Comparison between ARIMA and LSTM for EUR/USD prediction

Vo Minh Hieu

Abstract

Analysis of currency pair fluctuations has been of paramount importance for governments, financial institutions, and investors. Exchange rates play a crucial role as indicators guiding the economic policies of nations. Various models have been employed to forecast such time series data with reasonable accuracy. However, due to the inherently stochastic nature of these time series, achieving robust forecasting performance poses a significant challenge. In this investigation, we compare the effectiveness of a traditional statistical model and a deep learning model in predicting the EUR/USD exchange rate. The two evaluated models are Autoregressive Integrated Moving Average (ARIMA) and a Recurrent Neural Network utilizing Long Short-Term Memory (LSTM). The analyzed dataset spans from January 2, 2004 to March 1, 2023, and is segmented into training and validation datasets. The research findings reveal that, after 1000 epochs, the deep learning model, represented by the LSTM model, demonstrates high predictive accuracy, whereas the traditional statistical forecasting model ARIMA exhibits almost negligible predictive capability.

Keywords: LSTM, ARIMA, Random series modeling, EUR/USD exchange rate, Price prediction

1. Introduction

Since the inception of the foreign exchange (FOREX) market in the 1970s [1–3], a multitude of models have emerged for forecasting stock market prices. These models encompass various approaches, such as fundamental analysis, technical analysis, and hybrid analysis that combines statistical methods to model price behavior for generating future predictions [4,5]. Among the classical forecasting methods, the autoregressive combined moving average model (ARIMA) holds a prominent position [6]. Additionally, since the early 1990s, artificial neural networks (ANN) have been employed in economic and financial data studies as estimation and forecasting methods, exhibiting non-linear properties [7,8]. Deep Neural Networks (DNN), a class of neural networks designed based on Lyapunov's stability theory, have been utilized for learning given laws [8,9], demonstrating applications in forecasting stock indices such as DAX and S&P 500 [7].

In the pursuit of analyzing variable weights, a neural network (NN) system incorporating 12 economic variables was employed to assess their importance in forecasting the Peso/VND exchange rate [10]. Over the past decade, recurrent neural networks (RNN), particularly those employing short-long-term memory (LSTM), have gained widespread usage in predicting sequential data [11–14]. The unique mechanism of LSTM networks, allowing them to store long- and short-term information, positions them as powerful tools for forecasting historical data [15–18]. The application of LSTM models extends to forecasting currency pairs, trading actions on the New York Stock Exchange, and environmental prediction, with comparisons made against other neural networks and classical prediction methods [15–18].

Various comparisons and applications have been conducted to develop hybrid models aimed at improving prediction results [19–21]. These studies have played a pivotal role in elucidating pathways for creating new approximations based on standard methods, particularly applied to forecasting exchange rates and stock market fluctuations [24]. The challenge lies in achieving the best short-term forecast, given the inherent random fluctuations and noise in time series data within the financial sector.

In this study, we delve into a comparative analysis of the accuracy of two models, ARIMA and LSTM, in forecasting the EUR/USD exchange rate. The objective is to furnish valuable insights into the accuracy of deep learning models vis-à-vis classical statistical models. The study is presented as follows: (1) It commences with a concise overview of ARIMA and LSTM models, elucidating their algorithms and optimization strategies. (2) Subsequently, we detail the datasets used, segregated into distinct datasets for training and testing purposes. (3) The study then outlines the methodology for constructing both models, including parameter specifications. (4) Finally, the results of the two models are presented to facilitate comparisons and draw conclusive findings.

