American Journal of Theoretical and Applied Statistics
2013; 2(4): 94-101
Published online July 10, 2013 (http://www.sciencepublishinggroup.com/j/ajtas)
doi: 10.11648/j.ajtas.20130204.11

Evaluation of artificial neural networks in foreign exchange forecasting

Akintunde Mutairu Oyewale

Department of Statistics, University of Botswana, Botswana

Email address:
waleakintunde2004@yahoo.com

To cite this article:
Akintunde Mutairu Oyewale, Evaluation of Artificial Neural Networks in Foreign Exchange Forecasting. *American Journal of Theoretical and Applied Statistics.* Vol. 2, No. 4, 2013, pp. 94-101. doi: 10.11648/j.ajtas.20130204.11

Abstract: This study investigates the modeling, description and forecasting of exchange rates of four countries (Great Britain Pound, Japanese Yen, Nigerian Naira and Batswana Pula) using Artificial Neural Network, the objective of this paper is to use ANN to predict the trend of these four currencies. ANN was used in training and learning processes and thereafter the forecast performance was evaluated or measured making use of various loss functions such as root mean square error (RMSE), mean absolute error (MAE), mean absolute error (MAE), mean absolute precision error (MAPE) and Theill inequality coefficient (TIC). The loss functions used are good indicator of measuring the forecast performance of these series, the series with the lowest function gave a best forecast performance. Results show that the ANN is a very effective tool for exchange rate forecasting. Classical statistical methods are unable to efficiently handle the prediction of financial time series due to non-linearity, non-stationarity and high degree of noise. Advanced intelligence techniques have been used in many financial markets to forecast future development of different capital markets. Artificial neural network is a well tested method for financial markets analysis.

Keywords: Artificial Neural Networks, Foreign Exchange, Loss Functions, Training and Learning Processes

1. Introduction

Forecasting is a very important activity and essential factor in the financial markets that is useful for various players like investors, academia, practitioners, regulators and policy makers. Forecasting with a very week tools has an adverse effect on the economic development due to its negative effect on international trade and investment, using a weak model to forecast will lead to taking a wrong judgment. For this reason forecasting is very important to all categories listed above. An artificial neural network (ANNs), as an emerging discipline emulates the information processing capabilities of neurons of the human brain. It uses a distributed representation of the information stored in the network, and thus resulting in robustness against damage and corresponding fault tolerance (Shadbolt and Taylor, 2002). To develop a feed forward artificial neural network for forecasting exchange rate purpose, the specification of its architecture in terms of number of inputs, number of hidden layer, number of neurons and output is very important.

In the last four or five decades many different non-linear models have been proposed in the literature to model and forecast exchange rates. ANNs are nowadays used in a large variety of modeling and forecasting problems. The merits of using artificial neural networks in this study has to do with adaptive learning nature of ANNs that is ability to learn how to perform an operation on data given based on the training and self Organization. It provides promising alternatives tools for forecasters. The inherently non linear structure of neural networks is useful for capturing the complexity underlying relationship in many real world problems It is a more reliable methods for forecasting applications because not only that they can find linear structure in a problem they can also model linear processes. However, the weakness or limitations of neural networks in any study includes: Black-box problem which is a situation where all information are contained in weight matrices for their output and hidden layers, and these do not make apparent identities of salient predictor variables. This is in contrast with parametric variables which do not only identify influential variables but also provides the degree of their contributions.

Many years ago neural networks have found their ways into financial analysis as well, as evidenced by conferences such as Neural networks in the Capital markets' and a large

number of books (Trippi and Turban.1993: Azoff, 1994; Refenes, 19955; Gately, 1996) and articles in scientific journals dealing with financial applications of neural networks. A casual literature search suggests that many articles are published dealing with modelling and forecasting stock prices (Gencay,1996;) Haefke and Hlmenstein, (1996a) and (1996b), Gencay and Stengos, 1998; Qi and Maddala, 1999), exchange rates (Kuan and Liu,1995, Franses and van Griensven, 1998; Franses and Van Homelen, 1998; Gencay,1999). Neural Networks have successfully been used for exchange rate forecasting. However, due to a large number of parameters to be estimated empirically, it is not a simple task to select the appropriate neural networks architecture for an exchange rate forecasting problem. Researchers often overlook the effect of neural networks parameters on its performance.

