Time Series Forecasting Using Artificial Neural Networks
A Model for the IBEX 35 Index

Daniel González-Cortés[1(✉)], Enrique Onieva[2], Iker Pastor[2], and Jian Wu[1]

[1] NEOMA Business School, rue du Maréchal Juin, 76825 Mont Saint Aignan Cedex, France
{daniel-alejandro.gonzalez-cortes.20,jian.wu}@neoma-bs.com
[2] Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain
{enrique.onieva,iker.pastor}@deusto.es

Abstract. The amount of data generated daily in the financial markets is diverse and extensive; hence, creating systems that facilitate decision-making is crucial. In this paper, different intelligent systems are proposed and tested to predict the closing price of the IBEX 35 using ten years of historical data with four different neural networks architectures. The first was a multi-layer perceptron (MLP) with two different activation functions (AF) to continue with a simple recurrent neural network (RNN), a long-short-term memory (LSTM) network and a gated recurrent unit (GRU) network. The analytical results of these models have shown a strong, predictable power. Furthermore, by comparing the errors of predicted outcomes between the models, the LSTM presents the lowest error with the highest computational time in the training phase. Finally, the empirical results revealed that these models could efficiently predict financial data for trading purposes.

Keywords: Machine learning · IBEX35 · Stock market prediction · Artificial neural networks · Recurrent neural network · Gated recurrent unit · Long-short-term memory

1 Introduction

The main concern of many economic agents is to forecast the future trends of the financial markets to make better decisions. The methods used and the time frames to predict are diverse. The stock markets represent a fundamental piece of any modern economy by letting investors exchange financial instruments at an agreed price with many fluctuations over time. These variations are considered chaotic and non-stationary; however, there is some empirical evidence [18] suggesting that stock returns can have some predictable components rejecting the hypothesis of the random walks.

All the economic agents need to be aware of the stock market's implications at different economic levels. As seen in the global financial crisis of 2007–2009,

P. García Bringas et al. (Eds.): HAIS 2022, LNAI 13469, pp. 249–260, 2022.
https://doi.org/10.1007/978-3-031-15471-3_22

the financial contagion affected different sectors of the real economy, such as consumer goods, industrials, telecommunications and technology [4]. Therefore, forecasting future stock prices and trends can be crucial for better financial decisions. However, this is not an easy task because the nature of the stock market is intrinsically nonlinear, non-parametric and chaotic, where many variables interact, making prices move in one direction or another.

The prediction process in the stock market has been approached by two different methodologies, fundamental and technical analysis. The first one is based on the valuation of the intrinsic value of stocks by using the current and future earnings of the company to evaluate the fair value and then contrast this information with the market value indexed in the stock exchange. The second methodology does not count on the company's financial statements as the primary source of information. However, it merely relies on data using historical stock prices to make predictions trying to identify statistical trends.

Many investors use both methodologies to make buying or selling decisions, and the 87 % of fund managers use some technical analysis [19]. However, the increasing expansion and evolution of datification and automation prompted the financial markets to find new processes to remain competitive and reinvent their services. Then artificial intelligence and machine learning became a powerful tool for institutions, financial advisers, banks and wealth managers and disruptively transformed their business model [17].

This research focuses on studying Artificial Neural Networks (ANN) and attempts to clarify further the use of its different variations in predicting the stock market. This work starts with a bibliographic review in Sect. 2, about the use of ANN to solve some financial problems and continues with the Materials and methods in Sect. 3 to explain the initial settings of the experimentation. Then, to continue in Sect. 4 with the description of ANNs, starting with the description and creation of an Multi-Layer Perceptron (MLP) network in Sect. 4.1. Subsequently, the structure of a Recurrent Neural Network (RNN) is presented, introducing the Simple RNN, Long-Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures in Sect. 4.2. Finally, to finish with the results and discussion and conclusions about the performance of these models predicting financial times series in section in Sect. 5 and 6 respectively.