2. A brief overview of ARIMA and LSTM models

2.1. ARIMA

Classical statistical methods were traditionally employed for analyzing time series behavior, and the ARIMA model stands out as a renowned statistical approach developed through linear regression [6,25]. Originating from the research of George Box and Gwilym Jenkins, the ARIMA model is commonly referred to as the Box – Jenkins method [25]. Figure 1 illustrates the ARIMA algorithm, utilizing graphs, basic statistics, autocorrelation function (ACF), partial autocorrelation function (PACF), and transformations to identify patterns and model components. The model generates estimates through Ordinary Least Squares and Maximum Likelihood methods, employing ACF and PACF

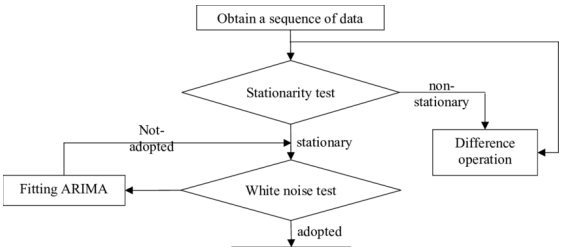

Figure 1: Flow chart of ARIMA model

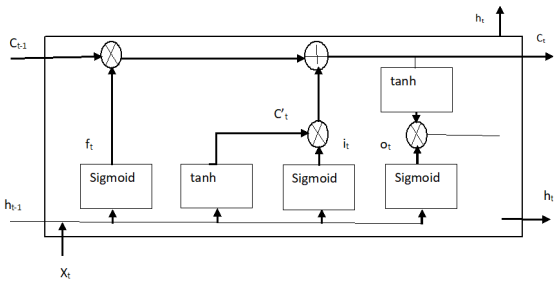

Figure 2: Diagram of a LSTM cell

plots of residuals to validate the model's accuracy. Ultimately, the algorithm predicts and assesses model performance using confidence intervals and basic statistical measures.

2.2. LSTM

Recurrent Neural Networks (RNNs), abbreviated as RNN, served as the precursor to Long Short-Term Memory (LSTM) networks and laid the groundwork for processing sequential data. RNNs were designed to capture temporal dependencies by maintaining hidden states that evolve over time. However, inherent challenges such as vanishing gradients impede their effective learning from lengthy sequences. Addressing these limitations, LSTM networks were introduced in 1997 by Sepp Hochreiter and Jürgen Schmidhuber, marking a significant advancement in deep learning [26].

LSTM represents a specialized iteration of RNN, tailor-made to overcome the constraints of its predecessor. A distinctive feature of the LSTM architecture is the incorporation of a memory cell, complemented by intricate gate mechanisms. These gates—forget, input, and output—regulate the controlled flow of information within the cell, enabling the model to selectively retain, update, or discard information. The gates are governed by sigmoid and hyperbolic tangent activation functions, empowering the model to learn and adapt to intricate sequential patterns.

In addition to the memory cell, LSTM maintains a hidden state, ensuring the preservation of pertinent information for subsequent time steps. The computations within an LSTM cell encompass forget gate computations, input gate computations, cell state updates, and output gate determinations, collectively contributing to the generation of the hidden state and the evolving cell state.

The training process of the LSTM model involves the application of Backpropagation Through Time [27], a mechanism for calculating gradients over the model parameters. Optimization algorithms such as Stochastic Gradient Descent [28] are subsequently employed to fine-tune these parameters, facilitating the iterative learning process.

In Figure 2, the input and output are denoted by x_t and y_t, respectively, with vector ht representing short-term memory and ct representing long-term memory. For time series predictions of x_t, the LSTM system updates the memory cell c_t and outputs the hidden state ht for each step t. Equation (1-6) elucidates the mechanism of LSTM [14,29]:

$$i_t = \sigma\left(W_{xi}^T x_t + W_{hi}^T h_{t-1} - 1 + b_i\right) \tag{1}$$

$$f_t = \sigma\left(W_{xf}^T x_t + W_{hf}^T h_{t-1} - 1 + b_f\right) \tag{2}$$

$$o_t = \sigma\left(W_{xo}^T x_t + W_{ho}^T h_{t-1} - 1 + b_o\right) \tag{3}$$

$$g_t = tanh\left(W_{xg}^T x_t + W_{hg}^T h_{t-1} - 1 + b_g\right) \tag{4}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{5}$$

$$y_t = h_t = o_t \otimes tanh(c_h) \tag{6}$$

The LSTM cell comprises the forget gate f_t, input gate i_t, and output gate o_t, determined by the input x_t and the preceding short-term state h_{t-1}, incorporating the gate g_t. The forget gate is represented by the standard logistic sigmoid function, denoted as $\sigma(x) = \frac{1}{1+e^{-x}}$, and tanh is denoted as $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

