UIN IMAM BONJOL PADANG
MAp Journal
Mathematics & Applications
Web : ejournal.uinib.ac.id/jurnal/index.php/MAp
Email : map_journal@uinib.ac.id

PERBANDINGAN AKURASI PERAMALAN ANTARA MODEL NEURAL NETWORK DAN REGRESI BERGANDA

COMPARISON OF ACCURACY BETWEEN NEURAL NETWORK AND MULTIPLE REGRESSION MODELS IN FORECASTING

Hermansah[1§], Muhammad Muhajir[2]

[1]Department of Mathematics, Universitas Riau Kepulauan, Batam [Email: hermansah@fkip.unrika.ac.id]
[2]Department Of Statistics, Universitas Islam Indonesia, Yogyakarta [Email: mmuhajir@uii.ac.id]
[§]*Corresponding Author*

Received 13 April 2023; Accepted 26 Juni 2023; Published 30 Juni 2023;

Abstrak

Penelitian ini membahas peramalan variabel MEDV data Boston yang dikumpulkan oleh Dinas Sensus AS mengenai nilai perumahan dengan menggunakan 13 variabel sebagai variabel prediktor. Metode yang digunakan adalah model *Feed Forward Neural Network* (FFNN) dan sebagai perbandingan dilakukan juga peramalan dengan menggunakan model regresi linier berganda. Hasil penelitian ini menunjukkan bahwa metode FFNN lebih baik dibandingkan dengan metode regresi linear berganda dalam peramalan variabel MEDV data Boston mengenai nilai perumahan berdasarkan kriteria nilai *Mean Square Error* (MSE) dan *Mean Absolute Percent Error* (MAPE). Hal ini dibuktikan dengan hasil nilai MSE dan MAPE dengan metode FFNN adalah 15,7518 dan 0,14563, sedangkan metode regresi linear berganda adalah 31,2630 dan 0,21040. Berdasarkan hasil penelitian ini, dapat disimpulkan bahwa metode FFNN yang memiliki nilai MSE dan MAPE semakin kecil, maka hasil ramalannya semakin tepat.

Kata Kunci: peramalan, *neural network*, FFNN, regresi linier berganda

Abstract

This research discusses the forecasting of the median value of owner occupied homes (MEDV) using all the other continuous variables available in the Boston dataset. The Boston dataset is a collection of data about housing values in the suburbs of Boston. The used method is Feed Forward Neural Network (FFNN) and the multiple linear regression method as a comparison. The result of the research indicates that the FFNN method is better than multiple linear regression in forecasting the median value of owner occupied homes (MEDV) using all the other continuous variables available in the Boston dataset. It is proven that the MSE and MAPE value of using the FFNN method is 15.7518 and 0.14563, whereas the value of multiple linear regression is 31.2630 and 0.21040. Based on this result, the research can be concluded that the FFNN method has the smaller MSE and MAPE values, the result of the forecasting is more accurate.

Keywords*: forecasting, neural network, FFNN, multiple linear regression*

1. Introduction

Neural network is one of the creation representations from the human brain that tries to stimulate learning processes in the human brain [1]. In general, the function of the human brain is divided into two parts: the study function and the thinking function. The human brain consists of billions of cells (called neuron) that process information. Each cell worked as a simple processor and interacted with each other and in parallel pairs in producing brain ability [2].

The structure of a neural network is similar to the human mind's structure. Haykin (2004) [3] said that a neural network is a machine designed to arrange the ways in which the human brain works in performing its functions or certain works. The neural network consists of some elements for processing information, which are called neurons. Neurons in a neural network are compiled in groups, which are called layers. The structure of neurons in the layers and the connection patterns in and inter-layers are called "network architecture" [4]. This architecture is one of the important characteristics that distinguish neural networks. Commonly, it consists of three layers that form a neural network. The three layers include the input layer, hidden layer, and output layer.

Units in the input layer are called units of input. The units of input accept the patterns of input from outside that create a problem. The total number of nodes or neurons in the input layer depends on how many inputs there are from the inner model, and each input determines one neuron [5]. A hidden layer is a layer that is unseen and invisible directly [6]. A hidden layer exists between the input layer and the output layer. The value derived from the output layer is the solution of a neural network to a problem. After passing the training process, the network gives a response of new input to produce output that is the result of the prediction [7].

