Bayesian Optimized Long-Short Term Memory Neural Network Framework in the Study of Stock Market Price Time Series Prediction

Sihang Zhao

Sendelta International Academ, Shenzhen, China

cosmozhao123@gmail.com

Keywords: Bayesian optimization, Lstm, Cnn, Stock market prediction

Abstract: Stock market values are erratic and difficult to forecast. We present a Bayesian Optimized Long-Short Term Memory (BO-LSTM) Neural Network in this study to anticipate stock market values. We discovered that standard LSTM fails to generate reliable prediction results because to problems in tuning in hyperparameter space, therefore we developed a technique that uses a Bayesian approach to automatically determine the best hyperparameters. BO-predicted LSTM's outputs have a relative error of 4% when compared to observed values, and it has significantly enhanced the LSTM approach in the field of time series prediction.

1. Introduction

Predicting the future is one of the most challenging prob- lems in data analysis. A time-series is a collection of data val- ues gathered throughout time at equally spaced intervals. It's a sequence that belongs to a certain domain. The technique of studying these sequences to find patterns and other statistical features is known as time series analysis. It may be thought of as employing interpolation or extrapolation while keeping prediction mistakes to a minimum (i.e., deviation between predicted and actual values) [1].

The objective is usually to create predictions about new or missing observations, with the goal of discovering a general pattern in previous observations that may be used to generate predictions about new ones. Statistical models are used to identify predictive connections between one set of data and another set of variables – frequently time series – for which we have no additional information.

The Auto-Regressive Integrated Moving-Average (ARIMA

) model was traditionally employed to perform the prediction job. The ARIMA model is a mathematical technique for predicting future values of a time series based on previous values and other assumptions about the series' statistical characteristics.

In other instances, however, ARIMA fails to forecast prop- erly. Predicting time series, for example, differs from predict- ing individual values (past and future) and necessitates the use of a predictive model and a learning algorithm. The crux of the issue is that ARIMA can't anticipate trend and seasonality from previous data unless these qualities are already apparent. In this situation, machine learning approaches for forecasting time series have shown to be more successful.

Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) are the two most used machine learning methods for predicting time series (CNN). These algorithms outperform ARIMA in terms of robustness. RNNs are based on the structure of biological neural networks, which are made up of hierarchically arranged components called neurons. RNNs are divided into layers, with each layer corresponding to a layer of neurons. Depending on the job at hand, these units are connected to one another in a variety of ways, including through directed and undirected connections. CNNs, on the other hand, have a structure of layers of neurons similar to RNNs. CNNs, on the other hand, lack directed connections, and the neurons are arranged in a series of convolutional and pooling layers. Auto-encoders, which aim to learn a com- pressed representation

  DOI: 10.25236/iccem.2021.061

from input data, have some similarities with them. Both RNN and CNN have had significant success in time series prediction jobs.

We use Long-Short Term Memory (LSTM), an extended RNN method, to forecast time seriesciteb2 in this study. However, the model's success is determined on how it analyzes sequences, and LSTM need precise hyperparameter changes in order to correctly grasp the sequence and produce accurate predictions. As a result, we use the Bayesian Optimization (BO) technique to improve the performance of the LSTM network. The BO is a Bayesian inference-based approach for optimizing model parameters. The BO's goal is to discover the best values for all parameters at the same time by estimating the posterior distribution across the parameter space. It may also be used to tune hyperparameters for neural networks, such as learning rate and number of layers, in machine learning. The objective is to identify a collection of hyperparameters that will result in the best overall performance or generalization ability. After implementing the BO-LSTM algorithm, we choose stock market to experiment on its capability in making ac- curate predictions. The effective prediction of financial time series data has long piqued the interest of the finance sector [3].The literature [4] introduced an approach based on ARIMA model to forecast price indices under the high volatility. The literature [5] provided a method to forecast non-stationary stock data by using AR model. To create a system for forecast- ing stock price, the literature [6] used a hybrid method com- bining Genetic Algorithm (GA) and Artificial Neural Network (ANN) methods [6].The literature [7] offered an application of ARIMA model to predict the future stock indices of the Indian economy.

We employed a database from Quantopian, a financial modeling and algorithmic trading platform, in this work. We used stock market data (referred to as "GSPC" or "The S&P 500"). Between 2011 and 2018, it covers daily closing and open prices of stocks.