The weight matrices $W_{xi}, W_{xf}, W_{xo}, W_{xg}$ and $W_{hi}, W_{hf}, W_{ho}, W_{hg}$ connect to short-term input vectors x_t and h_{t-1}, respectively. Additionally, bias terms b_i, b_f, b_o, b_g correspond to each of the four layers. Notably, b_f is initialized after 1 second to prevent early memory loss during training [14,29].

3. Data preparation

In this research, we utilize time series data that reflects the daily values of the EUR/USD exchange rate for predicting its future value on the following day. The dataset, spanning from January 2, 2004 to March 1, 2023, consists of 4999 observations sourced from the Investing.com database. Each observation corresponds to the EUR/USD exchange rate from Monday to Friday. To facilitate our analysis, we partition the time series dataset into distinct training and testing sets. The training set encompasses 4899 observations, spanning from January 2, 2004, to October 12, 2022. Notably, the validation set comprises the final 100 observations in the dataset.

4. Methodology

Figure 3 illustrates that the time series data of EUR/USD lacks a discernible pattern over time, with peaks not oscillating around the mean. Rather, the peaks are distant from the average and display a seasonality indicative of a random time series. The time series records a minimum price of 0.9592 and a maximum price of 1.5988, with an average daily price of 1.2443. Approximately 47.64% of the time, the data is at or above the market average, while 52.35% of the time, it falls below the average.

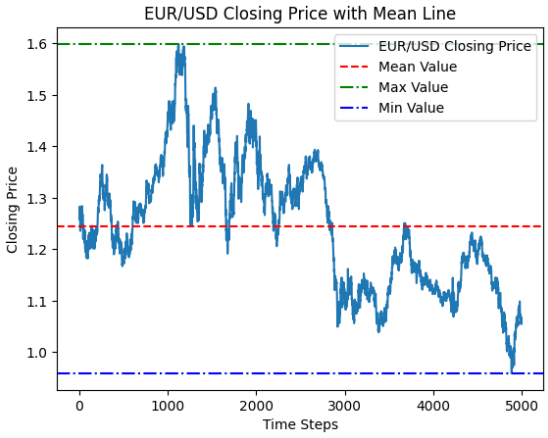

Figure 3: The movement of closing prices of EUR/USD

Python Output: Augmented Dickey - Fuller Test
ADF Statistics = -1.799832
P-value = 0.380545
Alternative Hypothesis: stationary

Table 1: The stationarity test for time series at the price level

4.1. ARIMA

Given the stochastic nature of the time series data, we perform a series analysis to select the most suitable ARIMA model, following the ARIMA algorithm depicted in Figure 1. As the ARIMA model assumes stationarity in the time series, we assess its stationarity using the Dickey–Fuller test [30] at a significance level of $\alpha = 0.05$. The analysis is conducted using the Python programming language in Google Colab, a modified version of Jupyter Notebook enabling the execution of Python code online through Google's cloud service.

Table 1 shows $p_{value} = 0.380545 > \alpha = 0.05$; therefore, the time series is non-stationary. Hence, the time series is considered non-stationary. To induce stationarity, we apply a transformation to convert the characteristic into a return rate using equation (7). The outcomes of the ADF test for this modified feature are presented in Table 2.

$$R_t = \left(\frac{P_t - P_{t-1}}{P_{t-1}} \right) \tag{7}$$

where: R_t is the rate of return on the day t. P_t and P_{t-1} are the closing price on the day t and $t-1$.