2. Structure of Ann Used in this Study

This study used the Feed forward back-propagation Artificial Neural Network and the procedure used is based on the previous work by Erims, et.al 2007.

STEP I: Evaluation of net input to the j^{th} node and k^{th} node in the hidden layer as below:

$$net_j = \sum_{i=1}^{n} w_{ij} x_i - \theta_j$$

$$net_k = \sum_{j=1}^{n} w_{jk} x_j - \theta_k$$

Where i is the input node, j is the hidden layer node, k is the output layer, w_{ij} is the weights connecting the i^{th} input node to the j^{th} hidden layer node, w_{jk} is the weights connecting the j^{th} hidden layer node to the k^{th} output layer, θ_j is the threshold between the input and hidden layers θ_k the threshold connecting the hidden and output layers.

STEP II: Evaluation of the j^{th} node in the hidden layer and the output of the k^{th} node in the output layer as:

$$h_j = f_h\left\{\sum_{i=1}^{n} w_{ij} x_i - \theta_j\right\},$$

$$y_k = f_k\left\{\sum_{j=1}^{n} w_{jk} x_j - \theta_k\right\}$$

Where

$$\text{and } f_k(x) = \frac{1}{\left(1+\exp\left(-\lambda_k x\right)\right)},$$

Where:

h_j is the vector of hidden neurons,

y_k is the output-layer of neurons

$f_h(x)$ and $f_k(x)$ are logistic sigmoid activation functions from input layer to the hidden layer and from hidden layer to output layer respectively, and λ_h and λ_k are the variables which control the slope of the sigmoid function. The output of each neurons is obtained by applying an activation functions $f_h(x)$ and $f_k(x)$. The nodes are used to perform the non-linear input/output transformation using a sigmoid activation function.

STEP III: The calculations of errors in the output and hidden layers can be expressed as follows:

The output layer error between the target and the observed output is expressed as:

$$\delta_k = -\left(d_k - y_k\right) f_k^1$$

$f_k^1 = y_k\left(1-y_k\right)$ for sigmoid function

Where δ_k is the vector of errors for each output neurons y_k and d_k are the target activation of output layer. The term δ_k depends only on the error $\left(d_k - y_k\right)$ and f_k^1 is the local slope of the node activation function for output nodes. The hidden layer error term is expressed as

$$\delta_j = f_h^1 \sum_{h=1}^{n} w_{kj}\delta_k,$$

$$f_h^1 = h_j\left(1-h_j\right), \text{ for sigmoid function}$$

δ_j is the vector of errors for each hidden layer neurons, δ_k is a weighted sum of all nodes and f_h^1 is the local slope of the node activation for hidden nodes.

STEP IV: The adjustment of weights and thresholds in the output layer and hidden layer is obtained as follows:

$$w_{kj}^{t+1} = w_{kj}^t + \alpha\delta_k h_j + \eta\left(w_{kj}^t - w_{kj}^{(t-1)}\right),$$

$$w_{ji}^{t+1} = w_{ji}^t + \alpha\delta_j h_i + \eta\left(w_{ji}^t - w_{ji}^{(t-1)}\right),$$

$$\theta_k^{t+1} = \theta_k^t + \alpha\delta_k, \theta_j^{t+1} = \theta_j^t + \alpha\delta_j$$

Where α is the learning rate, η is the momentum factor, and t the time period.

