The Use of LSTM Neural Networks to Implement the NARX Model. A Case Study of EUR-USD Exchange Rates

Cătălina-Lucia COCIANU, Mihai-Șerban AVRAMESCU
The Bucharest University of Economic Studies, Romania
catalina.cocianu@ie.ase.ro, mihai_serban.avramescu@yahoo.com

The paper focuses on financial data forecasting in terms of one-step-ahead nonlinear model with exogenous inputs. The main aim is the development of a methodology to forecast the exchange rate between EURO and US Dollar. The prediction task is carried out by two recurrent neural networks, the standard NARX neural network and a LSTM-based approach. The exogenous inputs consist of historical trading data and three widely used technical indicators, namely a variant of moving average, the Upper Bollinger Frequency Band and the Lower Bollinger Frequency Band. In order to obtain accurate forecasting algorithms, the exogenous inputs are filtered using the well-known Gaussian low-pass filter. The quality of each method is evaluated in terms of both quantitative and qualitative metrics, namely the root mean squared error, the mean absolute percentage error, and the prediction of change in direction. Extensive experiments point out that the most suited forecasting method is based on the proposed LSTM neural network for NARX model.

Keywords*: Exchange Rate Forecasting, NARX Model, NARX Networks, LSTM Networks, statistical metrics*
DOI: 10.24818/issn14531305/24.1.2020.01

1 Introduction

Modern financial markets are chaotic and very hard to predict systems mainly due to their volatile features and non-linearity. Therefore, stock market prediction is regarded as one of the most challenging applications among time series forecasting and it is under continuous investigations for years. A variety of statistical methods have been introduced to the time series domain and then applied to the problem of financial data analysis and forecasting, such as the autoregressive (AR) family models, exponential smoothing, and moving average (MA) techniques. However, conventional statistical models mostly failed to capture the complexity and non-stationary structure of financial data. During the past decades, machine learning (ML) models turned into serious competitors to classical statistical models in the time series analysis and forecasting field. The most successful ones, such as artificial neural networks (ANN) and support vector machines (SVM) have been widely used to predict financial time series due to their ability to learn from nonlinear data and successfully perform classification and prediction tasks. Several high forecasting accuracy machine-learning techniques have been reported in literature ([1], [2], [3], [4]).

Nowadays, the field of financial forecasting is developing fast, the financial data being the main valuable source of information for the investors and traders. Most of the modern studies on financial data forecasting are focused on developing hybridized models which can combine the advantages of statistical, data mining and machine learning algorithms (see de [5], [6], [7]). On one hand, the aim of using combined models is to process the financial datasets with data mining techniques before considering them for learning process in order to remove noise or to smooth data. Several data pre-processing techniques, as for instance feature extraction and feature selection, could decrease the redundancy and noise in time series data and consequently increase the prediction models performances ([8], [9], [10], [11]). On the other hand, since the exact price of stocks is unpredictable, a series of research works are focused on predicting the price movements, therefore the problem of data forecasting is reduced to a classification problem [12].

In the literature, one of the most recently explored ML field is the deep learning research area (DL). A series of recurrent neural networks, such as convolutional neural networks and long-short time memory (LSTM) have been developed to solve classification and prediction tasks ([13], [14], [15], [16]). The research reported in this paper aims to investigate the potential of the LSTM-based approaches to implement the NARX model.

The outline of the paper is as follows. In Section 2 the NARX prediction model is discussed briefly. Next, the NARX neural network and the LSTM networks are described. The proposed methodology is provided in the fourth section of the paper. In Section 5 the experimental results of the proposed methodology together with a comparative analysis are presented. The concluding remarks are given in the final section of the paper.

2 The NARX prediction model. The NARX Neural Networks

The NARX model (Nonlinear Autoregressive with eXogenous input) is well suited for modelling non-linear time series as well as for one-step and multi-step ahead prediction. To develop the general forecasting model in case of non-stationary time series one can use Partial Autocorrelation Function (PACF) to establish the delay variable.

In the following we denote by:

- T - the total number of time periods
- Y - the variable to be forecast
- XT - the set of N latent variables uses to forecast,
$$\mathrm{XT} = \left(XT(1), XT(2), \ \ldots, XT(N)\right)^T$$
- $d_Y, d_{XT} = \left(d_{XT}(1), \ldots, d_{XT}(N)\right)$ - the corresponding delays.