Python Output: Augmented Dickey - Fuller Test
ADF Statistics = -70.784176
P-value = 0.000000
Alternative Hypothesis: stationary

Table 2: The stationarity test for time series at the return level

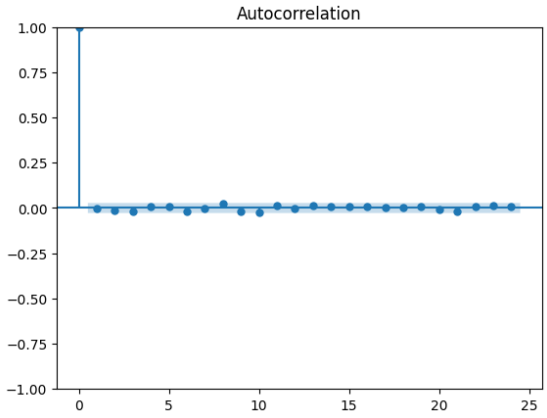

Figure 4: The ACF plot for return rates of EUR/USD

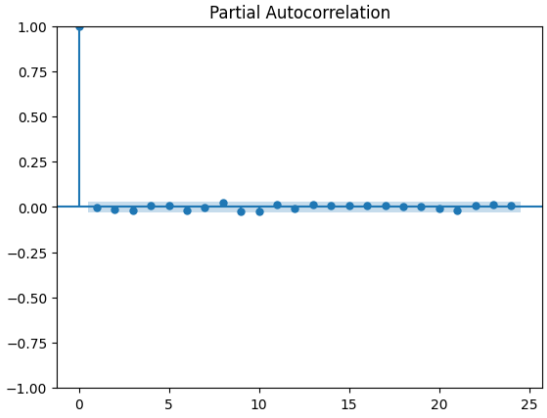

Figure 5: The PACF Plot for return rates of EUR/USD

Table 2 shows $p_{value} = 0 < \alpha = 0.05$; therefore, the time series is stationary. Consequently, in light of this outcome, we opt to employ an ARIMA model with $I = 0$, utilizing the profit margin variable instead of the closing price of EUR/USD. To determine the model parameters q and p, we examine the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots. Positive autocorrelation in the ACF plot indicates the possible need for a moving average (MA) term, aiding in the estimation of q, the order of the moving average component. Additionally, the PACF plot provides an estimate for p, the autoregressive part (AR) parameter, based on the presence of partial positive autocorrelation [31].

Figures 4 and 5 display time series data of returns devoid of autoregressive (AR) or moving average (MA) components. This indicates that the observed values in the time series are uncorrelated and do not necessitate differentiation for achieving stationarity. With parameter estimates of p=0, q=0, and I=0, the appropriate ARIMA model selection is ARIMA(0,0,0). The ARIMA(0,0,0) model implies constant and equal future values of the time series, equivalent to the average of observed values. This simplistic model suggests that historical values alone are adequate for predicting future values without considering any trend or seasonality. This approach is notably inadequate for capturing the fluctuations in the EUR/USD exchange rate, po-

tentially leading to unreliable predictions. Nevertheless, for the sake of comparison with the deep learning model, we proceed to use the ARIMA model on the time series data of returns.

4.2. LSTM

4.2.1. Data preprocessing

In contrast to classical statistical models, deep learning models like LSTM are not bound by initial assumptions, such as the requirement for the time series to be stationary or autocorrelated. Consequently, there is no need for preliminary statistical tests of the LSTM model, allowing for the direct application of the model to predict the EUR/USD closing price.

For predicting the next day's closing price of EUR/USD, three types of features are utilized: the historical value of the closing price, along with the highest and lowest prices of EUR/USD. This feature selection method aligns with common practices observed in previous studies, as evidenced by [32-34]. To facilitate effective learning and prevent overfitting, the data is normalized using the min-max scaling method. Normalizing data through min-max scaling, a prevalent technique in data preprocessing, brings variable values within the same range from 0 to 1 [35]. This approach ensures that variables with disparate scales share a uniform range, reducing the sensitivity of machine learning models or optimization algorithms to variables with larger values.

