SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2024
ORGANIZATION AND MANAGEMENT SERIES NO. 195

SIMULATION ANALYSIS OF ARTIFICIAL NEURAL NETWORK AND XGBOOST ALGORITHMS IN TIME SERIES FORECASTING

Łukasz SROKA

University of Economics in Katowice; lukasz.sroka@edu.uekat.pl, ORCID: 0000-0001-5721-2475

Purpose: The aim of the article was to prepare a simulation analysis of artificial neural network and XGBoost algorithm with determining which of the method was characterized by a lower level of forecast errors for time series predictions.
Design/methodology/approach: The objective of the article was reached by applying, a simulation study on a sample of 1000 artificially generated time series. The analyzed XGBoost algorithm and the artificial neural network ANN model were intended to prepare forecasts for five periods ahead. These forecasts were compared with the actual implementations of the time series and proposed forecast error measures.
Findings: It is possible to use simulated time series to check which of the presented algorithms were characterized by a lower forecast error. The study showed that applying of the artificial neural networks ANN to forecast future observations generated a lower level of MAPE, MAE and RMSE errors than in the case of the XGBoost algorithm. It was found that both methods generate a lower level of forecast error for time series characterized by a high level of mean value, standard deviation and variance, and levels of kurtosis and skewness close to 0.
Practical implications: The research results can be used by both investors and enterprises to better adjust their business decisions to changing market prices by using a model with a lower forecast bias.
Originality/value: The original contribution of this article is a comprehensive comparison of forecasts generated by the XGBoost and ANN algorithm, along with determining for which types of time series of the algorithms forecast future values with less error. Moreover, due to the use of simulated artificial time series, it was possible to test each algorithm for various market conditions.
Keywords: Artificial Neural Network, XGBoost, time series, forecasting, simulation.
Category of the paper: Research paper.

1. Introduction

Price forecasting of financial instruments is one of the main challenges facing investors and financial institutions. Properly determining the behavior of a given financial asset in the future can significantly increase the effectiveness of investment portfolio management and reduce the risk associated with unsuccessful investments. The literature contains many econometric, machine learning algorithms and artificial neural networks models to forecast the future values, however, selecting the appropriate method for modeling and forecasting the market still raises many discussions.

Price forecasting is almost impossible. Many models were created to help investors achieve positive rates of return on their investments, but such models are usually very unreliable. Investors or traders who have just entered the market often do not understand the complexity of the relationships existing on the financial world, and only experienced investors are able to effectively use certain models to make business decisions using data coming from the market (Biswas et al., 2019). Current research proves that deep learning with artificial neural network was the most commonly used model for forecasting stock price trends (Mintarya et al., 2023; Ozbayoglu et al., 2020). Discussions on the applying of econometric methods to forecast prices of financial instruments lead to the conclusion that these methods, due to their characteristics and the level of market volatility, may not be suitable for market forecasting. However, some researchers state that econometric models may provide better results than other methods in the case of some non-linear time series (Li et al., 2022). For forecasting more complex non-linear financial time series, algorithms such as: support vector machine (SVR), eXtreme Gradient Boosting (XGBoost), or Multilayer Perceptron (MLP) are becoming increasingly popular (Oukhouya, El Himdi, 2023). These methods are able to detect complex non-linear dependencies in forecasting prices of financial instruments and achieve a better level of fit to real data by tuning many hyperparameters (Kim, Kim, 2019). Hyperparameter optimization or tuning in machine learning algorithms refers to selecting the most appropriate parameters for a particular learning model (Al-Thanoon et al., 2019).

Research conducted by Ariyo et al. (2014) showed that the XGBoost algorithm can be an effective tool for predicting future prices on the American stock exchange, where researchers achieved 87% effectiveness using the XGBoost algorithm to predict stock prices. Cai-Xia et al. (2021) presented that the multistep XGBoost prediction model presented a much better prediction accuracy and model stability than the multistep ARIMA model. The XGBoost model performed better in predicting complicated and nonlinear data like HFRS. Additionally, multistep models are more practical than one‑step prediction models in forecasting the future values. Researchers Chung and Shin (2020) indicated that artificial intelligence models such as the ANN use predictors to a better extent to determine accurate forecasts for subsequent periods, regardless of whether the relationships are linear or non-linear, without the need to check the basic statistical assumptions. Gao et al. (2017) found that the ANN model with 20 neutrons delayed by 4 periods produces forecasts with a greater RMSE error than the ARIMA model. The researcher also found that as the forecast horizon increases, both the ANN model and ARIMA model predictions become more subject to the RMSE error. Yamin et al. (2004) performed a simulation analysis of the artificial neural network ANN based on a electricity prices. The model consisted of price simulations along with forecasts for subsequent periods.

