Advances in Social Science, Education and Humanities Research, volume 528
Proceedings of the 7th International Conference on Research, Implementation, and Education of Mathematics and Sciences (ICRIEMS 2020)

Multivariate Time Series Data Forecasting Using Multi-Output NARNN Model

Hermansah[1,2,*] Dedi Rosadi[1] Abdurakhman[1] Herni Utami[1] Gumgum Darmawan[1,3]

[1] *Department of Mathematics, Gadjah Mada University, Yogyakarta, Indonesia*
[2] *Department of Mathematics Education, Riau Kepulauan University, Batam, Indonesia*
[3] *Department of Statistics, Padjadjaran University, Bandung, Indonesia*
[*] *Corresponding author. Email: bankhermansah@gmail.com*

ABSTRACT
This research proposes the multi-output Nonlinear Autoregressive Neural Network (NARNN) method to forecast multivariate time series data containing the input layer, one hidden layer, and the output layer. The multi-output NARNN method is performed by applying the logistic activation function and the resilient backpropagation learning algorithm. The stage of determining the input variable is chosen based on the number of data frequencies. The number of neurons in the hidden layer is half of the number of input variables. Simulation and empirical studies are conducted to test whether the proposed method works well for multivariate time series data forecasting. The simulation results show that the best performance is the simulation data generated from the MESTAR nonlinear model. The simulation study results are as expected. Empirical studies on Indonesia's inflation and Bank Indonesia interest rate data show that the multi-output NARNN method provides better forecasting accuracy than the VAR, VMA, and VARMA methods with a total MSE value of 0.054655 and a total MAPE of 0.026853 in the testing data.

Keywords: *Multivariate Time Series Forecasting, Logistic Function, Resilient Backpropagation Learning, Multi-Output NARNN Model.*

1. INTRODUCTION

In daily life, time-series data are often found consisting of many interrelated variables known as multivariate time series data. Statistical methods that have been widely used for modeling and forecasting multivariate time series data are the Vector Autoregressive (VAR), Vector Moving Average (VMA), and Vector Autoregressive Moving Average (VARMA) models. These models are linear and require assumptions such as data stationarity [1]. This assumption is not practical considering the movement of time series data, especially in the financial sector, is very dynamic.

Nowadays, a more flexible approach has been developed to model linear and nonlinear relationships known as Neural Network (NN) model. The NN model is an alternative that is widely used because it does not require assumptions on data that are often difficult to fulfill [2]. The most popular and widely used NN model for modeling and forecasting time series data is the Nonlinear Autoregressive Neural Network (NARNN) model, also known as the Multilayer Perceptron (MLP) or Feedforward Neural Network (FFNN). In its application, the NARNN model contains a limited number of parameters. How to get the appropriate NARNN model, namely, how to determine the right combination between the number of input variables and the number of neurons in the hidden layer [3].

The [4] introduced the NARNN model for multivariate time series data forecasting, which in this study is called the multi-output NARNN method. They use statistical concepts to determine the multi-output NARNN method that is most suitable for multivariate data. They used two statistical tests, namely the Wald test, to obtain optimal input variables and the F test to determine the number of neurons in the hidden layer. This study proposes obtaining input variables through [5] approach based on the number of data frequencies without stepwise selection. If the data frequency is m (m > 3), then consecutive m lag starting from lag 1 will be taken. For example, for monthly data, 1:12 lag will be taken. However, if the condition m > 3 is not met, 1:4 lag is used. The [5] was further extended in [6] by implementing a combination of learning algorithms, activation functions, and ensemble operators. The [6]

approach was used the stepwise selection to obtain the most suitable NARNN model on univariate data. As an extension, this research was conducted, and the number of neurons in the hidden layer is half of the number of input variables. The multi-output NARNN method is performed by applying the logistic activation function and the resilient backpropagation learning algorithm.

Furthermore, simulation and empirical studies are carried out to test whether the proposed method works well for forecasting multivariate time series data. For reasons of simplicity of the model, research is carried out for bivariate cases. Three simulation data models are generated: the linear VAR model, the nonlinear Multivariate Exponential Smooth Transition Autoregressive (MESTAR), and MIXED (mixed of both). Empirical studies were conducted on Indonesia's inflation and Bank Indonesia's interest rate data. Measuring the forecasting accuracy is done with the value of Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE). MAPE is the average of the overall percentage of error between the actual data and forecasting data. A method has excellent performance if the MAPE value is below 10% and has good performance if the MAPE value is between 10% and 20% [7]. Besides, forecasting accuracy with the VAR, VMA, and VARMA methods is also provided.