The min-max scaling formula is computed as follows:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{8}$$

where X is the original value of the variable, X_{min} is the minimum value of the variable, X_{max} is the maximum value of the variable, X_{scaled} is the scaled value which ranges from 0 to 1

4.2.2. The proposed model

The model takes in three standardized time series, representing the closing price, highest price, and lowest price as features. Employing a lag of $t = 30$, equivalent to one month of historical data, the input tensor X_t is depicted in Figure 6 as a composite of three normalized price matrices. In which, V_t, V_t^{hi}, and V_t^{lo} denote three standardized price matrices corresponding to the closing price, highest price, and lowest price, respectively.

The short-long-term memory (LSTM) is implemented in Python, utilizing the PyTorch support library. The model comprises an input layer, two stacked LSTM layers, and three fully-connected (linear) layers responsible for calculating the model output. The input size is specified as 3, corresponding to the closing price, highest price, and lowest price, while the hidden layer encompasses 12 hidden units. The inclusion of two stacked LSTM layers ensures a hierarchical representation of temporal dependencies in the data [35].

To introduce nonlinearity into the model, the rectified linear unit (ReLU) function is applied after each linear layer, excluding the linear layer for output. The Kaiming initialization method is employed to initialize weights for the linear layers, promoting effective learning [36]. For training, L1 Loss is chosen as the loss function, a suitable option for regression tasks.

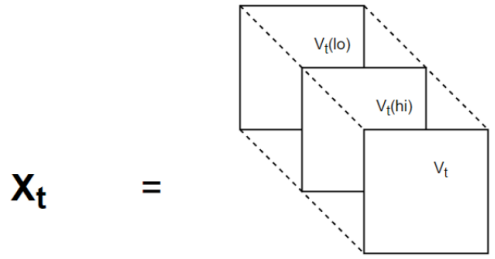

Figure 6: The input tensor for the LSTM model

ARIMA(1,0,1)
const
-0.0004 (0.0008)
ar.L1
-0.0015 (3.6647)
ma.L1
-0.0015 (3.6644)
sigma
20.0033*** (0.0000)
Standard errors in parentheses.
$*p < .1, **p < .05, ***p < .01$

Table 3: The estimates of the ARIMA(1,0,1) model

Additionally, the Adam optimization method, with a learning rate of 0.001, is employed to optimize the model parameters during training [37].

5. Results and discussions

In this section, we present the results of two models on a validation dataset comprising 100 time steps. Despite the absence of guaranteed adherence to the initial statistical assumptions for the ARIMA model, we still employ the ARIMA(1,0,1) model with the dependent variable being the daily EUR/USD return. The results align with expectations, indicating that the ARIMA model lacks forecasting ability for EUR/USD data.

Table 3 displays the estimates of the ARIMA(1,0,1) model for the time series parameters, specifically aimed at predicting EUR/USD daily returns. The estimated constant is approximately -0.0004, suggesting a baseline value for daily returns. The autoregressive (AR) coefficient for lag 1 is around -0.0015, signifying a negative relationship with the previous day's return. Similarly, the moving average (MA) coefficient for lag 1 is also approximately -0.0015, representing the impact of yesterday's error on the current return. The error variance (sigma2) is estimated to be 0.0033, and this estimate holds statistical significance at a high confidence level ($***p < .01$). However, the standard errors of the constant coefficients, AR, and MA are

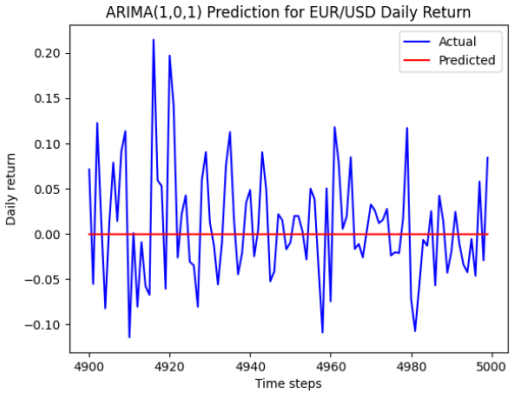

Figure 7: The prediction values by ARIMA(1,0,1)