2.1. Input Data

The inputs data are fed in through a file already created in excel software and same is imported into the SPSS which is the software used in analyzing the data. The study has four variables of length 444 each, each of the four variables are analyzed one after the other from where the forecast values are obtained and compared with observed/collected data. The selection of hidden layers is largely due to the numbers that give a satisfactory model performance. The number of hidden layers depend largely on the complexity or otherwise of problem theoretically. However, it has been proved that two hidden layers are sufficient and capable to solve any input-output problem. In the light of this, the study start from 2 hidden layers and different hidden layers used until the optimum number of hidden layers emerged. The rule of thumb actually recommended that the number of connections in a network should be ten percent of the training data. This study will use 1728 0f 1776 collected data for monthly forecasting; this implies that the nodes to be selected should be up to 173 connections. The above rule shall be taken seriously as a guide and the test shall be conducted for the optimum number of nodes so needed in each hidden layers by looking at the node with smallest mean square error. The search for the optimum number of nodes shall began by first looking at 2 neurons in each hidden layer and there after proceed to higher number of neurons until the optimum result is accomplished in terms of low mean square error. The output here refers to the predicted values of the monthly data of exchange rate loaded into ANN model. The output will then be compared with collected or observed data where in the test of adequacy shall be performed.

3. Train the Network and the Number of Iteration for Training

The objective of training is to find the set of weights between the neurons that determine the global minimum of error function. Unless the model is over fitted the set of weights should provide good generalization. The back propagation network uses a gradient descent training algorithm which adjusts the weights to move down the steepest slope of the error surface. Finding the local minimum is not guaranteed since the error surface can include many local minima in which the algorithm can become "stuck". This section discusses when to stop training a neural network and the selection of learning rate and corresponding momentum values. There are two schools of thoughts regarding the number of iteration for training before stopping. The first school of thought emphasized that the researcher can only stop training when there is no improvement in the error function based on the reasonable number of randomly selected starting weights. The point at which the network error does not improve is called point of convergence. The objective of convergence training is to attain a global minimum. This school of thought emphasized the need/danger of not getting trapped in a local minimum. The other school of thought advocates a series of train-test interruptions; here the training is stopped after a pre-determined number of iterations and the network ability to generalize on the testing set. Generalization refers to the idea that a model based on sample of the data is suitable for forecasting the entire population. It is the aim of the study to use the first school of thought as pre-determined the number of iteration ahead may lead to the bias of either over fitting or under fitting, instead the network is allowed to determine the number of iteration that give optimal result.

The learning algorithm that controls back-propagation follows a number of steps:

(i) Initialization: Initialize connection weights and neurons to small random values.

(ii) Data introduction: Introduce the continuous data set of inputs and actual outputs to the back-propagation network.

(iii) Calculations of outputs: Calculate the outputs, and adjust the connection weights several times applying the current network error.

(iv) Adjustments: Adjust the activation thresholds and weights of the neurons in the hidden layers according to Delta rule.

(v) Repeat steps (ii) to (iv) until desired network accuracy (error) is reached.

Delta rule recursive algorithm re-adjusts connection weights and biases starting at the output nodes and working back to the first hidden layer. Connection weights are adjusted by:

$$\gamma_{ij}(t+1) = i\gamma_{ij}(t) + \eta\delta_j x_i^1$$

Where

γ_{ij} = the weight from hidden node I or from an input to node j at time t.

x_i^1 = either the output of node I or an input.

η = the positive constant that measures the speed of the convergence of the weight vector (gain term or learning rate).

δ_j = an error term for j (k goes over all nodes in the layers before node j):

$$\delta_j = \begin{cases} y_j(1-y_j)(d_j - y_j), & when\ j \text{ is an output node} \\ x_j^1(1-x_j^1)\sum_k \delta_k \gamma_{jk}, & when\ j \text{ is a hidden node} \end{cases}$$

Where

d_j = the desired output of node j .

y_j = the actual output of the node j .

The node biases are adapted through the similar iterative algorithm by assuming they are connection weights on links from constant-valued input

4. Model Selection and Evaluation

Having obtained the outputs which are the forecast and the predicted values, these values are evaluated by comparing the predicted values with the targets values using the mean square error and for the fewer pipes deviation (the difference between the predicted from the actual in pipes).