Note that the delay variable of a time series X corresponds to the lag d, where the PACF function computed for X drops immediately after the d^{th} lag. For $1 \le t \le T$, $\mathrm{Y_t}$ is the value at the moment of time t, and we denote by $\hat{\mathrm{Y}}_{\mathrm{t}}$ the predicted value of $\mathrm{Y_t}$. The general NARX forecasting model is expressed by [17]:

$$\hat{\mathrm{Y}}_{(\mathrm{t+p})} = \mathrm{f}\left(Y_t^{(d_Y)}, XT_t^{(d_{XT})}\right) \qquad (1)$$

where

- f is a non-linear function
- $Y_t^{(d_Y)} = \left\{\mathrm{Y_t}, \mathrm{Y_{t-1}}, \mathrm{Y_{t-2}}, \ldots, \mathrm{Y_{t-d_Y+1}}\right\}$
- $XT_t^{(d_{XT})} = \left\{XT_t^{(d_{XT})}(1), XT_t^{(d_{XT})}(2), \ \ldots, XT_t^{(d_{XT})}(N)\right\}$
- for each $1 \le i \le N$ $XT_t^{(d_{XT})}(i) = \left\{\mathrm{XT_t}(i), \mathrm{XT_{t-1}}(i), \ \ldots, \mathrm{XT_{t-d_{XT}(i)+1}}(i)\right\}$.

In our study we consider p=1, the equation (1) describing the one-step ahead prediction model. In order to simplify (1), we define

$d = \max\{d_Y, \max\{d_{XT}(1), \ldots, d_{XT}(N)\}\}$

and the prediction model

$$\hat{\mathrm{Y}}_{(\mathrm{t+1})} = \mathrm{f}\left(Y_t^{(d)}, XT_t^{(d)}\right) \qquad (2)$$

where

- f is a non-linear function
- $Y_t^{(d)} = \{\mathrm{Y_t}, \mathrm{Y_{t-1}}, \mathrm{Y_{t-2}}, \ldots, \mathrm{Y_{t-d+1}}\}$
- $XT_t^{(d)} = \{\mathrm{XT_t}, \mathrm{XT_{t-1}}, \mathrm{XT_{t-2}}, \ldots, \mathrm{XT_{t-d+1}}\}$
- $\mathrm{XT_t} = \left(\mathrm{XT_t}(1), \mathrm{XT_t}(2), \ldots, XT_{\mathrm{t}}(N)\right)^T$

The non-linear function f can be computed by a neural network, the most commonly choice being the NARX networks.

The NARX neural networks (NARXNN) are recurrent dynamic networks (RNN) specially tailored to model nonlinear systems, as for instance time series. Moreover, NARX networks have high generalization performance and good learning ability which makes them well suited for financial data. The NARXNN topology consists of three layers connected with each other, namely an input layer F_X, a hidden layer F_H, and an output layer F_Y. The general architecture of the NARXNN is displayed in Figure 1.