2 Background

Over the years, ANNs have played a vital role in the decision-making process of banks and financial institutions due to the adaptation of new and automated systems to their operations. These new technologies have been quickly adopted because they are consistent and objective, eliminating human bias or wrong assumptions. In addition, ANNs empower investors and institutions to create new powerful models by extracting information from past observations and improving preconceived models.

The ANN are a bio-inspired computing system based on many connected processors called neurons activated by different types of activation functions

(AF) triggered according to specific weights and bias. This system aims to minimize the prediction error by using a feed-forward optimization that will change the weights of the different interconnected neurons. This model can capture nonlinearities and has been used in different business applications to predict financial distress, bankruptcy analysis, stock price predictions, and credit scoring Tkac et al. [26].

Different ANN applications to predict the stock market can be found in the literature. For example, Qui et al. [23] used an ANN to predict the return of the Japanese Nikkei 225 index by using a hybrid approach based on a genetic algorithm and simulated annealing. Additionally, Pyo et al. [22] analyzed the prediction of a stock exchange index, building three hypotheses to forecast the daily closing prices of the Korea Stock Price Index 2000 by using an ANN and two SVMs models. Also, Kara et al. [14] using a three-layered feed-forward ANN and an SVM model, predicted the direction of the Istanbul stock exchange index, concluding that the 75.74% performing prediction for the ANN was significantly better than for the SVM. Moreover, to predict the stock market index Moghaddam et al. [20] created an ANN using the four and nine previous days as inputs, concluding that there is no distinct difference between using different days.

Sagir et al. [24] presented a contrast between ANNs and classical statistical techniques to predict the Malaysian stock market index using three variables, showing that the ANN was more accurate than the multiple linear regression model in this research, with a coefficient of determination of 0.9256. Also, Ariyo Adebiyi et al. [2] compared an autoregressive integrated moving average with an ANN to predict the price of a single stock using 5680 observations. The authors concluded that the ANN model is better, having a higher forecasting accuracy; however, there is not statistically significant.

An example of the integration of metaheuristics to predict financial data with neural networks was performed by Gocken et al. [11], where an ANN is hybridized with a genetic algorithm and harmony search to make a feature selection to reduce the complexity of variable selection. In their model, the inputs were technical indicators concluding that hybrid ANN can be successfully used to forecast the stock market price movement. Additionally, Enke et al. [10] introduced an information gain technique to evaluate the prediction of stock market returns using data mining and ANN for level estimation and classification with macroeconomic variables as inputs. As a result, the ANN model was more accurate, showing more consistency than a linear regression forecast and generating higher profits than other strategies with the same risk exposure.

Another example of a hybrid model using ANN to predict financial time series is the work made by Kim et al. [15], combining LSTM, GARCH models and moving averages. These authors concluded that LSTM single models could effectively learn temporal patterns of time-series data with fewer prediction errors than deep feed-forward network-based integrated models.

3 Materials and Methods

This study coded and tested four different models using Python programming language. The data set used to create the inputs for this model was the Spanish Exchange Index, known as IBEX 35. The entire data set used in this research contains 2454 observations, covering the closing prices and volume values from June 1, 2010, to December 31, 2019, covering almost ten years of daily prices in which different trends took place that may represent a normal market cycle. The Anderson-Darling test was performed on the sample data and rejected the null hypothesis; therefore, the assumption of normal distribution in the data sample cannot be allowed. Because the values of the data-set did not follow a normal distribution, the closing prices and the volume values were rescaled between $(-1 < x < 1)$ to be used with the hyperbolic tangent functions and between $(0 < x < 1)$ with the sigmoid activation function. The performance metrics to measure the predictive ability of the different models used in this research were the mean square error (MSE), mean absolute error (MAE), mean squared log error (MLE) metrics, and the determination coefficient (R_2).