2. RESEARCH METHOD

2.1. Multi-Output NARNN Model

The multi-output NARNN modeling architecture has as many neurons in the output layer as the multivariate series used. Suppose the weight from the input layer to the hidden layer is denoted as v and the weight from the hidden layer to the output layer is w, and $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_h)$ and $\beta = (\beta_1, \beta_2, \dots, \beta_m)$ are the bias vectors in input and hidden layers. The output of the multi-output NARNN method can be written as follows:

$$Z_t = w\, F(Zv + \alpha) + \beta + \varepsilon_t \tag{1}$$

and

$$F(Zv + \alpha) = \frac{1}{1+\exp(-(Zv+\alpha))} \tag{2}$$

For example, $Z_t = (Z_{1,t}, \dots, Z_{m,t})$ is a time series process consisting of m variables that are affected by lag-p values. The input vector can be written:

$$Z = (Z_{1,t-1}, \dots, Z_{1,t-p}, \dots, Z_{m,t-1}, \dots, Z_{m,t-p}) \tag{3}$$

Furthermore, there are $p \times m$ neurons in the input layer. If the scalar h denotes the number of hidden units, the matrix weight (network parameter) for the hidden layer has the dimension $(p \times m) \times h$, where

$$v = \begin{bmatrix} v_{1,t-1,1} & v_{1,t-1,2} & \cdots & v_{1,t-1,h} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1,t-p,1} & v_{1,t-p,2} & \cdots & v_{1,t-p,h} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m,t-1,1} & v_{m,t-1,2} & \cdots & v_{m,t-1,h} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m,t-p,1} & v_{m,t-p,2} & \cdots & v_{m,t-p,h} \end{bmatrix} \tag{4}$$

The input unit constants are involved in the architecture and connected to each neuron in the hidden and output layers. This generates a bias vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_h)$ in the hidden layer and $\beta = (\beta_1, \beta_2, \dots, \beta_m)$ in the output layer. Since there are m variables in the output layer, the output layer's matrix weight will be as follows:

$$w = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,h} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,h} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,h} \end{bmatrix} \tag{5}$$

Then the output from the multi-output NARNN method can be defined as follows:

$$Z_t = w\, F(Zv + \alpha) + \beta + \varepsilon_t$$

Or more explicitly,

$$Z_{l,t} = \sum_{k=1}^{h} w_{l,k}\, F_k\left(\sum_{i=1}^{m}\sum_{j=1}^{p} v_{i,t-j,k}\, Z_{i,t-j} + \alpha_k\right) + \beta_l + \varepsilon_{l,t}, \text{ for } l = 1, \dots, m \tag{6}$$

Where $\varepsilon_t = (\varepsilon_{1,t}, \dots, \varepsilon_{m,t})$ is the error vector, and F is the transfer function operated to the vector element $Zv + \alpha$. The function that is commonly used is the logistics function. The F function in the above model becomes

$$F_k\left(\sum_{i=1}^{m}\sum_{j=1}^{p} v_{i,t-j,k}\, Z_{i,t-j} + \alpha_k\right) = \frac{1}{1+\exp\left(-\left(\sum_{i=1}^{m}\sum_{j=1}^{p} v_{i,t-j,k} Z_{i,t-j} + \alpha_k\right)\right)} \tag{7}$$

In this study, the input variable was taken based on the number of data frequencies. If the data frequency is $m > 3$, then consecutive lag m from lag 1 will be taken. For example, for monthly data, 1:12 lag will be taken. However, if the condition $m > 3$ is not met, the 1:4 lag is used. For descriptions and implementation details of determining input variables, see [5, 6, 8]. Furthermore, the number of neurons in the hidden layer is half of the number of input variables. The multi-output NARNN method is carried out by implementing the logistic activation function and the resilient backpropagation learning algorithm.

2.2. Logistic Function

Each neuron contains an activation function and a threshold value, which is the minimum needed by input to activate it. The activation function is applied, and the output is passed to the next neurons in the networks. It is designed to ensure the neuron's output limitation, usually to values between 0 to 1 or -1 to $+1$. In most cases, the same function is used for every neuron in a network.