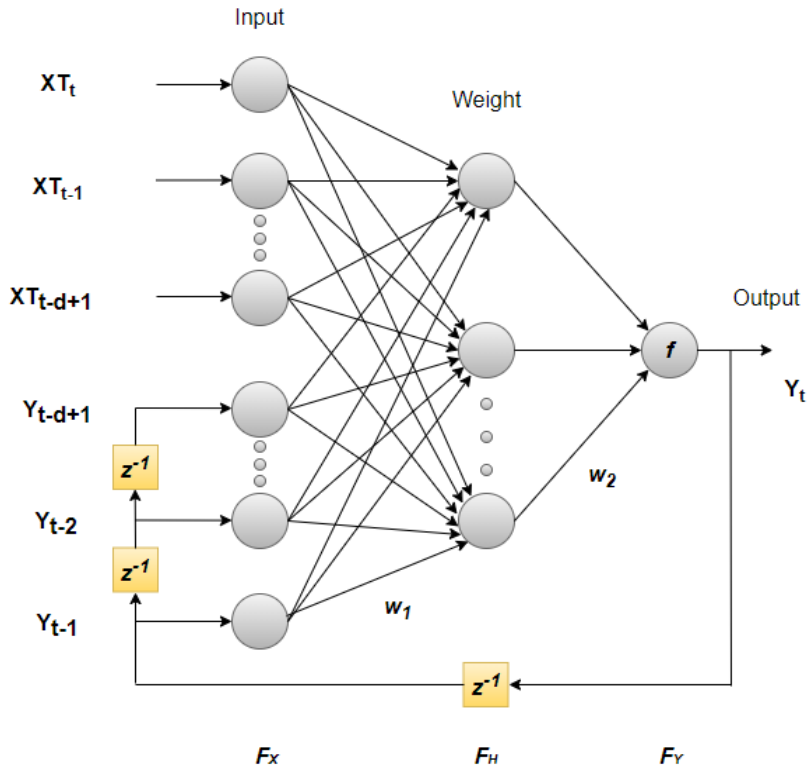

Fig. 1. The architecture of the three-layered NARX network model

The training of NARXNN is of supervised type and usually uses a gradient descent approach. In most of the cases, the local memories of F_H and F_Y are determined by the Levenberg-Marquardt variant of the backpropagation learning algorithm [18]. Since the actual output is available during the training of the network, a series-parallel architecture is created, where the estimated target is replaced by the actual output. After the training step, the series-parallel architecture is converted into a parallel configuration, in order to perform the prediction task. The standard performance function is defined in terms of mean square network errors. Denoting by $|.|$ the number of elements of the argument, the sizes of the hidden layers F_H can be computed many ways, some of the most frequently used expressions being

$$|F_H| = 2\left[\sqrt{(|F_Y| + 2)|F_X|}\right] \quad (3)$$

3 The LSTM Neural Networks

Long Short Term Memory is a type of RNN architecture that allows the long-time learning of time steps dependencies. The general idea of LSTM is to recurrently project the input sequence into a sequence of hidden representations. At each time-step, the LSTM learns the hidden representation by jointly considering the associate input and the previous hidden representation to capture the sequential dependency. [19]

The learning process is accomplished using specific memory blocks located in the recurrent hidden layer. The memory blocks are created from auto-connected cells in which neural network temporal states are saved. Also, each block has an input structure and an output structure. The LSTM has the abilities to remove or add information to the cell state. A memory cell consists of four units: an input gate, a forget gate, an output gate, and a self-recurrent neuron. Each unit cell controls the interactions between neighbouring memory cells and the memory cell itself. The input gate decides if the input modifies the state of the memory cell. The forget gate chooses whether to remember or to forget the previous state of the memory cell. The output gate has the role to decide if the

state of the memory cell should alter the state of other memory cells. [20]

A LSTM neural network maps a certain input sequence $x = (x_1, \dots, x_T)$ to an output sequence $y = (y_1, \dots, y_T)$ iteratively for $t = 1, \dots, T$. The topology of a LSTM network includes one or multiple hidden layers. The mathematics behind the most common model of the LSTM architecture is described as follows [20]. We denote by ° the Hadamard product.

First, one has to define the function used to remove a part from cell state information. The decision belongs to the so-called forget gate layer, usually modelled in terms of the sigmoid function, given by

$$f_t = \sigma\left(W_f \circ [y_{t-1}, x_t] + b_f\right) \quad (4)$$

where W_f is the weight matrix, b_f is the bias vector and f_t is the value of the forget gate at the time *t*.

Next, we have to decide what new information should be preserved in the cell state. The process consists of two steps: the selection of the values that should be updated and the computation of a new vector of candidate values. The updated values are usually computed by the input gate layer of sigmoid type (5) while the vector containing the new candidate values is compute by a tanh layer

$$i_t = \sigma(W_i \circ [y_{t-1}, x_t] + b_i) \quad (5)$$

$$\tilde{c}_t = tanh(W_c \circ [y_{t-1}, x_t] + b_c) \quad (6)$$

where W_i and W_c are the weight matrices, b_i and b_c are the biases, and y_{t-1} is the value of the memory cell at time *t*. Note that i_t is the value of the input gate while $\tilde{c}_t$ represents the candidate state of the memory cell at the time *t*.

Based on the first two steps c_t, an update to the state of the memory cell c_{t-1} is computed as follows

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \quad (7)$$

Finally, the output representing a filtered version of the cell state is given by

$$y_t = o_t \circ tanh(c_t) \quad (8)$$

$$o_t = \sigma(W_o \circ [y_{t-1}, x_t] + b_o) \quad (9)$$

where W_o is the weight matrix, b_0 is the bias vector, and o_t is the value of the output gate at the time *t*.

The inner structure of a LSTM cell containing the external dependencies is shown in Figure 2.