Information (as input) is sent into a neuron through an input weight. This input is processed by a propagation function, which raises the value of input heaviness. Its result is then compared with the threshold by the activation function [8]. The activation function will determine whether or not the signal from neuron input will be continued [9]. There are some activation functions that are always used in neural networks: the threshold function, the linier function, and the sigmoid function [7].

The information sent in a neural network is propagated layer-by-layer starting from input into output without or through one or more hidden layers. It will depend on the algorithm used, and then the information can also be propagated to the backside (backpropagation) [10].

There are three types method of neural networks: Feed Forward Neural Network (FFNN), Radial Basis Function (RBF), and Kohonen Network (KN). One of the three methods is the FFNN method, which is mostly used to do prediction [11]. According to [12], the FFNN method can be viewed as one flexible class of non-linear functions. This method was echoed by David E. Rumelhat, Geoffrei E., Hinton, and Ronald J. William in 1986. The development of

the algorithm becomes the early development of research in neural networks [13]. William and Li, in the year 1998, conducted research by using a neural network with a backpropagation algorithm. Based on this research, the result presented a very good prediction [14].

Backpropagation is such a supervised algorithm using a weight adaptation pattern to reach a minimum error value between the output of the prediction result and the real value [15]. This algorithm covers three stages: feed forward from the input pattern, extrapolation and backpropogation from error, and weight adaptation. In the stage of feed forward, each unit accepts the input signal (y_i) and expose it to the hidden unit $z_1, \ldots, z_H$. Thus, all hidden units count the activation and then send the signal (z_h) to the output unit. Afterwards, the unit of output counts its activation and predicts the response variable value [10].

Amongst kinds of neural networks, the FFNN is a model much often used because it is well-known to have good approachability and a universal sense [16]. Besides that, this model is known for its quality, which has a prediction value approaching the actual value, so it produces a low error rate and has the ability to predict or do an analysis of very complex problems. This model doesn't have a certain requirement or assumption [2].

Linear regression is a statistical method used to form a model of correlation between a response variable and one or more predictor variables. If there is one independent variable, it is called a simple linear regression. If there is more than one independent variable, it is called a multiple linear regression. In line with its name, the correlation stated in this method is a linear correlation between the response variable and the predictor variable. The analysis of linear regression has some assumptions, which are identical model rate, independent, and normal distribution. This method is popular enough to describe the correlation between the response variable and the predictor variable because it has the best sense of an unbiased line estimator.

Both methods, the FFNN method and multiple linear regression, will be compared. The performance of each model formed is measured using the Mean Square Error (MSE) and Mean Absolute Percent Error (MAPE). The best model is indicated with the lowest MSE and MAPE values.

2. Methods

The data used in this study is secondary data, specifically MEDV data (median value of occupied houses) as a response variable, and the predictor variables are CRIM (crime rate), ZN (proportion of residential land), INDUS (proportion of non-retail businesses), CHAS (Charles River), NOX (concentration of nitric oxide), RM (average number of rooms per dwelling), AGE (proportion of units built before 1940), DIS (distance to lower status percentage of the population). This information will be processed using two methods: multiple linear regression and Feed Forward Neural Network. (FFNN).

2.1 FFNN Method

Neural networks have been successfully applied in both univariate and multivariate time series forecasting. The most widely employed architecture is the common multilayer perceptron (MLP). These well researched properties have been shown to generalize any linear or nonlinear functional relationship to any degree of accuracy without any prior assumptions about the underlying data generation process [17].

In forecasting, feed-forward architectures of MLPs are used to model nonlinear autoregressive NAR(p)-processes or NARX(p)-processes using external variables to code exogenous events as intervention variables [18]. Given a time series y, at a point in time t, a one-step ahead forecast $\hat{y}_{t+1}$ is computed using $p = I$ observations y_t, y_{t-1}, ..., y_{t-I+1} from I preceding points in time t, $t-1$, $t-2$, ..., $t-I+1$, with I denoting the number of input units of the neural network [19]. The functional forms is,

$$y = w_0 + \sum_{h=1}^{H} w_h \; f(v_{0h} + \sum_{i=1}^{I} v_{ih}\, y_i) \quad (1)$$

where $y = [y_i, \ldots, y_{t-I+1}]$ is the vector of the lagged observations (inputs) of the time series. The network weights are $w_H = [w_1, w_2, \ldots, w_h]$ and $v_{IH} = [v_{11}, v_{12}, \ldots, v_{ih}]$ for the output and the hidden layer respectively. The w_0 and v_{0h} are the biases of each neuron. I and H are the number of input and hidden units in the network and f is a non-linear transfer function, which is usually either the sigmoid function or the hyperbolic tangent function.