The logistic function is also known as the sigmoid function. The activation tool is most commonly applied due to its ability to obtain a real-valued number and crushes it into numbers ranging from 0 to 1. This involves classifying large negative numbers as 0, while positive ones are 1 using the following equation.

$$f(x) = \frac{1}{1+\exp(-cx)} \tag{8}$$

where parameter c is a constant, the function became popular partly because it is possible to interpret the function's output as the probability of the artificial neuron ring.

2.3. Resilient Backpropagation Learning

Resilient backpropagation is defined as a significant new learning scheme applied to adapt the weight step concerning the local gradient information directly. An individual update-value $\Delta_{i,j}$ was introduced by [9] to ensure each weight strictly decides the weight-update size. The evolution of this adaptive update-value was observed in the learning process because of the local sight on the error function, E, through the learning-rule presented as follows:

$$\Delta_{i,j}^{(t)} = \begin{cases} \eta^+ * \Delta_{i,j}^{(t-1)}, & if \ \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} > 0 \\ \eta^- * \Delta_{i,j}^{(t-1)}, & if \ \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0 \\ \Delta_{i,j}^{(t-1)}, & else \end{cases} \tag{9}$$

where $0 < \eta^- < 1 < \eta^+$.

The working principle of the adaptation-rule is such that each time there is a change in the sign of the corresponding weight $w_{i,j}$ partial derivative to show the last update was too enormous and that there is a jump of the algorithm on a local minimum, there is usually a reduction in the update-value $\Delta_{i,j}$ by the factor η^-. Meanwhile, in a situation the sign is retained, there is usually a slight increase in the update-value to ensure the acceleration of convergence in shallow regions.

The weight-update is also usually a straight-forward rule after the adaption of the update-value for each weight, and, in a situation the derivative is positive (increasing error), there is a reduction in the weight using the update-value, but an adverse condition usually leads to the update-value addition

$$\Delta w_{i,j}^{(t)} = \begin{cases} -\Delta_{i,j}^{(t)}, & if \ \frac{\partial E}{\partial w_{i,j}}^{(t)} > 0 \\ +\Delta_{i,j}^{(t)}, & if \ \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0 \\ 0, & else \end{cases} \tag{10}$$

$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \Delta w_{i,j}^{(t)} \tag{11}$$

There is, however, one exception, and this is a situation there is a change in the sign on the partial derivative such that the initial step was observed to be too large while the minimum was missed, there is going to be the reversion of the previous weight-update

$$\Delta w_{i,j}^{(t)} = -\Delta w_{i,j}^{(t-1)}, \ if \ \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0 \tag{12}$$

The backtracking weight-step is expected to change the sign on the derivative once again in the following step without allowing the update-value to be adapted to avert double punishment. It is possible to practically achieve this through the setting of $\frac{\partial E}{\partial w_{i,j}}^{(t-1)} := 0$ in the Δ_{ij} previous adaptation-rule. Meanwhile, there are changes in the update-values and weights each time there is a presentation of the whole pattern set once to the network (learning by epoch). It is, however, possible to use resilient backpropagation with and without weight backtracking. In this study applied resilient backpropagation with weight backtracking. A detailed description of the learning algorithm can be seen in [9, 10].

2.4. Forecast Measure

In order to evaluate the forecast accuracy of the models, two forecast error measurements are used: Mean Squared Error (MSE) and Mean Absolute Percent Error (MAPE). MSE is defined as follows:

$$MSE = \sum_{t=1}^{N} \frac{(A_t - F_t)^2}{N} = \sum_{t=1}^{N} \frac{e_t^2}{N} \tag{13}$$

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

MAPE is defined as follows:

$$MAPE = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{A_t - F_t}{A_t} \right| \tag{14}$$

where A_t is actual values at data time t and F_t is forecast values at data time t.

3. RESULTS

3.1. Simulation Study

Simulation studies are conducted to see the multi-output NARNN method's accuracy in modeling and forecasting multivariate time series data. Furthermore, simulations are carried out for the bivariate case for reasons of model simplicity. Simulation data is generated from three models, namely linear of Vector Autoregressive (VAR), nonlinear of Multivariate Exponential Smooth Autoregressive Transition (MESTAR), and MIXED (mixed of both). Simulation data consists of 400 data divided into training and testing data. Guidelines for sharing training and testing data, in general, have not been standardized. This study uses 350 data for training and the rest for testing. The software used is software R.