The simulation study confirmed that the ANN model perform well with price outliers occurring in the data, and the use of an appropriate model preparation and testing strategy together with appropriate features selected for price forecasting significantly improves performance of the algorithm.

Some research focuses not only on the usage of the artificial neural network algorithms or the machine learning methods, but also on the appropriate preparation of the data for the model. The proper selection of the features included in the model can significantly improve performance and the quality of the model forecasts. Researchers Chen and Hao (2017), Wang et al. (2018), Naik and Mohan (2019) created technical indicators and applied them along with the historical financial instruments to improve the quality and the efficiency of the algorithms. As the number of features included in the algorithm was expanded, some researchers used extraction or appropriate selection of data included in the model. to prevent the curse of dimensionality along with the generation of technical indicators. Many studies have applied an expanded input feature space with technical indicators. However, the scope and number of the technical indicators are not yet established, but it is worth remembering that a large set of data with many characteristics causes the problem of irrelevant and redundant information, which can significantly deteriorate the performance of learning algorithms (Yun, 2021).

Since it was noticed that there was no comprehensive analysis and comparison of forecasts created using the XGBoost algorithm and ANN artificial neural networks in terms of their errors, the aim of this article is to prepare a simulation analysis of the tested models with determining which of the method is characterized by a lower level of the forecast errors. In this article, a simulation study was carried out on a sample of 1000 artificially generated time series. The tested XGBoost algorithm and the artificial neural network ANN were intended to prepare forecasts for five periods ahead. These forecasts were then compared with the actual implementations of the time series and calculated error measures.

The original contribution of this article is a comprehensive comparison of forecasts generated by the XGBoost and ANN, along with determining for which types of time series of the algorithms forecast future values with less forecast error. Moreover, due to the use of simulated artificial time series, it was possible to test each algorithm for various market conditions.

2. XGBoost

XGBoost (Extreme Gradient Boosting), is a gradient boosting algorithm. This technique is currently one of the most popular methods in the field of data mining. The XGBoost includes a set of classifiers, which can be decision tree models. In this method, as in the case of random forests, the final decision regarding the results is influenced by all trees used to build the algorithm. The XGBoost applies an incremental strategy which is less complicated and time-consuming than training all trees at once. This method introduces a regularization component into its calculations. The general form of the XGBoost consists of two parts. The first part is the component responsible for minimizing the error, called the loss function, or cost function. The second one, called *regularizer*, helps prevent overfitting and controls the complexity of the model (Grabowska, 2019). To compensate for errors resulting from possible missing data, the algorithm tries to determine the default missing direction by calculating whether it is better for all missing values to select the left subset in the current node or to the right set of subsets of the decision tree (Zhang, 2022) Importantly, the XGBoost is an ensemble model that relies on the efficient implementation of decision trees to create a combined model whose predictive performance is better than individual techniques used alone (Jabeur et al., 2021).

The operation of the XGBoost algorithm is as follows:

1. The data set is divided into a training set and a test set.
2. The algorithm creates the first decision tree, based on which it makes predictions using training data.
3. The next tree created by the algorithm is corrected with the residuals resulting from the predictions received from the previous tree.
4. Steps 2 and 3 are repeated until the algorithm completes or until satisfactory results are obtained.
5. The quality of the algorithm is checked on the test set. If the results are satisfactory, the algorithm can be used for further prediction. Otherwise, there is a need to change the set of hyperparameters.

The output function of the algorithm is calculated as follows (Mo et al., 2019):

$$\hat{Y}_i^T = \sum_{k=1}^{T} f_k(x_i) = \hat{Y}_i^{T-1} + f_T(x_i) \tag{1}$$

where:
$\hat{Y}_i^{T-1}$ is a created tree,
$f_T(x_i)$ is a new tree model,
T is the number of integer trees used in the algorithm.

3. Artificial neural network (ANN)

The artificial neural networks algorithm ANN was proposed in response to the need to create the artificial neural network algorithm which was able to effectively deal with nonlinear problems. The artificial neural networks ANN represent a non-linear approach that does not require prior strict assumption of the form of data or the type of linearity of the analyzed sample. Regardless of its inherent nonlinear nature, ANN can also effectively deal with linear problems. ANNs have accurate approximation capabilities that can fit a wide set of problems (Martinović et al., 2020). The ANN is a flexible and well-performing model that can be easily implemented for various data patterns. However, overdesign (e.g., large number of hidden nodes and repeated connections) may impose some drawbacks. Overfitting is common in complex systems that involve too many parameters. Therefore, sample data variance is built into the model design, which is not preferred.