The common input layer for the ANN architectures use in this study is presented in Eq. 1:

$$X_{input}(t) = f[v(t-4), v(t-3), v(t-2), v(t-1), \\ c(t-4), c(t-3), c(t-2), c(t-1)] \tag{1}$$

where $c(t)$ is the function for closing price and $v(t)$ for volume value at a given time t.

For the training phase, the data-set was split into two parts. The first portion contains the initial 80% of the data selected for the training set, while the remaining 20% for the test sets.

4 ANN Architectures

This section first introduces a model that consists of an ANN with a multi-layer network structure coded by the authors, using NumPy library for matrix multiplication, and continues with another three models constructed using Keras and TensorFlow [1] to build a simple RNN, LTSM and GRU commonly used in different artificial intelligence projects.

4.1 Multi-layer Neural Network

The most widely implemented neural network topologies [12] is the MLP, a multi-layer network structure where the neurons are displayed as input, output, and hidden layers. The other components in the models are weights, connecting coefficients between layers, and activation functions that trigger a signal given a weighted sum of its input. In the first part of this research, two AFs used will be used, the sigmoid (SF) and hyperbolic tangent functions (HTF), while for

the simple RNN, LSTM and GRU the rectified linear units (RELU) activation function.

The training phase of an ANN model consists of four steps: Initialization of weights to small random values, forward pass, backward pass and updating of the weights and biases [24]. In this study, for the MLP architecture, two hidden layers will be used and one input and output layer; therefore, three weights matrices (SW_i) were created and initialized with random values for the initialization of weights. Where n_i is the numbers of nodes of the input layer, n_j for the first hidden layer and n_k for the second hidden layer.

For the forward pass, the first step was the multiplication between the input values of the vector $X_{input}(t)$ and the SW_1 matrix (2) were evaluated in the AF.

$$AF_1 = X_{input}(t) \cdot SW_1 \tag{2}$$

in the following steps, when i is greater than one:

$$AF_i = AF_{i-1} \cdot SW_i \tag{3}$$

Once the final layer is reached, the output value must be compared with the target value (4).

$$\delta = y_{output} - y_{target} \tag{4}$$

In the back pass, the error derivate goes back to the input layer updating the weights (SW_1, SW_2, SW_3). The first vector in the back pass is the derivative of the activation function of the δ multiplied by a learning rate (α); the remaining steps will be the multiplication of the first back pass vector by the transpose matrix of SW_3 and by the derivative of the second activation AF_2 and the same procedure for a third error vector (5)

$$e_1 = e_2 \cdot SW_2^T \cdot AF'(X_{input}) \tag{5}$$

Then e_1 multiply by the transpose matrix of X_{input} is updating SW_1 and then the following steps (6) until the last weight.

$$\Delta SW_i = AF_{i-1}^T \cdot e_i \tag{6}$$

4.2 Recurrent Neural Networks

RNN are one type of ANN that deals with data that has sequential inputs. This architecture has been used to process speech, language and sentiment data [27], specially predicting the next character and word in a given text [5], and for more complex tasks [16]. As mentioned previously, RNNs can process sequential inputs using an internal memory to process these incoming inputs and as Le Cun et al. [16] pointed, this model is able to keep in their hidden units a sort of state vector, which can enclose details of previous parts of the sequence. Therefore RNN can process at the same time the previous and recent flow of inputs data by using this hidden unit or layer to keep a historical record. However, the RNN only takes one sequence at any given time. In this research, three commonly used types of RNN will be tested. The following section will describe the LSTM and the GRU architecture.