Mathematically, the simulation data generated from VAR linear model in this study are:

$$x_{1,t} = 0.3\, x_{1,t-1} + 0.5\, x_{2,t-1} + u_{1,t}$$
$$x_{2,t} = -0.7\, x_{1,t-1} + 0.8\, x_{2,t-1} + u_{2,t}$$

MESTAR nonlinear model simulation data, namely:

$$x_{1,t} = 6.5\, x_{1,t-1} \exp\left[-0.25\, x_{1,t-1}^2\right] + 3.5\, x_{2,t-1} \exp\left[-0.45\, x_{2,t-1}^2\right] + u_{1,t}$$
$$x_{2,t} = 4.3\, x_{1,t-1} \exp\left[-0.15\, x_{1,t-1}^2\right] + 3.7\, x_{2,t-1} \exp\left[-0.25\, x_{2,t-1}^2\right] + u_{2,t}$$

Simulation data for the MIXED model, namely:

$$x_{1,t} = -0.8\, x_{1,t-1} + u_{1,t}$$
$$x_{2,t} = 2.9\, x_{1,t-1} \exp\left[-0.15\, x_{1,t-1}^2\right] + 3.1\, x_{2,t-1} \exp\left[-0.25\, x_{2,t-1}^2\right] + u_{2,t}$$

with $u_t \sim IIDN(0, 0.5^2)$.

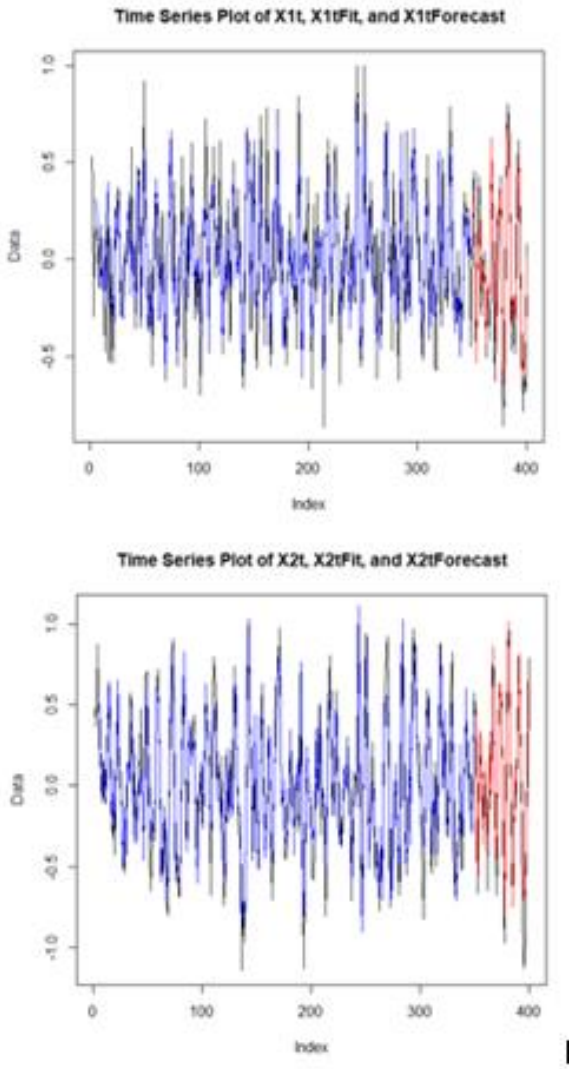

Figure 1 The plot of the results of the multi-output NARNN method on the simulation data of the VAR linear model.