The ANNs consist of many individual artificial neurons. The operation of the artificial neuron is basically as follows: each artificial neuron processes a certain finite number of inputs $x_i, i = 1, 2, \ldots, n$ into one output y. The input signals come either from outside the network or from the outputs of other nerve cells that create a given neural network. Network synapses process input signals through appropriate weight coefficients, which are determined during the learning process of the neural network. Input signals are introduced to the artificial neuron through connections with given weight coefficients $w_i, i = 1, 2, \ldots, n$, which both reflect the strength of the signals and constitute the memory of the neurons because they are able to remember the relationship between these signals and neuron output signals. Weights in the artificial neural networks can have both positive and negative values, and the sign of the weight in a given synapse may change during the learning process (Sarapata, 2011).

Based on the data entered at the input of the network, the total excitation of neuron e is calculated, most often as a linear combination of inputs, often supplemented with a free expression, which can be written as (Witkowska, 2002):

$$e = w_0 \sum_{i=1}^{n} w_i x_i = w_0 + w^T x \tag{2}$$

where:
x is a vector [n x 1] of input signals,
w is a vector [n x 1] of weights that express the degree of importance of information.

In the summing block, the weighted sum of inputs is determined (calculated as a linear combination of the vector of input signals *x* and the vectors of their corresponding weight coefficients *w*). Then, in the activation block, the signal representing the total activation of neuron *e* is transformed by a specific activation function of neuron *f*. The value determined by this function is the output value y of the neuron, which can be written as (Sarapata, 2011):

$$y = f(w^T x + w_0) = f(e) \tag{3}$$

The main task of the activation function is to determine the output, based on the input data and weights of a given neuron. An important condition in the use of the artificial neural networks is the requirement to apply differentiable activation functions. Many different proposals for activation functions have been considered in the literature, but the most popular ones include: a linear function (linear neuron), a sigmoid function (sigmoid neuron), a tangent function and a Gaussian function (radial neuron) (Tadeusiewicz, Szaleniec, 2015).

The artificial neural networks consist of many layers of individual neurons, in such a way that the outputs of the neurons of the previous layer create a vector fed to the input of each neuron of the next layer. The neurons of each layer always have the same number of inputs equal to the number of neurons in the previous layer + 1, while within one layer the neurons have no connections between each other (Balkin, Ord, 2000). In the ANN model, the first layer is the input layer, which can capture the signal and transmit it to subsequent layers of the network (called hidden layers). Typically, neural network models have at least one hidden layer. The last layer of the network is the output layer, in which output signals are formed in neurons.

Training of the ANN algorithm is most often performed applying the backpropagation method. The operation of the backpropagation algorithm is as follows: after preprocessing the response of the network to a given pattern, the gradient value of the error function for the neurons of the last layer is calculated. Then their weights are modified. The error is propagated to the previous (penultimate) layer. The values of the gradient function for neurons from this layer are generated based on the gradients for neurons from the next (i.e. last) layer. The weights of the next layer are modified. The procedure continues until the input layer. Therefore, a formula describing the components of the δ vector is obtained (Osowski, 2006):

$$\delta_{(i,n)} = (\textstyle\sum_k \delta_{(i+1,k)}\, w_{(k,n)})\mathrm{y}_n(1 - \mathrm{y}_n) \tag{4}$$

where:

the index *i* means the number of the layer (*i* - current, *i+1* - next),

k the number of the neuron in the next layer,

n the number of the currently considered neuron (*n-th* component of the input vector for the next layer).

4. The simulation analysis

The simulation study involved generating 1000 artificial time series. Each of the generated time series contained 2352 observations imitating daily closing prices of a financial instrument. The time series was generated applying a library available in Python called *mockseries*. This library allows to prepare synthetic time series with characteristics provided by the researcher. In addition to the *mockseries* library, the *random* library was used to calculated random numbers and the *statsmodels* library was used to check the stationarity of the artificially generated time series with the ADF test. In the table1 the possible parameters that were randomly selected for each of the artificially generated time series are presented.

Table 1.
Potential parameters for artificial time series

Paramiter	Possible values to draw
Seasonality	Yes/No
Data noise	Yes/No
In case of data noise – a level of standard deviation	From 1 to 9
In case of noisy data – a level of mean	From 0 to 2
Slopes of the time series	Up/down
Trend coefficient	From 1 to 9
Amplitude coefficient	From 1 to 9
Trend change (over periods)	From 50 to 350
Seasonality (in periods)	From 50 to 350

Source: own studies.

The randomization of the parameters presented in the table 1. was done using the random number generator and the Do -While loop implemented in Python. The purpose of the script was to generate 1000 non-stationary time series. Whether a given series was classified as a non-stationary series was the result of the ADF test performed with the adopted significance level of 0.05. When the ADF test indicated that the series was stationary, the script skipped a given series and generated the next one until the assumed number of non-stationary time series was created. The study focused on non-stationary time series to be able to reflect the real financial data as closely as possible. Due to the possibility of negative prices of futures contracts for financial instrument in practice, the script had no restriction on accepting negative values. This approach allows to test the analyzed algorithms in various market conditions. Adopting random values for the parameters presented in the table 1. let to generate time series with different levels of mean value, variance, seasonality, or noise in the data. This approach allows to holistically check how different time series structure influence on algorithm accuracy. Figure 1 shows an example of six randomly generated time series.