Note that the additional layer representing a memory block can be added to the network in order to prevent a continuous processing of data without a corresponding segmentation.

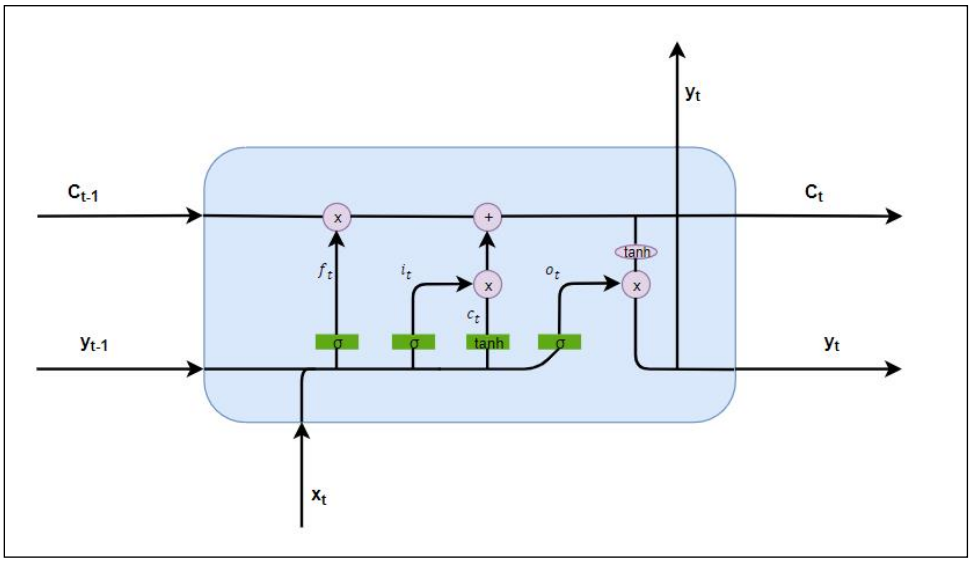

Fig. 2. Standard LSTM cell

The training difficulty occurs when there are temporal dependencies at high distances between the time series values. In this case, small changes in the iterative flow could lead to high effects over future iterations results.
The number of hidden neurons is usually computed based on the number of training samples, the input size and the dimension of output as follows [21]

$$|F_H| = \frac{T}{\alpha \cdot (|F_X| + |F_Y|)} \quad (10)$$

where α is an arbitrary scaling factor, $\alpha \in [2,10]$. Note that smaller values of the parameter α could lead to non-overfitting learning schemes, with significantly large computational effort.

4 The Proposed Methodology

4.1 Technical Analysis and Variable Selection

Technical analysis is used in financial evolutions studies, especially to determine various patterns of fluctuations in order to give the investor the advantage of profitable purchasing and selling decisions. The premise behind technical analysis is that all of the internal and external factors that affect a market at any given point in time are already factored into that market's price [22]. At the same time, the ability to predict future price fluctuations is still a challenge.
In order to establish a suitable methodology, we investigated several technical indicators to obtain more information about the monitored time series and, consequently to obtain more accurate prediction values. From the point of view of NARX model, the considered indicators are exogenous variables representing a subset of the neural network inputs. In the following we briefly describe the motivation and calculus formula for some of the most commonly used technical indicators involved by our study.
A moving average (MA) is a time series constructed by taking averages of several sequential values of another time series. [23] A moving average is an indicator that shows the average value of a time series over a specified period of time. The *n*-period MA for the time series *V* is computed by averaging the values of the past *n* recorded values, as follows

$$MA_n^V(t) = \sum_{i=1}^{n} \frac{V_{t-(i-1)}}{n} \quad (11)$$

Note that all the previous values used to compute the MA indicator (11) are weighted by the same value $p = \frac{1}{n}$. In case *n* is large, one can consider the weights such that higher values are associated to more current data and lower values are considered for the older data. The version of MA used in our study is inspired by the exponential ranking distribution probability and it is defined by:

$$MA_n^V(t) = \sum_{i=1}^{n} \alpha_i \cdot V_{t-(i-1)} \quad (12)$$

$$\alpha_i = \frac{\beta_i}{C}, \qquad i = 1, \dots, n \quad (13)$$

where $\beta_i = \frac{1}{exp\{i\}-1}$ and $C = \sum_{k=1}^{n} \beta_k$.