MLP is also known as a feed-forward neural network (FFNN) or nonlinear auto-regressive neural network (NARNN) or a back-propagation neural network (BPNN) [20]. The following an the explanation from the model of backpropagation.

Step 1 : Initiating weight (setting it to lower value randomly)

Step 2 : When the condition stops valuing wrongly, do:

a. For each training pair, do: Feed Forward

1. Each unit of input $(y_i, i = 1,2,\ldots,I)$ accepts the signal of input y_i and spreads this signal into all upper layer units (hidden units)

2. Each hidden unit $(z_h, h = 1,2,\ldots,H)$ counts up signal weight of input,

$$z\_in_h = v_{0h} + \sum_i v_{ih}\, y_i \quad (2)$$

and applies the activation function to count the signal of output,

$$z_h = f(z\_in_h) \quad (3)$$

and sends that signal into all upper layer units (units of output).

3. In unit output (y) counting up the weight signal of input,

$$y\_in_k = w_0 + \sum_h w_h\, z_h \quad (4)$$

and applies the activation function to count up the signal of output,

$$y = f(y\_in_k) \quad (5)$$

b. For each training, do: Backpropagation

1. In the unit output (y) accepting a target pattern matching the input training pattern, counting mistaken information,

$$\delta_k = (t_k - y) f'(y\_in_k) \quad (6)$$

then counting weight correction (used to repair

w_h),

$$\Delta w_h = \alpha \delta_k z_h \quad (7)$$

and finally, counting bias correction bias (used to repair w_0),

$$\Delta w_0 = \alpha \delta_k \quad (8)$$

afterwards, sending it to the most upper layer units

2. Each hidden unit $(z_h, h = 1,2,\dots,H)$ counting input delta (from the most upper unit),

$$\delta\_in_h = \sum_{h=1}^{H} \delta_k \, w_h \quad (9)$$

times the score with activation function derivation to count mistaken information,

$$\delta_h = \delta\_in_h f'(z\_in_h) \quad (10)$$

then count the weight repairs (used to revise v_{ih}),

$$\Delta v_{ih} = \alpha \delta_h y_i \quad (11)$$

after that, correct the bias correction (used to revise v_{0h}),

$$\Delta v_{0h} = \alpha \delta_h \quad (12)$$

c. Repairing the weight and bias

In units of output (y) repairing weight and bias $(h = 0,1,\dots,H)$,

$$w_h(t+1) = w_h(t) + \Delta w_h \quad (13)$$

Each hidden unit $(z_h, h = 1,2,\dots,H)$ repairs weight and bias, with $(i = 0,1,\dots,I)$,

$$v_{ih}(t+1) = v_{ih}(t) + \Delta v_{ih} \quad (14)$$

d. Testing the stopping condition

The stopping condition used epoch maximum and also could use the MSE error target, if the MSE target reached the error target, the training process would stop [3].

2.2 Multiple Regression Method

Simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables, an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

Multiple regression, also known simply as multiple linear regression (MLR), is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable. Formula and calculation of multiple linear regression are as follows:

$$y_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_p x_{ip} + \varepsilon \quad (15)$$

where $i = n$ observations, y_i dependent variable, x_i explanatory variables, α_0 y-intercept (constant term), α_p slope coefficients for each explanatory variable, and ε the model's error term (also known as the residuals). The multiple regression model is based on the following assumptions:

a. There is a linear relationship between the dependent variables and the independent variables.

b. The independent variables are not too highly correlated with each other.

c. y_i observations are selected independently and randomly from the population.

d. Residuals should be normally distributed with a mean of 0 and variance σ.

2.3 Forecast Measure

Two forecast error measurements, namely Mean Squared Error (MSE) and Mean Absolute Percent Error (MAPE), were used to evaluate the forecast accuracy. MSE is defined as follows:

$$MSE = \sum_{t=1}^{N} \frac{(A_t - F_t)^2}{N} = \sum_{t=1}^{N} \frac{{e_t}^2}{N} \qquad (16)$$

where e is error and N is the number of data.

MAPE is defined as follows:

$$MAPE = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{A_t - F_t}{A_t} \right| \qquad (17)$$

where A_t is the actual value at data time t and F_t is forecast value at data time t [21].