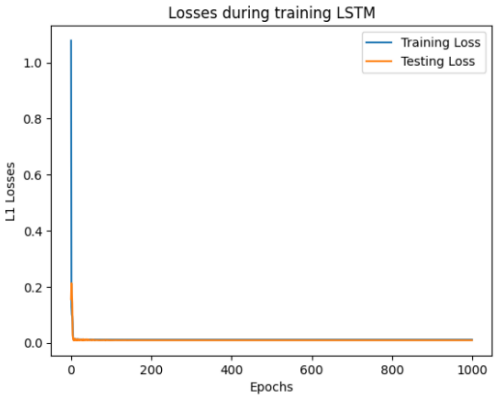

Figure 8: The track of losses during a training process of LSTM model

relatively high, hinting at the possibility of instability or weak statistical support for these estimates.

Observing Figure 7 makes it evident that the ARIMA(1,0,1) model exhibits zero prediction ability. The predicted result equals the average value of the time series data. This aligns with the outcomes of the initial statistical tests, implying that the series is not autocorrelated, and the future values of the time series are expected to be constant and equal to the observable average.

For the LSTM model, following a training process spanning 1000 epochs, the model demonstrates significantly superior predictive accuracy compared to the ARIMA model. Illustrated in Figure 8, the loss function is substantially reduced to 0.0108 for the training dataset and 0.009 for the validation dataset after just the initial 100 epochs. Further clarity on the model's outcomes is provided in Figures 9 and 10, where the predicted value's price graph is juxtaposed with the actual value's price graph. The predicted value exhibits a correlation of over 99% with the actual value in both the training and testing sets. Concurrently, the model's error in the test set is minimal, with approximately 0.00578 for the L1 loss function and 0.005713 for the mean squared error (MSE) function. As a representative of deep learning models, the LSTM model's favorable results in comparison to the ARIMA model suggest the potential superiority of deep learning models over classical statistical models.

However, practical application of the LSTM model still faces several challenges. Although the model's error is lower than that of ARIMA, it remains relatively large for foreign exchange transactions. The L1 loss function for the test set is 0.005713, equivalent to an error of 57 pips or 570 prices. In daily frequency trading, where the acceptable spread is approximately 20 pips, this error is noteworthy. Additionally, Figure 10 reveals that the price movement of predicted values consistently lags behind the actual values. This lag occurs because the model relies entirely on features related to the historical data of the price line, and its ability to forecast the future is not highly reliable.

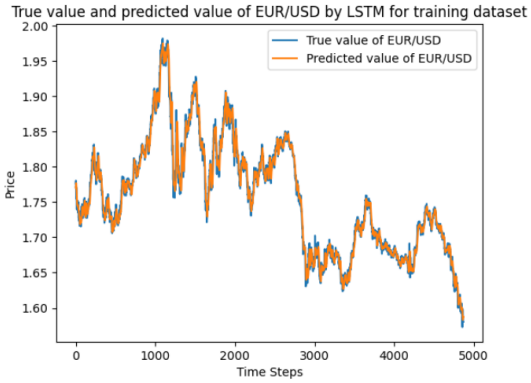

Figure 9: Comparison of true values and predicted values by LSTM model for training dataset

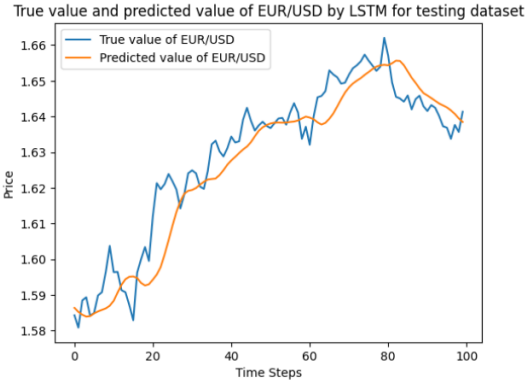

Figure 10: Comparison of true values and predicted values by LSTM model for testing dataset