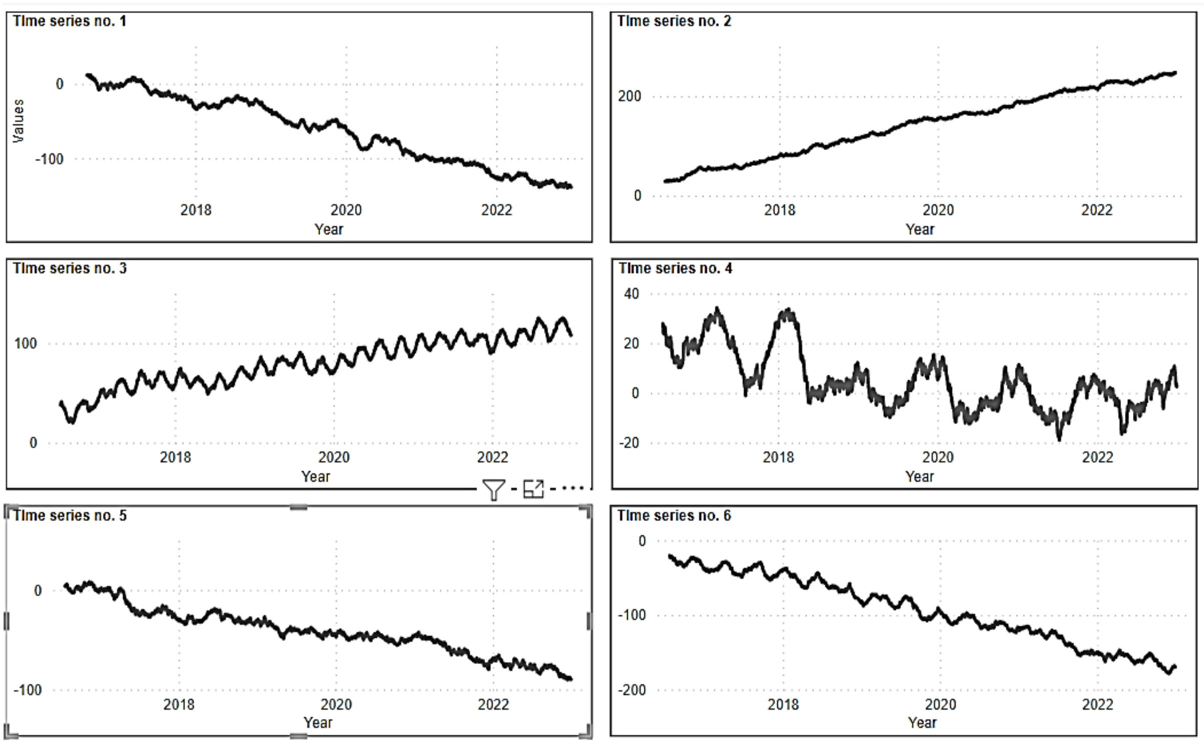

Figure 1. Six examples of randomly generated time series.

Source: own study.

5. Methodology of the simulation analysis

The data for each of the simulated time series were divided into two subsets: the training set and the prediction set. Each of the analyzed methods was intended to generate forecasts for five consecutive periods, then the obtained forecasts were compared with the actual values from the forecast set and forecast errors such as: average absolute percentage error, average absolute error and root mean square error were calculated. The last element of the study was to compare the obtained results of the tested models and check for which time series, statistically speaking, a given method forecast values with a smaller and for which ones with a larger forecast error.

In the case of the tested algorithms, the variables on the basis of which the model learned to predict prices for subsequent periods were five-, twenty-five-, fifty-, one-hundred- and two-hundred-fifty-period moving averages lagged by five periods. Moving averages were used to take into account time relationships in machine learning algorithms and the artificial neural networks. Lagging variables means that the algorithms examine the relationship that moving averages from period *t-5* have on the forecast variable *y* in period *t*. The explanatory variables were lagged in order to be able to easily implement the data into the model in order to prepare a forecast for the next five periods. The testing procedure was as follows: the ANN and the XGBoost models were fitted to the full set of training data. The trained algorithm was then used to make predictions using the forecast set. This approach significantly improved the simulation performance. The risk of overtraining the algorithms was reduced by each time searching the grid of defined parameters separately for each time series and by determining the first, second and third quartiles of the MAE, RMSE and MAPE errors.

In the simulation analysis, it was assumed that acceptable forecasts are considered to be forecasts for which the average absolute percentage error has values less or equal to 5%, acceptable forecasts are considered to be forecasts with an error between 5% and 10%, and unacceptable forecasts are considered to be forecasts for which whose average absolute forecast error was more than 10%.