Another class of technical indicators introduced to measure the degree of volatility for the security's prices consists of the Bollinger Frequency Bands. Bollinger Bands are quantitative technical analysis tools introduced by Bollinger [24]. The use of Bollinger Frequency Bands can lead to forecasting systems able to identify short-term trending properties in a financial time series and exploit them for superior investment returns [25].
The Bollinger Frequency Bands use simple moving average along with lower and higher frequency to measures volatility in price evolution. The frequency bands are usually set to two standard deviations away from the simple moving average. The Upper Bollinger Frequency Band (UPBB) is computed by

$$UPBB_n^V(t) = MA_n^V(t) + 2 \cdot \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(V_{t-i+1} - MA_n^V(t)\right)^2} \quad (14)$$

while the Lower Bollinger Frequency Band (LOBB) is given by

$$LOBB_n^V(t) = MA_n^V(t) - 2 \cdot \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(V_i - MA_n^V(t)\right)^2} \quad (15)$$

4.2 Data Pre-processing

The methodology is developed to analyse historical data representing the indicator of exchange rate between EURO and US Dollar, recorded between October 1999 and October 2019 [26].

The targeted series is EURO/USD Closing Price, which represents the last prices recorded for every bank day. To forecast the targeted values, we use the following exogenous variables:

- EURO/USD LOW, representing the indicator with the minimum value for every bank day;
- EURO/USD HIGH, representing the indicator with the maximum value for every bank day.
- three technical indicators computed based on the targeted time series, namely MA, UPBB and LOBB.

In order to estimate a suitable window of the delay values we computed the PACF value of each time series. Let *d* be such that, for all considered variables, PACF function drops immediately after the d^{th} lag. This means that the delay should be set in a certain neighborhood of *d*.

Next, all the time series recorded values are normalized, that is each variable is of zero mean and unit variance. Finally, each exogenous input is filtered using the standard Gaussian low-pass filter. We evaluated the forecasting capacity of the NARX based model by splitting the available data into train data (70%) and test data (30%), considered as new, unseen yet samples.

4.3 Performance Measures

To establish meaningful conclusions regarding the research work, various performance metrics have been selected to evaluate the prediction capabilities of models. We used two classes of measures to assess the forecasting capacity of the developed methods, namely quantitative indicators and qualitative (trend-based) metrics.

In the following we denote by $\{Y(1), \dots, Y(T)\}$ the set of actual values of Y and we assume that $\{\hat{Y}(1), \dots, \hat{Y}(T)\}$ is the collection of the forecasted values.

The mean absolute percentage error (MAPE) measures the magnitude of the forecasting error in percentage terms. Due to the fact that the predicted values are deviated both positively and negatively in relation to the actual values, the calculation of MAPEI becomes necessary in the analysis process. Naturally, past data should be discounted in a more gradual way [27].

$$MAPE = 100 \times \frac{1}{T}\sum_{i=1}^{T}\frac{\left|Y(i) - \hat{Y}(i)\right|}{Y(i)} \quad (16)$$

The root mean square error (RMSE) evaluates the standard deviation of the differences between forecasted values and actual ones. The advantage of using RMSE (17) is that it penalizes big errors relatively more than small errors [28]

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}\left(Y(i) - \hat{Y}(i)\right)^2} \quad (17)$$

The prediction of change in direction (POCID) is a metric used to measure the percentage of the number of correct decisions related to the changes in direction or trend

$$POCID = 100 \times \frac{\sum_{i=1}^{T-1} D_i}{T} \quad (18)$$

where

$$D_i = \begin{cases} 1, & \left(Y(i+1) - Y(i)\right)\left(\hat{Y}(i+1) - \hat{Y}(i)\right) \geq 0 \\ 0, & otherwise \end{cases}$$

Obviously, good forecasting performances correspond to small values of RMSE and MAPE measures and high POCID percentages.

5 Experimental Results

In order to develop a comparative analysis on the use of NARX neural network and LSTM neural network to implement the NARX forecasting model, a long series of experiments have been conducted. The results are presented in the following.

First, we had to establish the delay parameter in prediction model (2). In order to solve this problem, we computed the Partial Autocorrelation Function (PACF) corresponding to each time series.

Table 1. The PACF analysis

Series	Lag
X	2
X1	3
X2	3
EMA – 2 days	3
EMA – 3 days	2
LOBB	4
UPBB	4

Based on the values shown in table 1, we have tested the proposed methodology with delay parameters belonging to the window $W_d = \{2,3,4,5,6\}$.