6. Summary and conclusions

In this research, we employ two models, ARIMA and LSTM, to compare and assess their predictive capabilities regarding the EUR/USD exchange rate, representing two branches of models: statistical machine learning and deep learning. The dataset utilized comprises daily EUR/USD exchange rates, characterized as a type of time series data exhibiting random walks and a lack of stationarity. Even after transforming the data into daily profit rates, the time series fails to exhibit the autocorrelation property necessary for the effective utilization of the traditional ARIMA model. Empirical testing supports this observation, as the ARIMA(1,0,1) model predicts a constant equal to the historical average value of the profit rate.

Conversely, the LSTM model, not constrained by initial statistical assumptions, yields significantly higher prediction accuracy, achieving a correlation with actual values exceeding 99%. The model's error is also notably lower when compared to classical statistical models, registering at 0.005713. However, practical implementation requires substantial enhancements in the model architecture to achieve acceptable errors.

The demonstrated superiority of the LSTM model over ARIMA in this study provides a crucial indication of the heightened accuracy offered by modern deep learning models compared to traditional statistical models. Future research efforts may concentrate on further refining deep learning models to enhance prediction accuracy, particularly for forex pairs and financial asset classes in general.

References

[1] Goodman, S.H. Foreign Exchange Rate Forecasting Techniques: Implications for Business and Policy. J. Finance. 1979, 34, 415–427.

[2] Pacelli, V. Forecasting Exchange Rates: A Comparative Analysis. Int. J. Bus. Soc. Sci. 2012, 3, 12.

[3] Canova, F. Modelling and Forecasting Exchange Rates with a Bayesian Time-Varying Coefficient Model. J. Econ. Dyn. Control 1993, 17, 233–261.

[4] Jung, G.; Choi, S.-Y. Forecasting Foreign Exchange Volatility Using Deep Learning Autoencoder-LSTM Techniques. Complexity 2021, 2021, e6647534.

[5] Tripathi, M.; Kumar, S.; Inani, S.K. Exchange Rate Forecasting Using Ensemble Modeling for Better Policy Implications. J. Time Ser. Econom. 2021, 13, 43–71.

[6] Nyoni, T. Modeling and Forecasting Naira / USD Exchange Rate In Nigeria: A Box—Jenkins ARIMA Approach. MPRRA 2018, 88622, 1–36.

[7] Arango, F.O.; Cabrera Llanos, A.I.; Herrera, F.L. Pronóstico de los índices accionarios DAX y SP 500 con redes neuronales diferenciales. Contaduría Y Adm. 2013, 58, 203–225.

[8] Poznyak, A.S.; Sanchez, E.N.; Yu, W. Differential Neural Networks for Robust Nonlinear Control; World Scientific: Singapore, 2001;pp. 12–50.

[9] Arango, F.O.; Aranda, F.C. Redes Neuronales Diferenciales: Una Alternativa Confiable Para Analizar Series Financieras. In Proceedings of the XVI Congreso Internacional de Contaduría Administración e Informática; Mexico, D.F., Ed.; Medellín, U. D.: Ciudad de México, Mexico, 2011; pp. 49–64.

[10] Zapata, L.A.; Hugo, D. Predicción Del Tipo de Cambio Peso-Dólar Utilizando Redes Neuronales Artificiales (Rna). Pensam. Gestión 2008, 24, 29–42.

[11] Van Houdt, G.; Mosquera, C.; Nápoles, G. A Review on the Long Short-Term Memory Model. Artif. Intell. Rev. 2020, 53, 5929–5955.

[12] Yıldırım, D.C.; Toroslu, I.H.; Fiore, U. Forecasting Directional Movement of Forex Data Using LSTM with Technical and Macroeconomic Indicators. Financ. Innov. 2021, 7, 1.

[13] Zhang, C.; Fang, J. Application Research of Several LSTM Variants in Power Quality Time Series Data Prediction. In Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition; Association for Computing Machinery: New York, NY, USA, 2019; pp. 171–175.