The following criteria will be used to evaluate the forecast performance of this model

$$RMSE = \sqrt{\left\{T^{-1}\sum_{t=1}^{T}\left(Y_t - \hat{Y}_t\right)\right\}}$$

$$MAE = T^{-1}\sum_{t=1}^{T}\left|Y_t - \hat{Y}_t\right|$$

$$MAD = T^{-1}\sum_{t=1}^{T}\left(Y_t - \overline{Y}_t\right)$$

$$MAPE = T^{-1}\sum_{t=1}^{T}\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right| X\ 100$$

The smaller the values of these criteria the better the forecast performance; If the values are close to zero we say it is better, if the results are not consistent among the first three we choose the MAPE to be the benchmark.

5. Data Analysis and Discussion of Results

```
GET DATA
/TYPE=XLS
/FILE='C:\Users\Akintunde\Desktop\2 NORMALIZED DATA OF 04012013.xls'
/SHEET=name 'NORMALIZED DATA OF 17122012'
/CELLRANGE=full
/READNAMES=on
/ASSUMEDSTRWIDTH=32767.
EXECUTE.
DATASET NAME DataSet1 WINDOW=FRONT.
*Multilayer Perceptron Network.
MLP ZNAIRA (MLEVEL=S) ZPOUND (MLEVEL=S) ZPULA (MLEVEL=S) ZYEN (MLEVEL=S) BY DATE
/PARTITION TRAINING=7 TESTING=3 HOLDOUT=0
/ARCHITECTURE AUTOMATIC=YES (MINUNITS=1 MAXUNITS=50)
/CRITERIA TRAINING=BATCH OPTIMIZATION=SCALEDCONJUGATE LAMBDAINITIAL=0.0000005 SIGMAINITIAL=0.00005 INTERVALCENTER=0 INTERVALOFFSET=0.5 MEMSIZE=1000
/PRINT CPS NETWORKINFO SUMMARY CLASSIFICATION
/PLOT NETWORK
/STOPPINGRULES ERRORSTEPS= 1 (DATA=AUTO) TRAININGTIMER=ON (MAXTIME=15) MAXEPOCHS=AUTO ERRORCHANGE=1.0E-4 ERRORRATIO=0.0010
/MISSING USERMISSING=EXCLUDE.
```

5.1. Case Processing Summary

Table 1 below shows that 320 cases were assigned to the training sample and 124 to the testing sample (hold out), this is in conformity with the standard rules that at least 70% of the sample must be set for training and the remaining sample left is for testing.

Table 1. *Case Processing Summary*

		N	Percent
Sample	Training	320	72.1%
	Testing	124	27.9%
Valid		444	100.0%
Excluded		0	
Total		444	

Table 2. *Network Information*

Input Layer	Factors	1	TIME
	Number of Units[a]		37
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		8
	Activation Function		Hyperbolic tangent
Output Layer	Dependent Variables	1	POUND
		2	YEN
		3	PULA
		4	NAIRA
	Number of Units		4
	Rescaling Method for Scale Dependents		Standardized
	Activation Function		Identity
	Error Function		Sum of Squares