Taking into account that the simplest non-linear auto-regressive model (NAR), which does not involve exogenous data, is often the best choice in time series forecasting, we developed a LSTM neural network to implement (19)

$$\widehat{Y}_{t+1} = f1(Y_t, Y_{t-1}, \dots, Y_{t-d+1}) \quad (19)$$

The analysis of the corresponding forecasting accuracy is reported in Table 2.

Table 2. The forecasting results using the model (19)

DELAY	MAPEI	RMSE
1	0.5415 ± 27.8 %	0.0078 ± 22.5 %
2	0.5000 ± 22.8 %	0.0073 ± 19.0 %
3	0.6170 ± 35.8 %	0.0086 ± 29.4 %
4	0.5997 ± 35.1 %	0.0085 ± 29.6 %

Table 3. The forecasting results using NARX NN for NARX model

DELAY	MAPEI	RMSE
2	1.1871 ± 67.7%	0.0157 ± 62.2%
3	0.7687 ± 35.4%	0.0112 ± 31.9%
4	0.8029 ± 36.6%	0.0116 ± 33.3%
5	0.7746 ± 32.2%	0.0111 ± 27.9%
6	0.8815 ± 19.1%	0.0125 ± 18.2%

Table 4. The forecasting results using LSTM NN for NARX model

DELAY	MAPEI	RMSE
2	0.4171 ± 22.73%	0.0061 ± 20.15%
3	0.4150 ± 21.91%	0.0060 ± 19.46%
4	0.4207 ± 19.61%	0.0061 ± 17.07%
5	0.4066 ± 17.87%	0.0060 ± 16.38%
6	0.4196 ± 14.83%	0.0062 ± 14.77%

We implemented the NARX forecasting model (2) using either the standard NARX neural network described in Section 2 or the LSTM neural network presented in Section 3. The results are shown in Table 3 and Table 4. The graphic representation of forecasted values versus the true ones corresponding to the data in Table 2, Table 3 and Table 4 is depicted in Figure 3, Figure 4 and Figure 5, respectively.

The POCID values vary between 55% in case of using NARX NN and 60% when LSTM NN is considered.

Based on the obtained results, we conclude that the LSTM neural network is more suited than standard NARX neural network to forecast future values of the analysed data from both accuracy and stability points of view.

6 Conclusions and Suggestions for Further Work

The reported work focuses on the development of financial data forecasting methods to implement the one-step-ahead nonlinear model with exogenous inputs. In our study the main aim is to obtain a prediction methodology to forecast the exchange rate between EURO and US Dollar. The prediction task is carried out by two recurrent neural networks, the standard NARX neural network and a LSTM-based approach. We analysed five exogenous time series, two of them of stock-based type and the other three belonging to the technical indicators class. In order to obtain accurate results, the exogenous inputs are filtered using the well-known Gaussian low-pass filter.

To establish meaningful conclusions regarding the accuracy of the obtained results, various performance metrics have been settled. Extensive experiments pointed out that the most suited forecasting method is based on the proposed LSTM neural network for NARX model.

We conclude that the results are encouraging and entail future work toward extending this approach to more complex NN-based models as well as hybrid techniques.

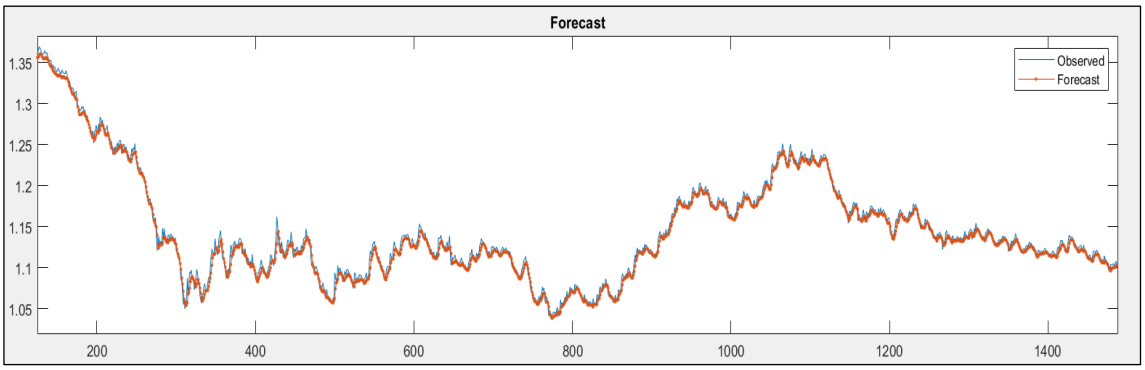

Fig. 3. The forecasting results using the model (19)