[14] Choi, J.Y.; Lee, B. Combining LSTM Network Ensemble via Adaptive Weighting for Improved Time Series Forecasting. Math. Probl. Eng. 2018, 2018, 1–8.

[15] Babu, A.S.; Reddy, S.K. Exchange Rate Forecasting Using ARIMA, Neural Network and Fuzzy Neuron. J. Stock Trad. 2015, 4.

[16] Adebiyi, A.A.; Adewumi, A.O.; Ayo, C.K. Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. J. Appl. Math. 2014, 2014, 1–7.

[17] Li, M.; Ji, S.; Liu, G. Forecasting of Chinese E-Commerce Sales: An Empirical Comparison of ARIMA, Nonlinear Autoregressive Neural Network, and a Combined ARIMA-NARNN Model. Math. Probl. Eng. 2018, 2018, 1–12.

[18] Son, H.; Kim, C. A Deep Learning Approach to Forecasting Monthly Demand for Residential–Sector Electricity. Sustainability 2020, 12, 3103.

[19] Wang, J.-J.; Wang, J.-Z.; Zhang, Z.-G.; Guo, S.-P. Stock Index Forecasting Based on a Hybrid Model. Omega 2012, 40, 758–766.

[20] Islam, M.S.; Hossain, E. Foreign Exchange Currency Rate Prediction Using a GRU-LSTM Hybrid Network. Soft Comput. Lett. 2020, 100009.

[21] Musa, Y.; Joshua, S. Analysis of ARIMA-Artificial Neural Network Hybrid Model in Forecasting of Stock Market Returns. Asian J. Probab. Stat. 2020, 42–53.

[22] Wang, J.-N.; Du, J.; Jiang, C.; Lai, K.-K. Chinese Currency Exchange Rates Forecasting with EMD-Based Neural Network. Complexity 2019, 2019, e7458961.

[23] Qiu, Y.; Yang, H.-Y.; Lu, S.; Chen, W. A Novel Hybrid Model Based on Recurrent Neural Networks for Stock Market Timing. Soft Comput. 2020, 24, 15273–15290.

[24] Hu, Z.; Zhao, Y.; Khushi, M. A Survey of Forex and Stock Price Prediction Using Deep Learning. Appl. Syst. Innov. 2021, 4, 9.

[25] Box, G. and Jenkins, G. (1970) Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco.

[26] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[27] Werbos, P. J. (1990). "Backpropagation through time: what it does and how to do it." Proceedings of the IEEE, 78(10), 1550-1560.

[28] Robbins, H., Monro, S. (1951). "A stochastic approximation method." The Annals of Mathematical Statistics, 22(3), 400-407.

[29] Géron, A. Hands-On Machine Learning with Scikit-Learn and TensorFlow; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017; ISBN 978-1-4919-6229-9.

[30] Dickey, D. A., Fuller, W. A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." Journal of the American Statistical Association, 74(366a), 427–431.

[31] Wooldridge, J. M. (2003). Introductory Econometrics: A Modern Approach. Chapter 6: "Univariate Time-Series Modelling and Forecasting," Section 6.6: ARMA Processes, pages 351-352.

[32] Shen, S., Jiang, H., Zhang, Y. (2015). "Stock Market Forecasting Using Machine Learning Algorithms." International Journal of Computer Science and Information Technology Research, 3(2), 177-189.

[33] Kitsos, C., Ntzoufras, I., Kourentzes, N. (2018). "Time Series Forecasting of High-Frequency Financial Markets Data." Journal of Financial Data Science, 1(1), 24-36.

[34] Yang, Z., Liu, X., Tan, X. (2017). "Long Short-Term Memory Recurrent Neural Network for High-Frequency Trading Decision." Journal of Computational Science, 24, 216-222.

[35] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.

[36] He, K., Zhang, X., Ren, S., Sun, J. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1026-1034.

[37] Kingma, D. P., Ba, J. (2014). "Adam: A Method for Stochastic Optimization." arXiv preprint arXiv:1412.6980