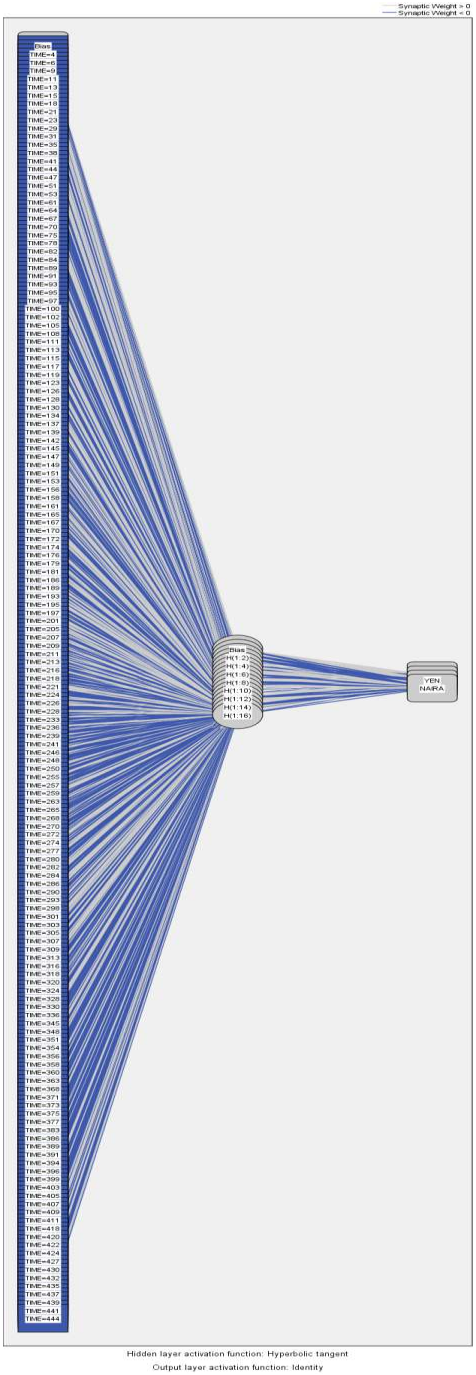

Figure 1. *Network Architecture for The Study*

5.2. Network Information

The network information table 2 displays information about neural networks and its usefulness in ensuring that the specifications are correct. We note here that the number of units in the input layer is the number of factors which is 37 (years) in all, likewise, a separate output units is created for each category of previously defaulted, for a total of 4 units in the output layer. Automatic architecture selection has chosen 9 units in the hidden layer. All other network information is default for the procedure,

5.3. Model Summary

Table 3 below shows information on the result of training and applying the network to the testing (hold out) sample. Sums of squares error is displayed *because* the output layer has scale-dependent variables. This is the error function that the network tries to minimize during training. It should be noted that the sums of squares and all errors values are computed for the rescaled values of the dependent variables as shown in the table below.

The relative error for each scale-dependent variable is the ratio of the sum of squares error for dependent variable to the sum of squares error for the "null" model in which the mean value of the dependent variable is used as the predicted value for each case. There appeared to be more errors in Pound (0.014), Pula (0.012)and Yen (0.018) compared with Naira (0.001).

The average overall error is the ratio of sum of squares for all dependent variable to the sum of squares error for the "null" model, in which the mean values of dependent variables are used as the predicted values for each case. In this case the average overall error differs considerably for Naira and somehow closes to other variables. The average overall relative error and relative errors are fairly constant for training and testing, which gives us some confidence that the model is not over trained and that the error in future cases scored by the network will be close to the error reported in the table.

Table 3. *Model Summary of the series used in the study*

Training	Sum of Squares Error		7.095
	Average Overall Relative Error		.011
	Relative Error for Scale Dependents	POUND	.014
		YEN	.018
		PULA	.012
		NAIRA	.001
	Stopping Rule Used		1 consecutive step(s) with no decrease in error[a]
	Training Time		0:00:00.45
Testing	Sum of Squares Error		2.846
	Average Overall Relative Error		.012
	Relative Error for Scale Dependents	POUND	.011
		YEN	.020
		PULA	.016
		NAIRA	.001

a. Error computations are based on the testing sample.

5.4. Parameter Estimates of the Series Used in the Study

Table 4 show the parameters estimates and the predicted values as shown in the table below, the total number of hidden layers used for the architecture of the model under study are nine and the number of years (Time) upon which the data for the study was collected is 37. The predicted values for the ninth hidden layers are seen in the column titled output layer. This prediction we shall later used in chapter to determine if this model has actually predicted very well.