LSTM. The advantage of an RNN is that it learns long term dependencies over time; however, there are some error back-flow problems [13], showing difficulties to achieve a proper learning process. Therefore the information can not be stored for a long time. A novel solution was introduced by hochreiter and schmidhuber [13] proposing the LSTM model in order to correct this problem, augmenting the network with explicit memory, using hidden units to remember short and long term values. The total number of units of the LSTM is displayed to create a network with an input node, an input, an output, and a forget gates, where the gates will regulate the flow of the information. In the following equations (7–12) is possible to see the forward pass of the LSTM unit [13].

$$g_t = \widetilde{C} = \varphi(W_g x_t + U_g h_{t-1} + b_g) \tag{7}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{8}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{9}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{10}$$

$$C_t = g_t * i_t + f_t * C_{t-1} \tag{11}$$

$$h_t = o_t * \varphi(C_t) \tag{12}$$

where i_t represents the input gate activation vector,f_t is the forget gate activation vector, o_t the output gate activation vector, C_t represents the cell state vector, h_t is the output vector of the LSTM unit, $\sigma$is SF function, b: biases for the respective gates, W, U are weight matrices for the respective gates, φ represents a HTF and $*$ is an element-wise product.

Several studies used LSTM to predict the stock market in the last five years, where this technique and its modifications dominate the financial time series forecasting [25]. However, despite their popularity, there are some variations on this model. One modification of LSTM is the GRU model that aims to resolve the vanishing and exploding gradient problem presented in the previous model [6].

GRU. This model was introduced by Cho et al. [8] in 2014, proposing a novel ANN called RNN Encoder-Decoder consisting of two RNNs. This model has fewer parameters and has proven a competitive performance to others models like LSTM. Furthermore, it is possible to observe that GRU and LSTM have gating units that modulate the flow of information inside the unit cell. However, it does not have a separate memory cell [9]. According to Alom et al. [3] this model is now popular among people who work with RNN because the computational cost and the simplicity of GRU are better compared to others. Furthermore,

the decrease in the computational cost is due to the absence of an output gate, accelerating the speed of the model [21]. GRUs have been successfully applied to many applications for pattern analysis where sequential data is used as input and multivariate time series with missing values [7]. Among the classical GRU's disadvantages is that it is very easy to fall into local minimum with small time series and complex time order. Apart from the sensitivity to the time order, Pei et al. [21] describe some disadvantages that this model has with the data, quoting that for GRUs is hard to detect the implication information of time series and that an imbalanced data can affect the performance of the model by influencing the convergence.

5 Results and Discussion

Diverse ANN structures were tested with the SF and HTF functions to determine the best topology for the MLP net, and in this investigation we have tested different values for the different layers, ranging from one to fifty nodes. As shown in Table 1 different results were obtained by changing the configuration of the ANN. The learning process was made with ten thousand iterations and an $\alpha = 0.01$, in all the MLP architectures tested.

Table 1. R_2 of the ANN model using the SF and the HTF.

		R_2	
n^o	Structure	*SF*	*HTF*
1	5-5	0.4789	0.6294
2	5-40	<0	0.5775
3	10-20	0.8646	0.7428
4	10-30	<0	0.7819
5	20-20	0.7172	0.7694
6	20-30	0.9242	0.7522
7	30-15	0.8213	0.8485
8	30-30	0.9213	0.8950
9	39-21	0.8166	0.9241
10	40-40	<0	0.9065
11	40-25	**0.9275**	0.9159
12	50-50	<0	0.8633

The prediction accuracy of the ANN model that uses the SF function is statistically different from the others that used HTF. The Wilcoxon signed-rank test was performed to compare the accuracy with the null hypothesis as follows:

$$\begin{aligned} H_0 &: \mu R_2/SF = \mu R_2/HTF \\ H_1 &: \mu R_2/SF \neq \mu R_2/HTF \end{aligned} \tag{13}$$

Even though it is possible to accept differences because the null hypothesis was rejected is not possible to affirm which activation function works better with this model, but the best coefficient of determination found was 0.9275, obtained by using the SF.