In both the case of the XGBoost algorithm and the artificial neural network ANN, the *GridSearchCV* function from the *sklearn.model_selection* library was used to select the optimal hyperparameters of the algorithm. The *GridSearchCV* method allows you to automatically search a defined grid of hyperparameters to determine the optimal set of parameters for a given model. The study defined a variable selection strategy using triple cross-validation of data.

In the case of the XGBoost algorithm, the parameter grid searched by the *GridSearchCV* functions contained the following parameters: the number of boosting stages performed by the model (this parameter allows to limit the over-fitting of the model to the data), the alpha coefficient (affects the level of L1 regularization) and the eta coefficient (a parameter preventing over-fitting by weight correction). The searched values in the parameter grid of the XGBoost algorithm are presented in table 2.

Table 2.
Parameters of the XGBoost algorithm used in simulation studies

Parameter	Possible values to use
number of boosting stages	25, 50, 100, 250, 500
eta	0.01, 0.025, 0.05, 0.075, 0.1, 0.25 ,0.5, 0.75, 1
alfa	1, 5, 10, 25, 75

Source: own studies.

In the case of the artificial neural network algorithm ANN, *keras* libraries were chosen to create the model. Using the *keras* library and the function contained in it, it was possible to prepare a neural network consisting of one input layer, one dimension reduction layer, three hidden layers and one output layer. 32 neurons were used in the input layer, 16 in the first hidden layer, 8 in the second hidden layer, 4 in the third hidden layer and 1 neuron in the output layer. The optimization function was set to "Adam", the number of test samples propagated in the network was set to 512, and the number of network iterations (epochs) was set to 200. A sequential model was applied in the study. Model loss functions set to "mape". Before training the neural network using the *StandardScaler* function available in the *sklearn* library, the data was scaled. Moreover, the *GridSearchCV* function was used to search for the optimal activation, initialization and so-called learning rate of the optimizing function.

In the table 3 the parameter grid searched by the *GridSearchCV* function for the ANN model was presented.

Table 3.
Parameters of the ANN algorithm used in simulation studies

Parameter	Possible values to use
Activation function	linear, relu, tahn, sigmoid
Initialization function	normal, uniform
Learning rate	0.0001, 0.001, 0.01

Source: own studies.

6. Results and discussion

The simulation studies carried out allowed for the preparation of forecasts for 5 subsequent periods for 1000 time series, which gave a total of 5.000 forecasts. For each simulated time series forecasts, the MAE, MAPE and RMSE errors were calculated. Table 4 presents quartile I, median and quartile III of the analyzed errors of the XGBoost algorithm.

Table 4.
XGBoost model prediction errors for the performed simulations

Prediction error	Quartile I	Median	Quartile III
MAPE	0.732%	1.547%	3.390%
MAE	0.647	1.176	2.126
RMSE	0.754	1.333	2.301

Source: own studies.

As presented in the table 4, the median level of the average absolute percentage error of the XGBoost algorithm was 1.547%. This means that 50% of the MAPE errors are lower than or equal to 1.547% and the remaining 50% of the MAPE errors are equal to or higher than 1.547%. The level of MAPE quartile I was 0.723%, while quartile III was 3.390%. This means that 25% of the MAPE errors are less than or equal to 0.723%, while 75% of the MAPE errors are higher than this value. In the case of quartile III, 75% of forecasts have an error lower than or equal to 3.390, while 25% of forecasts have a MAPE error higher than this value. The median of the mean absolute error of the MAE was 1.176. This means that for 50% of forecasts the forecast value deviates from the actual implementation of the time series by 1,176 units or less, while for the remaining 50% by more than 1,176 units. The value for quartile I of the MAE error was 0.647 units, while for quartile III it was 2.126 units. The median mean square error of the RMSE forecast was 1,333 units. This means that the midpoint value of the average expected forecast varies by plus or minus 1,333 units between the predicted value and the actual value. The value for quartile I was 0.754 units, while for quartile III it was 2.301 units. The figure 2 and figure 3 show examples of a time series for which the MAPE prediction error was less than 5% and one example of a time series for which the error was higher than 10%.

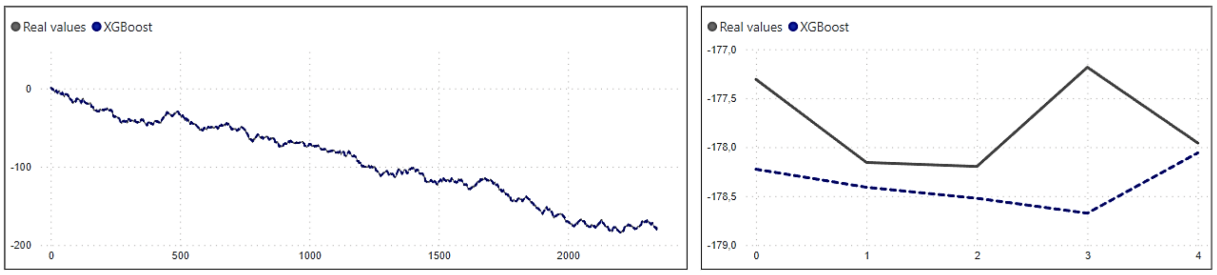

Figure 2. An example of a time series with forecast error <5% (series no. 94).