3. Results and Discussion

Basically, to form the established network, it is started by determining the network input. Afterwards, it needs to estimate much more neuron that exists in hidden layers with weight estimation processes. As a starting point, it needs to divide the data into two parts: first, training data, and second, testing data. The next step is the data normalization and heaving process.

a. Normalization of the Data

Normalization of data was done to display the input data and target within a certain range. The instruction for normalization in the R program is:

```
# Set a seed
> set.seed(500)
> library(MASS)
> data <- Boston
# Check that no data is missing
> apply(data,2,function(x) sum(is.na(x)))
# Train-test random splitting
> index <- sample(1:nrow(data),
round(0.75*nrow(data)))
> train <- data[index,]
> test <- data[-index,]
# Neural net fitting
# Scaling data for the NN
> maxs <- apply(data, 2, max)
> mins <- apply(data, 2, min)
> scaled <- as.data.frame(scale(data, center =
mins, scale = maxs - mins))
# Train-test split
> train_ <- scaled[index,]
> test_ <- scaled[-index,]
```

b. Weight Estimation

In a backpropagation network, the learning was done by initial weight estimation of both the initial weight and final weight and the determination of the learning parameter. The weight lifting is to maximize the network performance function. The performance function of backpropagation is the MSE value, which is counted from the mean of the error quadrat that happens between the output network and target.

In the R program, to build the backpropagation network of the research, the instructions use “neuralnet”. The used function is as follows:

```
# NN training
> library(neuralnet)
> n <- names(train_)
> f <- as.formula(paste("medv ~", paste(n[!n
%in% "medv"], collapse = " + ")))
```

```
> nn <- neuralnet(f, data=train_, hidden=c(5,3),
linear.output=T)
# Visual plot of the model
> plot(nn)
```

The architecture of backpropagation with 13 inputs and two hidden layers (in the first layer there are five neurons and in the second layer there are two layers), and one output can be seen in Figure 1 as follows.

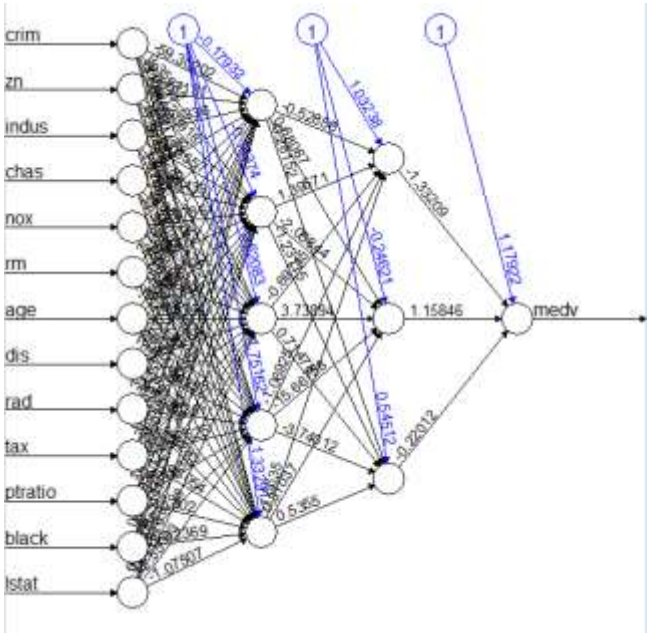

Figure 1. The graphical representation of the model with the weights on each connection

c. Renormalization

The data that are normalized will be returned into the initial form that is often called by renormalizations. The instructions for transforming the data in the R program are as follows:

```
# Predict
> pr.nn <- compute(nn,test_[,1:13])
# Results from NN are normalized (scaled)
# Descaling for comparison
> pr.nn_ <- pr.nn$net.result*(max(data$medv)-
min(data$medv))+min(data$medv)
> test.r <- (test_$medv)*(max(data$medv)-
min(data$medv))+min(data$medv)
```