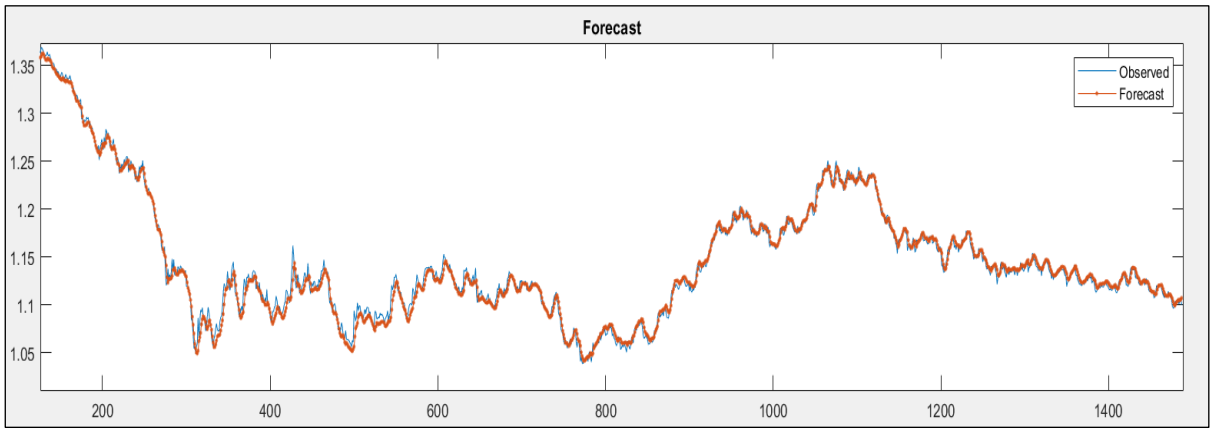

Fig. 4. The forecasting results using NARX NN in model (2)

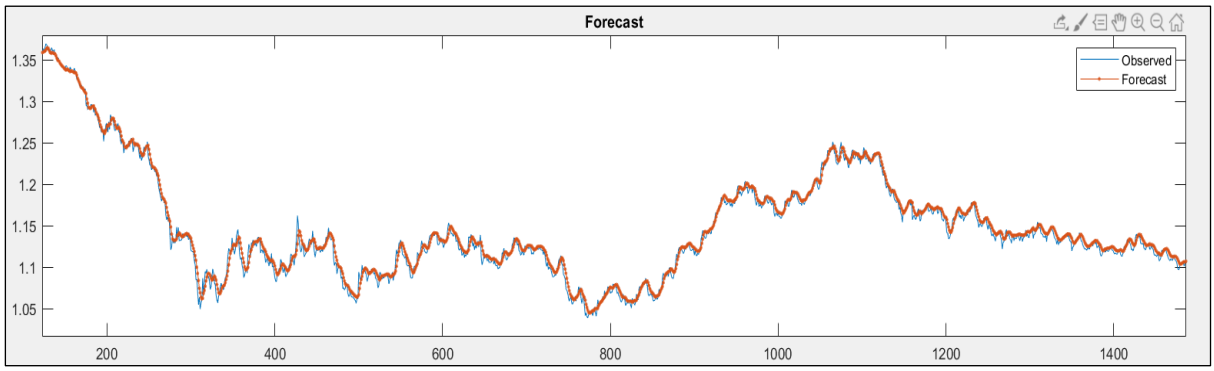

Fig. 5. The forecasting results using LSTM NN in model (2)

References

[1] P. Sujin , L. Jaewook, C. Mincheol, J. Huisu, Predictability of machine learning techniques to forecast the trends of market index prices: Hypothesis testing for the Korean stock market, 2017.

[2] K. M. Okasha, Using Support Vector Machines in Financial Time Series Forecasting, 2014.

[3] P. Pallabi, K. Binita, Stock Market Prediction Using ANN, SVM, ELM: A Review, 2017.