Table 4. *Parameter Estimates of the series used in the study*

Predictor		**Predicted**											
		Hidden Layer 1								**Output Layer**			
		H(1:1)	**H(1:2)**	**H(1:3)**	**H(1:4)**	**H(1:5)**	**H(1:6)**	**H(1:7)**	**H(1:8)**	**POUND**	**YEN**	**PULA**	**NAIRA**
	(Bias)	-0.265	-0.132	-0.257	0.316	-0.796	-0.321	0.344	0.198				
	[TIME=1]	-0.275	-0.389	-0.058	-0.698	-0.725	-0.471	-1.155	0.257				
	[TIME=2]	-0.247	-0.659	-0.026	-0.576	0.104	-0.062	-1.385	0.028				
	[TIME=3]	0.217	-0.113	0.458	-0.452	-0.082	-0.014	-0.849	0.32				
	[TIME=4]	-0.316	0.422	-0.179	-0.001	-0.425	-0.603	-0.656	-0.355				
	[TIME=5]	0.354	0.232	0.161	-0.041	-0.528	-0.784	-0.383	0.121				
	[TIME=6]	0.554	0.022	-0.182	-0.336	-0.132	-0.634	-0.357	0.085				
	[TIME=7]	0.476	0.098	-0.28	-0.353	0.235	-0.197	-0.428	-0.051				
	[TIME=8]	0.056	-0.34	0.105	-0.032	-0.117	-0.381	-0.768	-0.369				
	[TIME=9]	-0.31	-0.356	-0.036	0.18	-0.072	-0.395	-0.883	-0.369				
	[TIME=10]	-0.074	-0.534	0.07	0.091	-0.028	-0.14	-0.793	0.178				
	[TIME=11]	0.185	-0.724	0.422	-0.034	-0.686	0.137	-0.611	0.199				
	[TIME=12]	0.319	-0.101	0.001	0.36	-0.567	-0.141	-0.172	0.455				
	[TIME=13]	0.314	0.51	-0.506	0.007	-0.055	0.178	-0.07	-0.247				
	[TIME=14]	0.508	0.164	-0.625	0.461	-0.285	-0.407	0.193	0.3				
	[TIME=15]	0.656	0.057	-0.166	0.458	-0.371	-0.236	0.205	0.159				
	[TIME=16]	0.605	-0.026	-0.621	0.5	-0.197	-0.527	0.211	-0.382				
	[TIME=17]	0.393	0.204	-0.485	0.41	-0.429	-0.367	0.152	-0.365				
	[TIME=18]	-0.134	0.113	-0.355	0.638	0.639	0.462	-0.359	0.099				
	[TIME=19]	-0.165	0.62	0.067	0.649	0.117	0.504	-0.153	-0.311				
	[TIME=20]	-0.145	0.698	-0.008	0.304	0.001	0.745	-0.081	0.151				
	[TIME=21]	-0.136	0.508	-0.11	0.578	-0.015	0.596	-0.028	0.3				
	[TIME=22]	0.16	0.039	-0.158	0.365	-0.126	0.789	0.111	0				
	[TIME=23]	0.076	-0.205	-0.249	0.1	-0.437	1.006	0.051	0.242				
	[TIME=24]	0.38	-0.788	-0.218	-0.027	-0.493	1.244	0.193	0.312				
	[TIME=25]	-0.072	-0.022	0.262	0.04	0.44	0.169	0.279	-0.136				
	[TIME=26]	-0.142	0.041	0.311	-0.259	0.167	0.276	0.471	0.321				
	[TIME=27]	0.191	-0.306	0.55	-0.31	-0.083	0.337	0.994	-0.168				
	[TIME=28]	-0.392	-0.664	0.307	-0.172	-0.221	-0.134	0.601	0.413				
	[TIME=29]	-0.519	0.521	0.846	-0.188	0.564	-0.17	0.204	-0.012				
	[TIME=30]	-0.523	0.318	0.23	-0.14	1.288	-0.233	0.071	-0.255				
	[TIME=31]	-0.184	0.167	0.676	-0.193	0.695	-0.131	0.38	0.301				
	[TIME=32]	-0.237	0.34	0.773	-0.556	0.218	0.489	0.531	-0.193				
Input	[TIME=33]	0.036	-0.416	0.632	-0.169	0.153	0.184	0.896	-0.19				
Layer	[TIME=34]	-0.491	-0.101	0.15	-0.52	-0.376	0.611	0.798	-0.169				