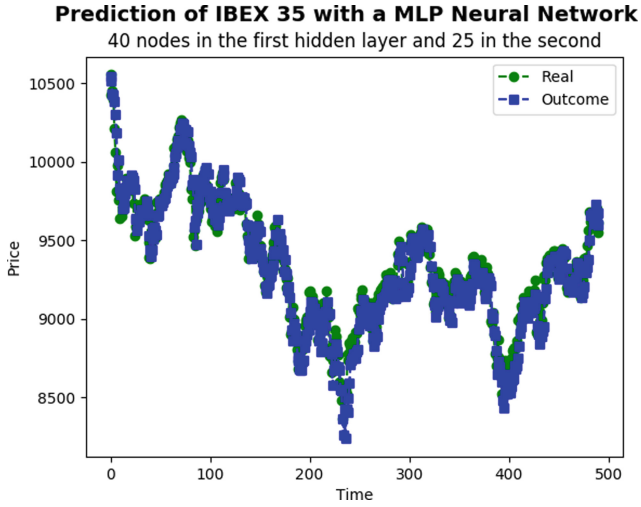

Fig. 1. Output and predicted values of the IBEX 35 index using SF (40-25), performed in test set.

When the number of nodes in the hidden layer is low, the model performs worse than with more numbers of nodes, this phenomenon was clearer using SF than HTF, but this might not always necessarily true. For example when using SF the R_2 is not constantly growing at a given rate. At first, a low number of nodes returned a low R_2, and as the number of nodes starts to increment also, the R_2 increases. However, when there are 40 nodes in each layer, the R_2 is negative; therefore, there is no optimal number of nodes for each layer. Also, negative R_2 values can be found in some specific ANN structures. The highest R_2 found was 0.9275 by using SF with 40 nodes in the first hidden layer and 25 in the second as shown in Fig. 1. For the ANN with HTF, the best R_2 found was 0.9241, as shown in Table 1, with an ANN with 39 and 21 nodes.

It is possible to observe that the ANN with SF has lower MSE, MAE, and MLE levels than those using HFT when the number of nodes is relatively low. However, this trend changed when the number of nodes was more extensive than 30 in the first hidden layer.

5.1 Recurrent Neural Network Performance

This section will present the implementation of the simple RNN, LSTM and GRU models. These models were implemented using the Keras library for Python

backup by the Tensorflow library. The data were pre-processed in the same way that was done for the previous models, but in the following cases, it was normalized only between $(0 < x < 1)$. The simple RNN sequential model was implemented using one Keras Simple RNN layer of 8 nodes and a hidden dense layer with 32 nodes and a RELU activation function. The MSE loss function was applied for compilation with an RMSprop optimizer, while epochs were 50 with a batch size of 1. The MSE for this model was 17520.86 while the $R_2 = 0.8814$. The MAE was 116.30, and MLE was equal to 0.00020.

For the LSTM, a sequential model was applied using one Keras LSTM layer of 32 nodes and one dense layer with eight nodes as an input to match the shape of the matrix that holds the arranged data. The loss function applied was the MSE with an ADAM optimizer and a RELU activation function. The number of epochs was equal to 50 with a batch size of 1. The results were, MAE = 69.33 with a MSE = 7192.27, while the $r_2 = 0.9513$ with a MLE = 0.000083. The GRU model was implemented using one Keras GRU layer of 8 nodes, one hidden dense layer with 32 nodes with a RELU activation function and an MSE loss function for compilation with an RMSprop optimizer. The number of epochs was 50 with a batch size of 1. The MSE for this model was 8608.07 while the $R_2 = 0.9417$. The MAE was 71.40, and MLE was equal to 0.000092.

5.2 Results Comparison

The best result in terms of R_2 is obtained by LSTM, with a $R_2 = 0.9513$ and GRU with a $R_2 = 0.9417$. Also, these two models presented the lowest errors compared to the others; however, it is also important to consider the computational time to analyse the model's performance fully. It is possible to observe in Table 2 the results of Simple RNN, LSTM and GRU models.

Table 2. Performance of MLP (SF 20-30), Simple RNN, LSTM and GRU models, using the test set.

