175, total $(t-a)^2$ is 10.54 is given as:

$$\text{mse} = \frac{1}{175} \times 10.54 = 0.06 \qquad (5.1)$$

This gives the measure of distance between the targets and the outputs of the network. It is observed that that the predicted values are close to the line of perfect agreement due to the relatively small values of the error. The system has an accuracy average accuracy of 97.92%.

7. Result and Analysis

The outputs (predicted stock prices) and actual stock prices are plotted against the number of days as shown in Figure 3.

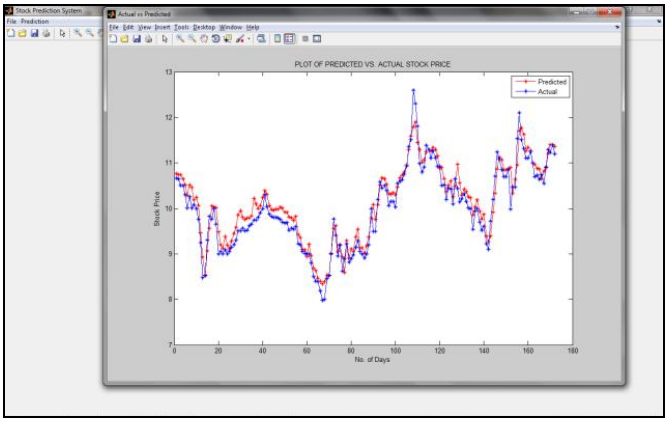

Fig 3: Plot of predicted versus actual stock price

The validation error is computed and a plot of the training errors, validation errors, and test errors appears, as shown in the following figure. The best validation performance occurred at iteration 9, and the network at this iteration is returned.

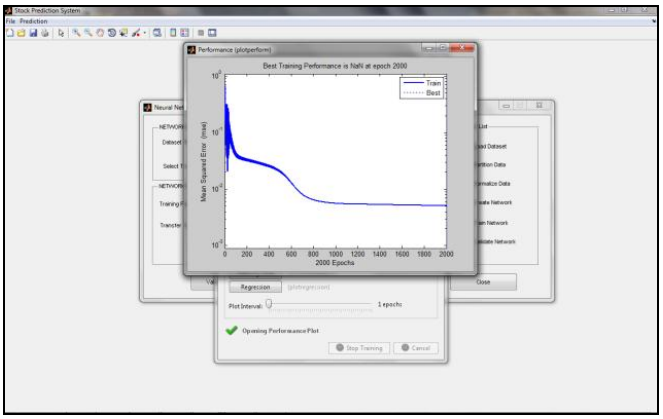

Fig 4: Performance Plot

The plot shows the mean squared error of the network starting at a large value and decreasing to a smaller value. In other words, it shows that the network is learning. The plot has three lines, because the 506 input and targets vectors are randomly divided into three sets. 70% of the vectors are used to train the network. 30% of the vectors are used to validate how well the network generalized. Training on the training vectors continues as long the training reduces the network's error on the validation vectors. After the network memorizes the training set (at the expense of generalizing more poorly), training is stopped. This technique automatically avoids the problem of overfitting, which plagues many optimization and learning algorithms.

The last 30% of the vectors provide an independent test of network generalization to data that the network has never seen. The analysis of the network response was performed between the network outputs and the corresponding targets. The following figure shows the results.

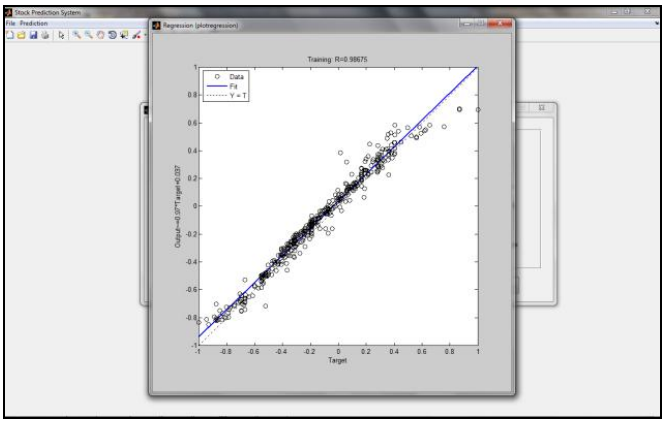

Fig 5: Regression Analysis Result

The output tracks the targets very well for training, testing, and validation, and the R-value is over 0.99 for the total response.

8. Conclusion

This study utilizes the ANN techniques approach to forecast stock price. For this purpose, an efficient three-layer neural network with revised back propagation algorithms was developed to forecast the First Bank of Nigeria PLC (FBN) stock. As a result, the following day price of the stock was predicted with a mean square error (mse) of 0.0441. This result shows that it is possible to model Nigerian stock market prices based on historical trading data by using a three layer neural network.

9. Future Work

Future work can be done on this study by using different network systems such as recurrent network system and by considering more than one time series variables with the hope of investigation the performance of neural network in time series forecasting. The result shows that neural network is a good method for predicting time series data.

10. References

1. Adebiyi AA, Ayo CK, Adebiyi MO, Otokiti SO. An Improved Stock Price Prediction using Hybridized Market

Indicators, Journal of Emerging Trends in Computing and Information Sciences. 2012, 3(1).
2. Adeyemo OO, Osofisan AO. Empirical Study of Statistical Based Method and Artificial Neural Networks for Credit Rating in Nigerian Banks, International Journal of Research and Reviews in Computer Science (IJRRCS). 2012, 3(3). ISSN: 2079-2557.
3. Akintola KG, Alese BK, Thompson AF. Time Series Forecasting With Neural Network: A Case Study Of Stock Prices Of Intercontinental Bank Nigeria, International Journal of Research and Revies in Applied Sciences. 2011, 9(3).
4. Bobby. Basic Neural Network Tutorial – Theory. 2008. http://takinginitiative.net/2008/04/03/basic-neural-network-tutorial-theory/
5. Kamich BM. How TA Works, New York Institute of Finance. 2003.
6. Kara Y, Boyacioglu MB, Baykan OK. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange, Journal of Expert Systems with Applications. 2011, 5311-5319.
7. Manna Majumder, MD Anwar Hussian. Forecasting Of Indian Stock Market Index Using Artificial Neural Network. 2010.
8. Neenwi S, Asagba PO, Kabari LG. Predicting The Nigerian Stock Market Using Artificial Neural Network, European Journal of Computer Science and Information. 2013; 1(1):30-39.
9. Nwaiwu IC. The Beginner's Guide to Investing in the Nigerian Stock Market", Stockmarketnigeria.com™. 2004.
10. Pan H, Tilakaratne C, Yearwood J. Predicting Australian Stock Market Index Using Neural Networks Exploiting Dynamical Swings and Intermarket Influences, Journal of Research and Practice in Information Technology. 2005, 37(1).
11. Thenmozhi M. Forecasting Stock Index Returns Using Neural Networks, Delhi Business Review. 2006, 10(7-2).
12. Wojciech G. Neural Network Predictions of Stock Price Fluctuations. 2007.
13. Zhang YP, Wu T, Zhang L. A Self-Adjusting and Probabilistic Decision Making Classifier Based on The Constructive Covering Algorithm In Neural Networks, Proc. Int. Conf. on Machine Learning and Cybernetics. 2002; 4:2171-2174.