[4] C. Deepika, S.S. Manminder, Stock Direction Forecasting Techniques: An Empirical Study Combining Machine Learning System with Market Indicators in the Indian Context, 2014.

[5] R.D. Smruti, M. Debahuti, R. Minakhi, A hybridized ELM-Jaya forecasting model for currency exchangeprediction, 2017.

[6] K. Mergani, Xu-Ning, T. Nashat, AL-Jallad, Hybrid Forecasting Scheme for Financial Time-Series Data using Neural Network and Statistical Methods, 2017.

[7] J. G. Osório, M. Lotfi, M. Shafie-khah, M.A. Vasco Campos and P. S. Catalão João, Hybrid Forecasting Model for Short-Term Electricity Market Prices with Renewable Integration, 2018.

[8] A. Azadeh, M. Sheikhalishahi, M. Tabesh and A. Negahban, The Effects of Pre-Processing Methods on Forecasting Improvement of Artificial Neural Networks, 2011.

[9] P.Y. Zhou and C.K. Chan, A Feature Extraction Method for Multivariate Time Series Classification Using Temporal Patterns, 2015, pp. 409–421.

[10] G. Jianbo, S. Hussain, J. Hu and W.W. Tung, Denoising Nonlinear Time Series by Adaptive Filtering and Wavelet Shrinkage: A Comparison, 2010.

[11] K. Torsten and D. Lorenz, A comparison of denoising methods for one dimensional time series, 2005.

[12] F. Feng, H. Chen, X. He, J. Ding, M. Sun, T.S. Chua, Improving Stock Movement Prediction with Adversarial Training, 2018.

[13] A. Azzouni and G. Pujolle, A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction, 2017.

[14] Y.L. Kong, Q. Huang, C. Wang, J. Chen, J. Chen and D. He, Long Short-Term Memory Neural Networks for Online Disturbance Detection in Satellite Image Time Series, 2018.

[15] Yi Fei Li, Han Cao, Prediction for Tourism Flow based on LSTM Neural Network, 2017.

[16] K. Greff, R. K. Srivastava, J. Koutnık, B.R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, 2017.

[17] J. Maria P. Junior and G. A. Barreto, Long-Term Time Series Prediction with the NARX Network: An Empirical Evaluation, 2007.

[18] G.A.F. Seber, C.J. Wild, Nonlinear Regression, John Wiley & Sons, 2003.

[19] F. Feng, H. Chen, X. He, J. Ding, M. Sun, T.S. Chua, Improving Stock Movement Prediction with Adversarial Training, 2018.

[20] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using

stacked autoencoders and long-short term memory, 2017.
[21] M. Hagan, H. Demuth, M. Hudson Beale and O. De Jesus, Neural Network Design, Second Edition, Ed. Martin Hagan, 2012
[22] M. Louis, Trend Forecasing With Technical Analysis, 2000, ISBN 1-883272-91-2, pg. 35.
[23] R. Hyndman, Moving averages, 2009, pg. 1.
[24] J. Bollinger, Bollinger On Bollinger Bonds, McGraw-Hill, 2002.
[25] M. Butler, D. Kazakov, A learning adaptive Bollinger band system, in IEEE Conference on Computational Intelligence on Financial Engineering and Economics, (2012), pp. 1–8.
[26] Available online at: https://www.investing.com/
[27] L.Y. Ren & P. Ren, Revised Mean Absolute Percentage Errors (MAPE) on Errors from Simple Exponential Smoothing Methods for Independent Normal Time Series, Proceedings of SWDSI, Oklahoma, USA, pp. 602-611, 2009.
[28] R. Nau, Forecasting with Moving Averages. Fuqua School of Business, Duke University, 2014

Catalina-Lucia COCIANU, Professor, PhD, currently working with Bucharest University of Economic Studies, Faculty of Cybernetics, Statistics and Informatics, Department of Informatics and Cybernetics in Economy. Competence areas: machine learning, statistical pattern recognition, digital image processing. Research in the fields of pattern recognition, data mining, signal processing. Author of 20 books and more than 100 papers published in national and international journals and conference proceedings.

Mihai-Șerban AVRAMESCU graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2015. He has graduated the Economic Informatics Master program organized by the Bucharest University of Economic Studies, in 2017. In present he is working on his PhD degree in Economic Informatics with the Doctor's Degree Thesis: "Analyzing and predicting economic data using Machine Learning techniques".