	[TIME=35]	-0.674	0.035	0.282	-0.61	-0.24	0.277	1.572	-0.115			
	[TIME=36]	-0.588	-0.001	0.185	-0.458	0.158	0.136	1.375	-0.308			
	[TIME=37]	-1.019	0.328	0.249	-0.569	0.014	0.231	1.519	-0.29			
	(Bias)									-0.058	0.511	-0.43	-0.047
	H(1:1)									1.159	0.568	-0.94	-0.992
	H(1:2)									0.504	-0.46	-0.61	-0.286
	H(1:3)									-0.735	0.291	0.393	0.559
	H(1:4)									-0.963	-0.77	-0.3	-0.57
	H(1:5)									-0.84	-0.42	0.184	0.721
	H(1:6)									-1.212	-0.39	0.276	-0.258
Hidden	H(1:7)									-0.827	-1.73	1.627	1.444
Layer 1	H(1:8)									-0.147	-0.18	-0.13	0.011

6. Prediction by Observed Chart

Figures 2-4 below, shows that for scale dependent variables, the predicted –by-observed chart displays scatter plots of predicted values on the y-axis by observed values 0n the x-axis for the combined training and testing samples. The network shows that all series appears to be reasonably good at predicting as they all shows show how upward movement of each series.

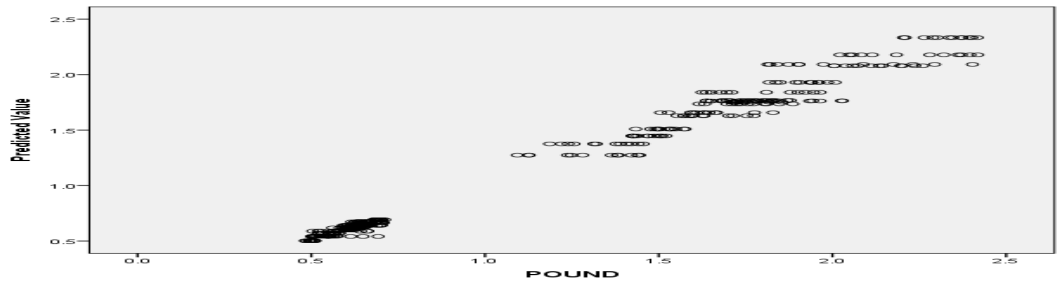

Figure 2. *predicted-by observed chart for Pound*

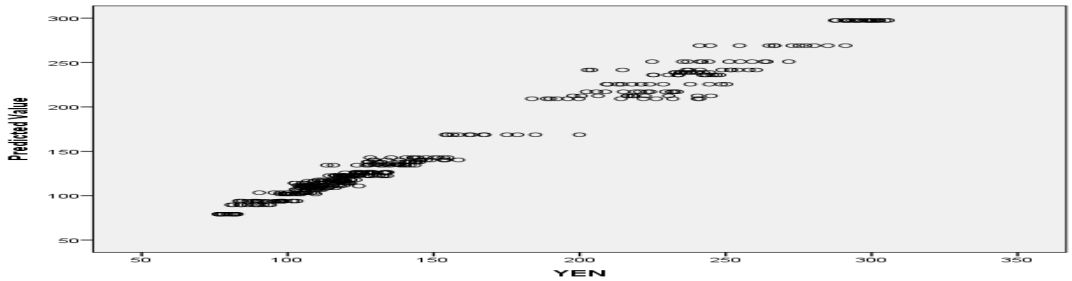

Figure 3. *predicted-by observed chart for Yen*

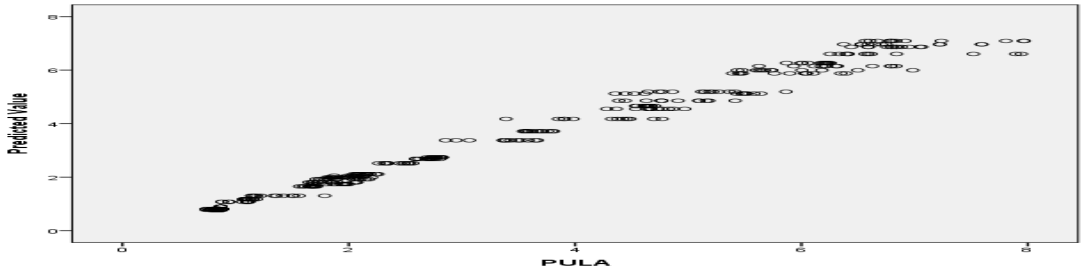

Figure 4. *predicted-by observed chart for Pula*

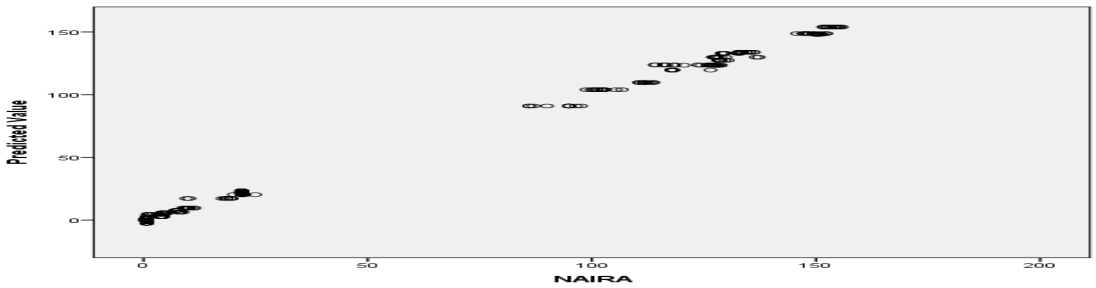

Figure 5. *predicted-by observed chart for Naira*

7. Conclusion

From summary table there appeared to be more errors in Pound (0.014), Pula (0.012) and Yen (0.018) compared with Naira (0.001). So also the average overall relative error and relative errors are fairly constant for training and testing, which gives us some confidence that the model is not over trained and that the error in future cases scored by the network will be close to the error reported in the table. Figures 2-4 below, shows that for scale dependent variables, the predicted –by-observed chart displays scatter plots of predicted values on the y-axis by observed values on the x-axis for the combined training and testing samples. The network shows that all series appears to be reasonably good at predicting as they all shows show how upward movement of each series. This implies that any forecast made on the basis of this model will be very reasonable and produce excellent results.