	R_2	MSE	MAE	MLE	Time (seconds)
MLP	0.9275	10701.29	80.18	0.00012	**136.3**
RNN	0.8814	17520.86	116.30	0.00020	141.9
LSTM	**0.9513**	**7192.27**	**69.33**	**0.000083**	273
GRU	0.9417	8608.07	71.40	0.000092	228.6

The simple RNN showed the worst performance in terms of R_2, MAE, MSE and MLE errors; however, this is true compared to the best MLP results because some MLP configurations showed worst performance than RNN as shown in Table 2.

In this study, in order to have an analysis of the computational time, the platform Google Colaboratory was used as a tool for accelerating the learning

applications to have similar performance levels to those acquired with a dedicated hardware. The computational time results show that the model with the lowest computational time was the MLP made by the authors, with 40 and 25 nodes using an SF activation function. This model took 136.3 s to complete, while the Simple RNN was the second-fastest, taking 141.9 s. The model that took more time to be completed was the LSTM model taking 273 s to complete all the routines, followed by the GRU, which took 228.6 s. These performances took almost twice the time that MLP and Simple RNN.

6 Conclusions

Implementing different neural network architectures to forecast financial time series has shown a predictive capacity with low errors. Although all forms of ANN have successfully predicted the IBEX-35, LSTM has the best results. It is essential to consider that the ANN structure and the number of iterations in the training phase will determine the model's predictive capacity. However, there are no clear procedures to define a proper structure because the error does not decrease linearly. Therefore is necessary to include in further studies different types of heuristics to optimize the computational time of the training phase and the search for the optimal ANN architecture. Even though these ANNs have a high R2 at predicting the closing price of a stock market index, using these models as a tool for financial trading can be challenging because investors need to consider price predictions and the risk and the size of any given trading position. Hence, any potential strategy needs to consider rigorous risk management with a robust backtest.

In addition, because financial markets are interconnected, there might be other variables impacting the prices and bias of the market so that an extensive feature selection could improve the prediction of the model. Consequently, further research can be done to increment the number of significant variables used by the model and search for different ANN architectures to predict closing values and make a trading system that can be a reliable tool to predict financial markets.

In order to have proper management and performance of deep learning methods applied to financial time series, it is crucial to consider a robust data set with a proper and reliable testing phase for the model. Especially during times of high uncertainty, when the complexity and size of the financial markets grow. Due to the nature of the markets, the predicted outcomes will not necessarily represent future outcomes because there are rapid changes in the market dynamics. Thus, when picking the training data, it is necessary to keep in mind that one asset or instrument can only have one historical price record; therefore, any model trained to forecast financial data is vulnerable or prone to overfitting.

Not choosing a correct data set is dangerous because the model can predict outcomes similar or exact to the underlying data set but will fail to predict future values. Therefore, it is important to consider this vulnerability for future works, where models can be tested with historical data from multiple assets

and adding some synthetic data. Furthermore, using other deep learning models, such as generative adversarial models, could create endless data-sets for multiple scenarios that are not real but close enough to a potentially real scenario.

The forecast of future financial values is essential to investors and private companies and for government policymakers who need to make an appropriate asset allocation of scarce resources. In this way, better financial predictive models will not only help financial agents but could potentially affect everyone.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org
2. 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**, 1–7 (2014). https://doi.org/10.1155/2014/614342
3. Alom, M.Z., et al.: A state-of-the-art survey on deep learning theory and architectures. Electronics **8**(3), 292 (2019). https://doi.org/10.3390/electronics8030292
4. Baur, D.: Financial contagion and the real economy. J. Banking Financ. **36**(10), 2680–2692 (2012)
5. Bengio, Y.: Probabilistic neural network models for sequential data. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (2000). https://doi.org/10.1109/ijcnn.2000.861438
6. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994). https://doi.org/10.1109/72.279181
7. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Sci. Rep. **8**(1), 1–12 (2018). https://doi.org/10.1038/s41598-018-24271-9
8. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014). https://doi.org/10.3115/v1/d14-1179
9. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)
10. Enke, D., Thawornwong, S.: The use of data mining and neural networks for forecasting stock market returns. Expert Syst. Appl. **29**(4), 927–940 (2005). https://doi.org/10.1016/j.eswa.2005.06.024
11. Gocken, M., Ozcalici, M., Boru, A., Dosdogru, A.T.: Integrating metaheuristics and artificial neural networks for improved stock price prediction. Expert Syst. Appl. **44**, 320–331 (2016). https://doi.org/10.1016/j.eswa.2015.09.029
12. Guresen, E., Kayakutlu, G., Daim, T.U.: Using artificial neural network models in stock market index prediction. Expert Syst. Appl. **38**(8), 10389–10397 (2011). https://doi.org/10.1016/j.eswa.2011.02.068
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