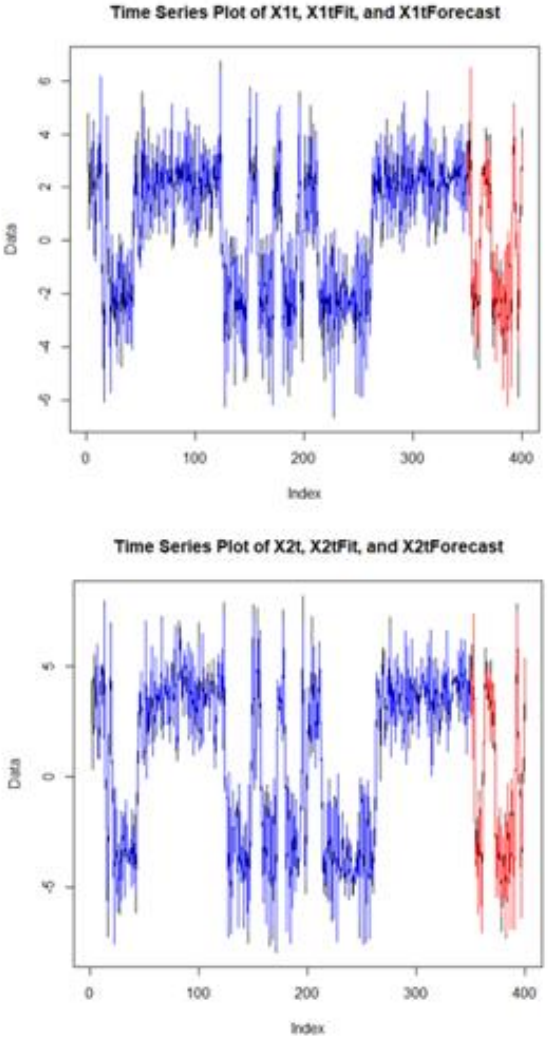

Figure 2 The plot of the results of the multi-output NARNN method on the simulation data of the MESTAR nonlinear model.

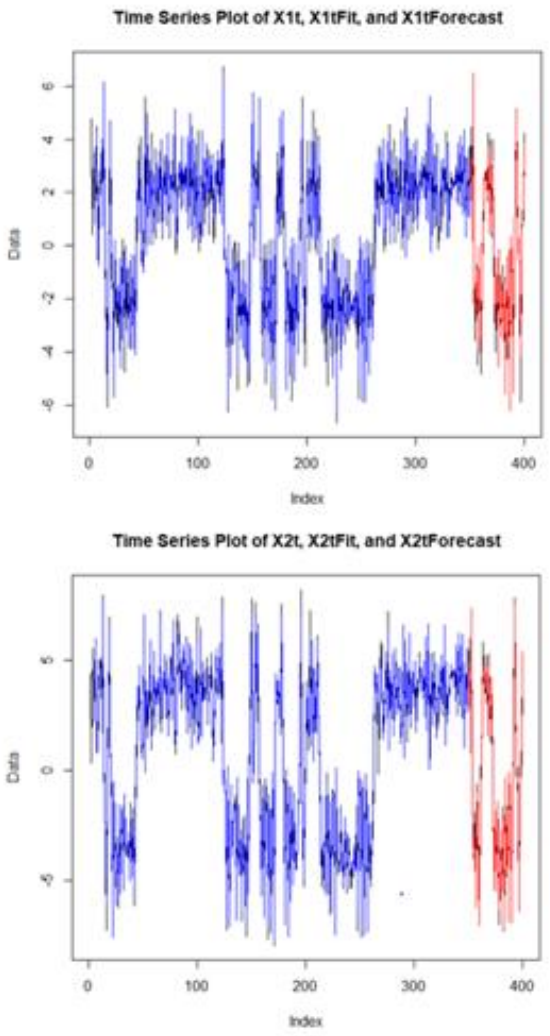

Figure 3 The plot of the results of the multi-output NARNN method on the simulation data of the MIXED model.

The multi-output NARNN method uses one hidden layer where the number of hidden neurons is half the input variables. The number of neurons in the output layer is as much as the multivariate time series data used. In this study, the input variable was taken based on the number of data frequencies. The number of frequencies in the simulation data is 1, then the lag 1:4 is taken on the $x_{1,t}$ and $x_{2,t}$ variables, namely $x_{1,t-1}$, $x_{1,t-2}$, $x_{1,t-3}$, $x_{1,t-4}$, $x_{2,t-1}$, $x_{2,t-2}$, $x_{2,t-3}$, and $x_{2,t-4}$. The simulation study provides a multi-output NARNN architecture with the input variable being 8 neurons in the input layer, 4 neurons in the hidden layer, and 2 neurons in the output layer. The study results from the three models' simulation data are illustrated in Figure 1, Figure 2, and Figure 3. In the figure, the black line is the original simulation data, blue is the result of the training data, and red is the result of the testing data. It can be seen that the data plots of the two variables generated by the multi-output NARNN method are close to the original simulation data. These results indicate that the multi-output NARNN method can work well for forecasting simulation data of the linear VAR, nonlinear MESTAR, and MIXED models.