Source: own study.

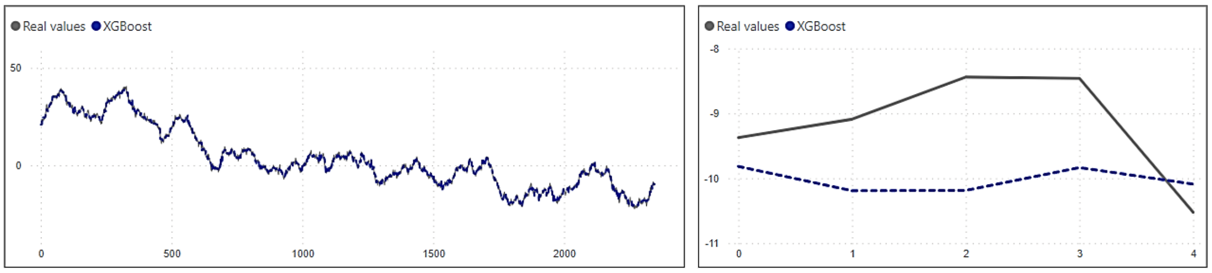

Figure 3. An example of a time series with forecast error <5% (series no. 94).

Source: own study.

The next analyzed algorithm was the artificial neural networks ANN. The results of the prediction errors of the simulation analysis of the ANN are presented in the table 5.

Table 5.
The ANN model prediction errors for the performed simulations

Prediction error	Quartile I	Median	Quartile III
MAPE	0.668%	1.490%	3.411%
MAE	0.604	1.145	2.045
RMSE	0.687	1.304	2.178

Source: own studies.

As presented in the exhibit 8 the median level of the average absolute percentage error of the ANN algorithm was 1.490%. This means that 50% of the MAPE errors are lower than or equal to 1.490% and the remaining 50% of the MAPE errors are equal to or higher than the value. The level of MAPE quartile I was 0.668%, while quartile III was 3.411%. The median of the mean absolute error MAE was 1.145. This means that for 50% of forecasts the forecast value deviates from the actual implementation of the time series by 1.145 units or less, while for the remaining 50% by more than 1.145 units. The value for quartile I of the MAE error was 0.604 units, while for quartile III it was 2.045 units. The median mean square error of the RMSE forecast was 1,304 units. This means that the midpoint value of the average expected forecast varies by plus or minus 1,304 units between the predicted value and the actual value. The value for quartile I was 0.687 units, while for quartile III it was 2.178 units. The figure4 and the figure 5 present one example of a time where the level of the MAPE error for the ANN algorithm was less than 5% and one example of a time series for which the error was higher than 10%.

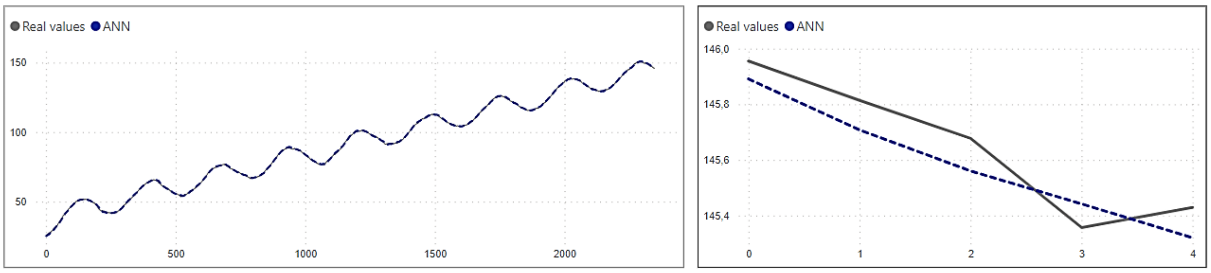

Figure 4. An example of a time series with forecast error <5% (series no. 94).

Source: own study.