Based on the architecture formed, we obtained the prediction result from the backpropagation model. The comparison of the prediction results of backpropagation and multiple linear regression was achieved using the following plots as follows:

```
# Fitting linear model
> lm.fit <- glm(medv~., data=train)
> summary(lm.fit)
# Predicted data from lm
> pr.lm <- predict(lm.fit,test)
# Calculating MSE
# Test MSE
> MSE.lm <- sum((pr.lm -
test$medv)^2)/nrow(test)
> MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)
# Compare the two MSEs
> print(paste(MSE.lm,MSE.nn))
# Plot predictions
> par(mfrow=c(1,2))
> plot(test$medv,pr.nn_,col='red',main='Real vs
predicted NN',pch=18,cex=0.7)
> abline(0,1,lwd=2)
> legend('bottomright', legend='NN', pch=18,
col='red', bty='n')
> plot(test$medv,pr.lm,col='blue',main='Real vs
predicted lm',pch=18, cex=0.7)
> abline(0,1,lwd=2)
> legend('bottomright', legend='LM', pch=18,
col='blue', bty='n', cex=.95)
```

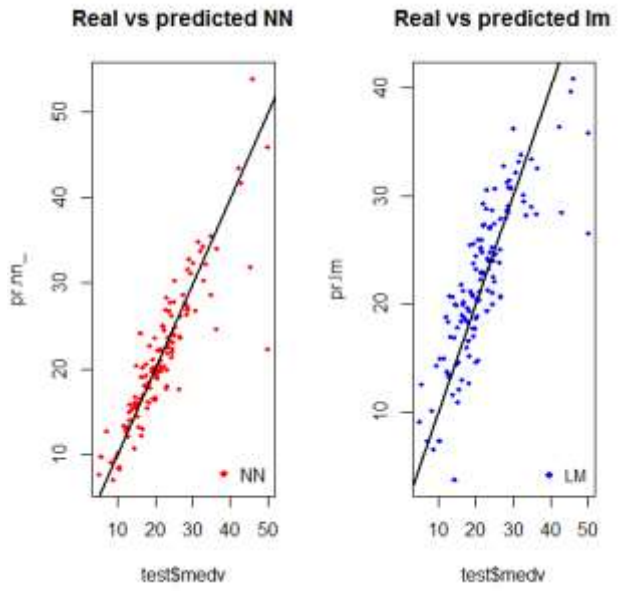

Figure 2. The graphical representation of the model with the weights on each connection

```
# Compare predictions on the same plot
> par(mfrow=c(1,1))
> plot(test$medv,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
> points(test$medv, pr.lm, col='blue', pch=18, cex=0.7)
> abline(0,1,lwd=2)
> legend('bottomright', legend=c('NN','LM'), pch=18, col=c('red','blue'))
```

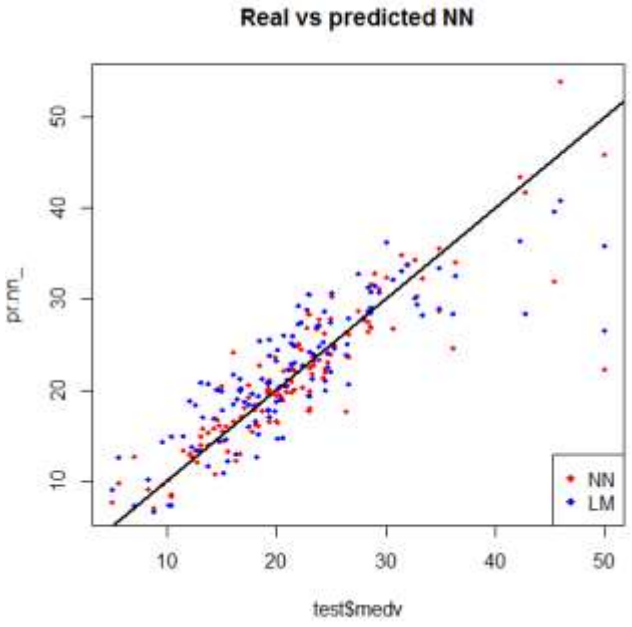

Figure 3. The graphical representation of the model with the weights on each connection

4. Conclusion

Based on the result of the analysis, the score of Mean Square Error (MSE) of Feed Forward Neural Network (FFNN) model was 15.7518 and the score of multiple linear regression was 31.2630. At the same time, the score of Mean Absolute Percent Error (MAPE) in the FFNN model was 0.14563 and the score of multiple linear regression was 0.21040. The FFNN model yielded the lower MSE and MAPE values. So, it can be concluded that the prediction of owner occupied homes (MEDV) using all the other continuous variables available in the Boston dataset is the best predictor when using the FFNN method.

References

[1] Sari, R., Gunawan. 2015. Arsitektur Jaringan Saraf Tiruan Untuk Pemodelan Proses Ekstraksi Aturan dengan Search Tree. *Seminar Nasional Inovasi Desain dan Teknologi*, 306-313.