References

[1] Azoff, E.M., (1994). Neural networks time series forecasting of financial markets. Wiley publishing company (Chichester and New York).

[2] Erims, K., Midilli, A., Dincer, I. and Rosen, M. A. (2007). Artifical Neural Network analysis of World Green Energy Use. *Energy Policy*, 35, no. 3, p. 1731-1743.

[3] Franses, P.H and Van Griensen, K. (1998). Forecasting exchange rates using neural networks for technical trading rules. Studies in non-linear dynamics and Econometrics, 4, 109-114.

[4] Franses, P.H and Van Homelin, P. (1998). On forecasting exchange rates using neural networks. Journal of applied financial Economics, 8, 589-596.

[5] Gately, E. (1995). Neural Networks for financial forecasting. Wiley trader's exchange publishing company

[6] Gencay, R.(1999). Linear non-linear and essential foreign exchange prediction with simple technical rules. Journals of international Economics, 47, 91-107.

[7] Gencay, R. and Stengos, T. (1998). Moving average rules, volumes and predictability of security returns with feed forward neural networks. Journals of forecasting, 17, 401-414.

[8] Haefke, C. and Helmenstein, C., (1996a,1996b). Forecasting Austrian IPOs: An Application of Linear and Neural Network Error-Correction Models

[9] Kuan, C.M. and Liu, T. (1995).Forecasting exchange rates using feed forward and recurrent neural networks. Journal of Applied Econometrics 10(4), 347-364.

[10] Qi, M. and Madalla, G.S. (1999). Economic factors and stock market: A new perspective. Journal of forecasting,18, 151-166.

[11] Refenes, A. P N. (1995). "Neural Networks in the Capital Markets", Wiley & Sons, Chichester, ISBN 0-471-94364-9.

[12] Shadboth, J. and Taylor, J. (2002). neural networks and the financial markets. Springer publishing company Limited.

[13] Trippi, R. R. and Turban, E. (1993). Neural Networks in finance and investing: using artificial intelligence to improve real world performances. Probus publishing company (Chicago III).