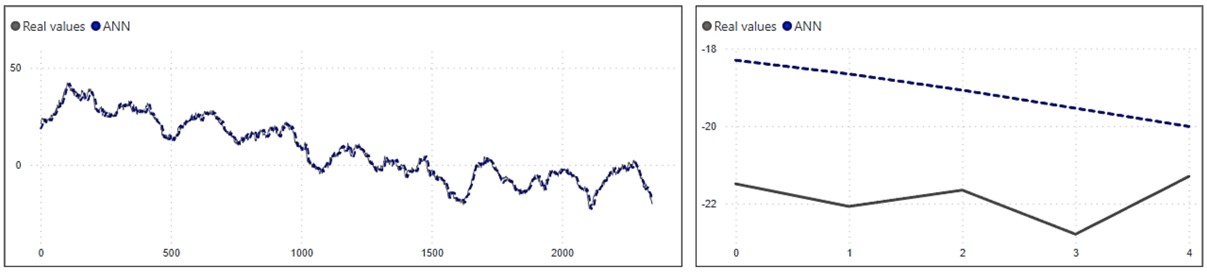

Figure 5. An example of a time series with forecast error <5% (series no. 94).

Source: own study.

Comparing the forecast errors of the XGBoost and the artificial neural networks ANN algorithms, it can be noticed that the medians of all analyzed forecast errors are lower in the case of the ANN model. The values of quartiles I and II are also at lower levels for the artificial neural networks ANN. This means that the forecasts made by the ANN model on a sample of 1,000 simulated time series are characterized by a lower level of error than the forecasts made by the XGBoost algorithm.

The next stage of the analysis was to statistically characterize the time series for which the MAPE error was lower than 5% and the time series for which this error was higher than 10%.

In the case of the ANN artificial neural network algorithm, the number of time series for which the MAPE error was lower than 5% was 839 series, the number of series with the MAPE error was than 10% was 72 series. For the XGBoost algorithm, 837 time series had a MAPE error of less than 5%, while for 75 time series this error was higher than 10%. The table 6 presents the statistical analysis of time series for which the MAPE error was lower than 5%, while the table 7 presents the characteristics of time series with MAPE error higher than 10%.

Table 6.
Characteristics of time series with MAPE forecast error <5%

Quartile	**Skewness**	**Kurtosis**	**Standard deviation**	**Variance**	**Mean**
Quartile I	-0,085	-1,199	16,562	274,313	-34,840
Median	-0,013	-1,124	23,523	553,314	18,177
Quartile III	0,056	-0,965	36,950	1 365,282	62,528

Source: own studies.

Table 7.
Characteristics of time series with MAPE forecast error >10%

Quartile	Skewness	Kurtosis	Standard deviation	Variance	Mean
Quartile I	-0,130	-1,052	8,833	78,017	0,958
Median	-0,021	-0,803	10,774	116,084	6,465
Quartile III	0,188	-0,479	13,681	187,176	11,194

Source: own studies.

Analyzing the above tables, it is worth noting that the XGBoost and ANN algorithm generate lower MAPE forecast errors for time series characterized by a higher level of median variance, standard deviation and mean value than in the case of time series for which the forecast error was higher. It is also worth paying attention to the fact that the level of skewness in models with MAPE error below 5% is close to 0 in practically every quartile, while for series with MAPE error greater than 10%, a weak left-sided asymmetry of the distribution determined by quartile I and a weak right-sided asymmetry can be observed. determined by quartile III. The median kurtosis level is lower for time series with the MAPE error greater than 10%.

7. Conclusion

The aim of this article was to prepare a simulation analysis of the tested models with determining which of the method is characterized by a lower level of forecast errors. The objective was achieved by preparing 1,000 artificial time series, which were the basis for preparing forecasts by the XGBoost and the artificial neural networks ANN. For each time series, future values were forecast for five subsequent periods, then the forecasts were compared with the actual implementations of the time series and the forecast errors of MAE, MAPE and RMSE were calculated.

Simulation studies have shown that the artificial neural network model predicts future time series values with a lower error than the XGBoost algorithm. The median MAPE error of the artificial neural networks ANN was 1,490 and was 3,685% lower than for the XGBoost algorithm. Also, the median MAE and RSME error of the ANN were lower than those of the XGBoost (1.145 vs 1.176 and 1.304 vs 1.333, respectively).

Additionally, statistical characteristics of time series for which the MAPE error was below 5% and time series with a MAPE error above 10% were performed. The analysis showed that time series with a lower level of MAPE error are characterized by a higher level of the median value of the mean standard deviation and variance compared to time series for which this error was higher. The level of median skewness was at a similar level in both groups of time series, while the level of median kurtosis was slightly higher for series with MAPE forecast error above 10%.

Another possibility of conducting the research is to compare the XGBoost algorithm and the artificial neural networks ANN with other models from the group of machine learning algorithms, artificial neural networks or econometric models in order to determine which of the selected models forecasts future values with a lower forecast error.