In the end, the comparison of forecasting errors for the multi-output NARNN method from the three simulation data uses the MAPE. The results of the comparison of forecasting errors can be seen in Table 1. Based on the MAPE values in the table, it can be concluded that the multi-output NARNN method provides the best performance on simulation data generated from the MESTAR nonlinear model.

3.2. Empirical Study

Empirical studies were conducted on data on the Indonesian inflation rate and the Bank Indonesia interest rate. Data used are monthly data observed from July 2005 to August 2016. The model formation was carried out on the first 129 data from July 2005 to March 2016 (training data), and the last 5 data from April 2016 to August 2016 used to evaluate the accuracy of forecasting (testing data).

The multi-output NARNN architecture provides 24 neurons in the input layer, 12 neurons in the hidden layer, and 2 neurons in the output layer based on the monthly data frequency. The plot of the multi-output NARNN method on data on the Indonesian inflation rate and the Bank Indonesia interest rate is illustrated in Figure 4. The black line is the original simulation data, blue is the result of the training data, and red is the result of the testing data.

Table 1. MSE and MAPE results of the multi-output NARNN method on simulation data.

Simulation data	MSE of testing data			MAPE of testing data		
	$x_{1,t}$	$x_{2,t}$	Total	$x_{1,t}$	$x_{2,t}$	Total
VAR	0.046351	0.042731	0.044541	4.613215	1.454693	3.033954
MESTAR	0.257786	0.381383	0.319584	0.732585	0.577299	0.654942
MIXED	0.048242	0.053111	0.050676	1.219823	0.404704	0.812263

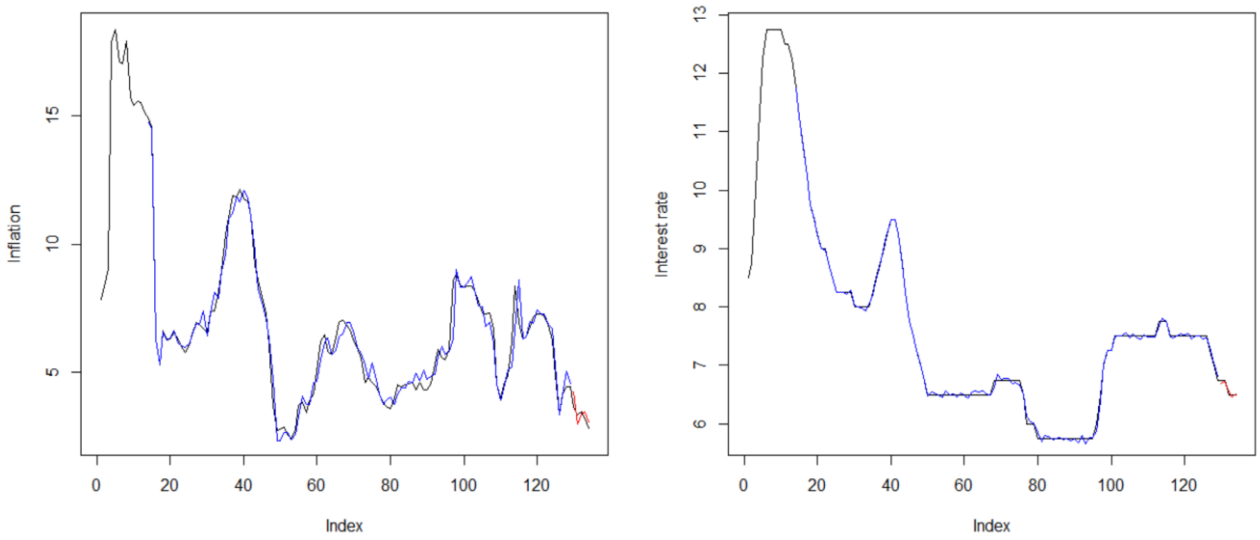

Figure 4 The plot of the multi-output NARNN method on data on the Indonesian inflation rate and the Bank Indonesia interest rate.

Table 2. The comparison between the multi-output NARNN, VAR, VMA and VARMA methods on the Indonesian inflation rate and the Bank Indonesia interest rate.