[2] Setiawan, W. 2008. Prediksi harga saham menggunakan jaringan syaraf tiruan multilayer feedforward network dengan algoritma backpropagation. *Konferensi Nasional Sistem dan Informatika*, 108-113.

[3] Haykin, S. 2004. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 2nd Edition.

[4] Sari, I. P., Wuryandari, T., Yasin, H. 2014. Prediksi Data Harga Saham Harian Menggunakan Feed Forward Neural Networks (FFNN) dengan Pelatihan Algoritma Genetika (Studi Kasus pada Harga Saham Harian PT. XL Axiata Tbk). *Jurnal Gaussian*, 3(3), 441-450.

[5] Faraway, J., Chatfield, C. 1998. Time series

forecasting with neural networks: a comparative study using the air line data. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 47(2), 231-250.

[6] Kao, J. J., Huang, S. S. 2000. Forecasts using neural network versus Box-Jenkins methodology for ambient air quality monitoring data. *Journal of the Air & Waste Management Association*, 50(2), 219-226.

[7] Kusumadewi, F. 2014. *Peramalan Harga Emas menggunakan Feedforward Neural Network dengan Algoritma Backpropagation*. Yogyakarta: Universitas Negeri Yogyakarta.

[8] Suhartono, S. 2005. Neural Networks, ARIMA and ARIMAX Models for Forecasting Indonesian Inflation. *Widya Journal of Management and Accounting*, 5(3), 219732.

[9] Siang, J. J. 2005. *Jaringan syaraf tiruan dan pemrogramannya menggunakan Matlab*. Penerbit Andi, Yogyakarta.

[10] Warsito, B. 2006. Perbandingan Model Feed Forward Neural Network dan Generalized Regression Neural Network pada Data Nilai Tukar Yen terhadap Dolar AS. *Prosiding SEMINAR NASIONAL SPMIPA*, 1(1), 127-131.

[11] Meinanda, M. H., Annisa, M., Muhandri, N., Suryadi, K. 2009. Prediksi masa studi sarjana dengan artificial neural network. *Internetworking Indonesia Journal*, 1(2), 31-35.

[12] Suhartono, S. 2007. *Feedforward Neural Networks untuk Pemodelan Runtun Waktu*. Yogyakarta: Universitas Gadjah Mada.

[13] Fajar, M., Jatmiko, W. 2011. Implementasi Feedforward Neural Network pada Field Programmable Gate Array untuk Mendeteksi Sleep Apnea menggunakan Data Ekstraksi ECG. *Konferensi Nasional Sistem dan Informatika*, 290-296.

[14] Farber, R., Lapedes, A. 2008. How Neural Nets Works. *Evolution, Learning, and Cognition*, 331-345.

[15] Bambang, B., Widodo, R. J., Sutalaksana, I. Z., Singgih, M. L. 2004. Teknik Jaringan Syaraf Tiruan Feedforward Untuk Prediksi Harga Saham Pada Pasar Modal Indonesia. *Jurnal Informatika*, 1(1), 33-37.

[16] Handaga, B., Asy'ari, H. 2012. Kombinasi Algoritma Cuckoo-Search Dan Levenberg-Marquadt (CS-LM) pada Proses Pelatihan Artificial Neural Network (ANN). *Simposium Nasional RAPI XI FT UMS*.

[17] Fausett, L. V. 2006. *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India.

[18] Hermansah, Rosadi, D., Abdurakhman, Utami, H. 2020. Selection of input variables of nonlinear autoregressive neural network model for time series data forecasting. *Media Statistika*, 13(2), 116-124.

[19] Hermansah, Rosadi, D., Abdurakhman, dan Utami, H. 2021. Automatic time series forecasting using nonlinear autoregressive neural network model with exogenous input. *Bulletin of Electrical Engineering and Informatics*, 10(5), 2836-2844.

[20] Hermansah, Rosadi, D., Abdurakhman, dan Utami, H. 2021. A comparison of learning algorithms for seasonal time series forecasting using NARX model. *Journal of Mathematical and Computational Science*, 11(6), 6638-6656.

[21] Hermansah, Rosadi, D., Abdurakhman, dan Utami, H. 2021. Time series forecasting with trend and seasonal patterns using NARX network ensembles. *Mathematics and Statistics*, 9(4), 511-520.