14. Kara, Y., Boyacioglu, M.A., Baykan, Ö.K.: Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul stock exchange. Expert Syst. Appl. **38**(5), 5311–5319 (2011). https://doi.org/10.1016/j.eswa.2010.10.027
15. Kim, H.Y., Won, C.H.: Forecasting the volatility of stock price index: a hybrid model integrating LSTM with multiple GARCH-type models. Expert Syst. Appl. **103**, 25–37 (2018). https://doi.org/10.1016/j.eswa.2018.03.002
16. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
17. Lee, J., Suh, T., Roy, D., Baucus, M.: Emerging technology and business model innovation: the case of artificial intelligence. MDPI (2019). https://www.mdpi.com/2199-8531/5/3/44
18. Lo, A.W., MacKinlay, A.C.: Stock market prices do not follow random walks: evidence from a simple specification test. Rev. Financ. Stud. **1**(1), 41–66 (1988)
19. Menkhoff, L.: The use of technical analysis by fund managers international evidence. J. Banking Financ. **34**(11), 2573–2586 (2010). https://doi.org/10.1016/j.jbankfin.2010.04.014
20. Moghaddam, A.H., Moghaddam, M.H., Esfandyari, M.: Stock market index prediction using artificial neural network. J. Econ. Financ. Adm. Sci. **21**(41), 89–93 (2016). https://doi.org/10.1016/j.jefas.2016.07.002
21. Pei, S., Shen, T., Wang, X., Gu, C., Ning, Z., Ye, X., Xiong, N.: 3DACN: 3D augmented convolutional network for time series data. Inf. Sci. **513**, 17–29 (2020). https://doi.org/10.1016/j.ins.2019.11.040
22. Pyo, S., Lee, J., Cha, M., Jang, H.: Predictability of machine learning techniques to forecast the trends of market index prices: hypothesis testing for the Korean stock markets. PLoS One **12**, e0188107 (2017). https://doi.org/10.1371/journal.pone.0188107
23. Qiu, M., Song, Y., Akagi, F.: Application of artificial neural network for the prediction of stock market returns: the case of the Japanese stock market. Chaos Solitons Fractals **85**, 1–7 (2016). https://doi.org/10.1016/j.chaos.2016.01.004
24. Sagir, A., Sathasivan, S.: The use of artificial neural network and multiple linear regressions for stock market forecasting. Matematika **33**, 1–10 (2017)
25. Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning?: a systematic literature review: 2005–2019. Appl. Soft Comput. **90**, 106181 (2020). https://doi.org/10.1016/j.asoc.2020.106181
26. Tkáč, M., Verner, R.: Artificial neural networks in business: two decades of research. Appl. Soft Computi. **38**, 788–804 (2016). https://doi.org/10.1016/j.asoc.2015.09.040
27. Wang, J., Zhang, Y., Yu, L.C., Zhang, X.: Contextual sentiment embeddings via bi-directional GRU language model. Knowl.-Based Syst. **235**, 107663 (2022). https://doi.org/10.1016/j.knosys.2021.107663