Method	MSE of testing data			MAPE of testing data		
	Inflation	Interest rate	Total	Inflation	Interest rate	Total
Multi-output NARNN	0.101603	0.007708	0.054655	0.045213	0.008494	0.026853
VAR(2)	0.056952	0.041704	0.049328	0.060560	0.029746	0.045153
VMA(3)	0.450419	0.015718	0.233068	0.200310	0.011816	0.106063
VARMA(1,1)	0.059634	0.050119	0.054876	0.064748	0.032761	0.048754

The multi-output NARNN method's accuracy is compared with the VAR, VMA, and VARMA methods. The comparison results of the four methods can be seen in Table 2. The multi-output NARNN, VAR, and VARMA methods give a total MAPE value below 10%, so it can be concluded that the three methods have excellent performance. Simultaneously, the VMA method has good performance because the total MAPE value is between 10% and 20%. Compared to the multi-output NARNN method with the VAR, VMA, and VARMA methods, the multi-output NARNN method is superior because it provides the smallest total MAPE value.

4. CONCLUSION

The NARNN model for multivariate time series data in this study is called the multi-output NARNN method, which contains an input layer, one hidden layer, and an output layer. This modeling's architecture has as many neurons in the output layer as the multivariate time series used. The activation function used is a logistic function at the hidden layer and a linear function at the output layer. Parameter estimation is done by applying a resilient backpropagation learning algorithm, and input variables are taken based on the number of data frequencies. Simulation and empirical studies were conducted to test whether the proposed multi-output NARNN method works well for modeling multivariate time series data. The simulation study provides a multi-output NARNN architecture with the input variable being 8 neurons in the input layer, 4 neurons in the hidden layer, and 2 neurons in the output layer. The simulation results show that the best performance is the simulation data generated from the MESTAR nonlinear model. The simulation study results are as expected. Furthermore, empirical studies on the Indonesian inflation rate and the Bank Indonesia interest rate provide 24 neurons in the input layer, 12 neurons in the hidden layer, and 2 neurons in the output layer. The empirical studies show that the multi-output NARNN method provides better forecasting accuracy than the VAR, VMA, and VARMA methods.

ACKNOWLEDGMENTS

We acknowledge the receipt of research funding PDD 2020 from Deputy of Research and Development, Ministry of Research and Technology / National Agency for Research and Innovation of Republic of Indonesia.

REFERENCES

[1] R.S. Tsay, Multivariate Time Series Analysis: With R and Financial Applications, John Wiley & Sons, 2014. URI: https://www.wiley.com/en-us/Multivariate+Time+Series+Analysis%3A+With+R+and+Financial+Applications-p-9781118617908

[2] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018. URI: https://otexts.org/fpp2/

[3] Suhartono, Subanar, S. Guritno, Model Selection in Neural Networks by Using Inference of $R^2_{Incremental}$, PCA, and SIC Criteria for Time Series Forecasting, Journal of Quantitative Methods: Journal Devoted to the Mathematical and Statistical Application in Various Fields 2(1) (2006) 41–57. URI: https://repository.ugm.ac.id/id/eprint/32902

[4] D.U. Wutsqa, Seleksi Model Neural Network Menggunakan Inferensi Statistik dari $R^2_{Increment}$ dan Uji Wald untuk Peramalan Time Series Multivariat, Pythagoras 4(2) (2008) 91–103. URI: https://journal.uny.ac.id/index.php/pythagoras/article/view/564/422

[5] S.F. Crone, N. Kourentzes, Feature Selection for Time Series Prediction - A Combined Filter and Wrapper Approach for Neural Networks, Neurocomputing 73(10) (2010) 1923–1936. DOI: https://doi.org/10.1016/j.neucom.2010.01.017

[6] Hermansah, D. Rosadi, Abdurakhman, H. Utami, Selection of Input Variables of Nonlinear Autoregressive Neural Network Model for Time Series Data Forecasting, Media Statistika 13(2) (2020) 116–124. DOI: https://doi.org/10.14710/medstat.13.2.116-124

[7] R.J. Hyndman, A.B. Koehler, Another Look at Measures of Forecast Accuracy, International Journal of Forecasting 22(4) (2006) 679–688. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001

[8] N. Kourentzes, D.K. Barrow, S.F. Crone, Neural Network Ensemble Operators for Time Series Forecasting, Expert Systems with Applications 41(9) (2014) 4235–4244. DOI: https://doi.org/10.1016/j.eswa.2013.12.011

[9] M. Riedmiller, H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm, IEEE International Conference on Neural Networks 1(1) (1993) 586–591. DOI: https://doi.org/10.1109/ICNN.1993.298623

[10] M. Riedmiller, RPROP: Description and Implementation Details, University of Karlsruhe, 1994.