References

1. Adebiyi, A., Adewumi, A., Ayo, C. (2014). *Stock price prediction using the ARIMA model*. Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, pp. 106-112.
2. Al-Thanoon, N.A., Algamal, Z.Y., Qasim, O.S. (2022). Hyper Parameters Optimization of Support Vector Regression Based on a Chaotic Pigeon-Inspired Optimization Algorithm. *Mathematical Statistician and Engineering Applications, Vol. 71, No. 4,* pp. 4997-5008.
3. Ben Jabeur, S., Mefteh-Wali, S., Jean-Laurent, V. (2021). Forecasting gold price with the XGBoost algorithm and SHAP interaction values. *Annals of Operations Research*.
4. Biswas, M., Shome, A., Islam, A., Ahmed, S. (2021). *Predicting Stock Market Price: A Logical Strategy using Deep Learning.* 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 218-223.
5. Cai-Xia, L., Shu-Yi, A., Bao-Jun, Q., Wei, W. (2021). Time series analysis of hemorrhagic fever with renal syndrome in mainland China by using an XGBoost forecasting model. *BMC Infect. Dis., Vol. 21, No. 1,* p. 839.
6. Chen, Y., Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications, No. 80,* pp. 340-355.
7. Chung, H., Shin, K. (2020). Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications, Vol. 32, No. 12,* pp. 7897-7914.
8. Gao, G., Kwoklun, L., Fulin, F. (2017). Comparison of ARIMA and ANN Models Used in Electricity Price Forecasting for Power Market. *Energy and Power Engineering, No. 9,* pp. 120-126.
9. Grabowska, E. (2019). *Jak udoskonalić algorytm drzew decyzyjnych?* Predictive Solution, from: https://predictivesolutions.pl/jak-udoskonalic-algorytm-drzew-decyzyjnych
10. Kim, T., Kim Ha, Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PLoS ONE 2019, Vol. 14, No. 2.*
11. Kyung, K., Yun, S., Won, Y., Daehan, W. (2021). Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. *Expert Systems with Applications, No. 186.*

12. Latrisha, M., Jeta, H., Callista, A., Said, A., Aditya, K. (2023). Machine learning approaches in stock market prediction: A systematic literature review. *Procedia Computer Science, No. 216*, pp. 96-102.
13. Li, R., Han, T., Song, X. (2022). Stock price index forecasting using a multiscale modelling strategy based on frequency components analysis and intelligent optimization. *Applied Soft Computing, Vol. 124, No. 2.*
14. Martinović, M., Anica, H., Ioan, T. (2020). Time Series Forecasting of the Austrian Traded Index (ATX) Using Artificial Neural Network Model. *Tehnički vjesnik, Vol. 6, No. 27*, pp. 2053-2061.
15. Mo, H., Sun, H., Liu, J., Wei, S. (2019). Developing window behavior models for residential buildings using XGBoost algorithm. *Energy and Buildings, No. 205.*
16. Nagaraj, N., Mohan, B.R. (2019). *Stock Price Movements Classification Using Machine and Deep Learning Techniques-The Case Study of Indian Stock Market*. International Conference on Engineering Applications of Neural Networks. Springer International Publishing, pp. 445-452.
17. Ord, K., Balkin, S. (2000). Automatic neural network modeling for univariate time series. *International Journal of Forecasting, No. 16*, pp. 509-515.
18. Osowski, S. (2006). *Sieci neuronowe do przetwarzania informacji*. Warszawa: OW PW.
19. Oukhouya, H., Khalid El Himd (2023). Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP— For Forecasting the Moroccan Stock Market. *Computer Sciences & Mathematics Forum, 7, No. 1.*
20. Ozbayoglu, A., Murat, G., Mehmet, U.S., Omer, B. (2020). Deep learning for financial applications: A survey. *Appl. Soft Comput., No. 93.*
21. Sarapata, M. (2014). Prognozowanie finansowych szeregów czasowych z wykorzystaniem modeli jednokierunkowych sieci neuronowych. In: W. Szkutnik (ed.), *Studia Ekonomiczne* (pp. 114-126). Katowice: Wydawnictwo Uniwersytetu Ekonomicznego.
22. Tadeusiewicz, R., Szaleniec, M. (2015). *Leksykon sieci neuronowych*. Wrocław: Projekt Nauka. Fundacja Na Rzecz Promocji Nauki Polskiej.
23. Wang, Q., Xu, W., Zheng, H. (2018). Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing, No. 299*, pp. 51-61.
24. Witkowska, D. (2002). *Sztuczne sieci neuronowe i metody statystyczne: wybrane zagadnienia finansowe*. Warszawa: C.H.Beck.
25. Yamin, H.Y., Shahidehpour, S.M., Zuyi, L. (2004). Adaptive short-term electricity price forecasting using artificial neural networks in the restructured power markets. *International journal of electrical power & energy systems, Vol. 26, No. 8*, pp. 571-581.
26. Zhang, Y. (2022). *Stock Price Prediction Method Based on XGboost Algorithm*. International Conference on Bigdata Blockchain and Economy Management (ICBBEM 2022).