2. Method Description

2.1 Arima Model

ARIMA is an acronym for: Auto-Regressive (AR): it uses estimations of historical dependent variable levels to forecast the dependent variable's future value. It is integrated because it calculates difference variables using data from previous periods. Moving Average (MA) smooths out short-term ran- dom fluctuations in auto-regressive models by averaging prior periods and thereby "smoothing" them out.

$$y_t = \alpha + \beta_1 y_{t-1} + \cdots + \beta_p y_{t-p} s_t + m_1 s_{t-1} + \cdots + m_q s_{t-q} \quad (1)$$

ARIMA model is a well-known time series prediction model [12].This is the equation of the ARIMA Model where the order of the AR and MA model is represented by p and

q, respectively; to make the input time series a stationary sequence, we need to differentiate the series *d* time(s).

$$y_t = a_1 y_{t-1} + s_t \quad (2)$$

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + s_t \quad (3)$$

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} + s_t \quad (4)$$

These are the functions of the first, second, and p order AR models, where *ap* denotes the auto-correlation coefficients, which must be greater than 0.5; *yt* denotes the expected value; and *epsilont* denotes a random error value with a mean of 0 and a standard deviation of *sigma*. Because every predicted value *yt* has a relationship with its prior value *yt p*, the AR model may not be accurate in forecasting time series that are heavily influenced by social variables.

$$y_t = m_1 s_{t-1} + s_t \quad (5)$$

$$y_t = m_1 s_{t-1} + m_2 s_{t-2} + s_t \quad (6)$$

$$y_t = m_1 s_{t-1} + m_2 s_{t-2} + \cdots + m_q s_{t-q} + s_t \quad (7)$$

These are the functions of the first, second and the q order MA model, where yt is the expected value ; st is assumed to be a random error value that has a mean equals to 0 and the standard deviation equals to σ; st−1, st−q2 st−q are the random error in previous time periods; mq are the MA coefficient to be estimated. The influence of rapid swings in demand on forecast results can be

smoothed out by using the MA model for forecasting. However, because it is an average value, the forecast value will always remain at the previous level and cannot be expected to produce future higher or lower fluctuations; the moving average approach necessitates the collection of a vast amount of historical data.

The ARIMA model is created when a time series is sepa- rated and merged using the AR and MA models [8]. When d is equal to 0, ARIMA model is the same as ARMA model, which implies the only difference that may differentiate ARIMA model from ARMA model is whether *d* is equal to 0 or whether the series is stable. The ARIMA model fits non- stationary time series, while the ARMA model fits stationary time series.

2.2 Cnn Model

CNN is a type of deep, feed-forward ANN that may be local or global in nature [9]. They are multi-layer perceptions that contain convolutional layers, which are special layers, and max-pooling layers, which are non-fully linked layers. CNN is well-known for its ability to cope with enormous volumes of data.

The weights of the inputs are shared in the convolution layer, and the field of the input is controlled by the units in the hidden layer. Convolution is essentially the sum of the element-by-element product. It is feasible to generate a smaller matrix containing essential characteristics by multiplying a specified section of the input with the filter.

[n + 2p − f + 1] × [n + 2p − f + 1] (8)

This is the equation of calculating the size of the convolution matrix where $n \times n$ is the input size and the filter size is represented by $f \times f$; p stands for the padding and the stride is represented by *s*.

The characteristics of the input sequence can be recognized by the filter in the CNN, regardless of where they appear, since they have been taught to do so. Dilated convolutions will be assumed when using CNN for predicting, and the output of the preceding layers will be skipped.

By dilating convolution, fewer connections and weights are required to create and train the model, allowing it to reach more distant values in the time series. Furthermore, causal padding is essential for maintaining the output dimension and allowing all filter weights to be applied to all inputs.

2.3 Lstm Model

Because it can deal with long-term dependence, LSTM is a branch of RNN [13]. The LSTM model comprises a unique set of memory cells [16], and the time step will be split down into smaller steps, with memories from earlier timesteps being utilized at each step. Each time step can have its own state, and at each timestep, a gate function is applied to that memory cell and a new input vector to determine whether or not the value of that memory cell should be changed. An input gate is followed by an output gate, which is followed by a forget gate, which is followed by another output gate, which is followed by an extra forget gate if necessary. At each time step, one must determine whether or not to update the memory cell's value. If this is not the case, one can ignore updating the memory cell's value; if it is, one must determine which values from the input vector to use to update the memory cell.

ft = σ(Wf ·[ht−1, xt] + bf) (9)

This is the equation of the forgetting gate,σ function is used to reach to the value of ft. In this formula, the forgetting rate is represented by the ft and σ is the Sigmoid activation function; the weight of f1 is represented by the Wf , ht−1 stands for the value of output of the previous time, and the offset of the

ft is represented by bf [14].

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (10)$$

the BO has completed evaluating the function, the prior will be changed in order to create a posterior distribution. In the following phase, the posterior distribution will be utilized to create an Acquisition function, which will be used to determine the hyperparameters. The model's

hyperparameters generate a target function that maximizes the objective output. The search space of the hyperparameters of the BO is shown in the Table I.

$Ct = tanh(W_c[h_{t-1}, x_t] + b_c)$ (11)

The information needed to be updated through the sigmoid layer will be defined by the input gate after the forgetting rate ft is determined. In the equation (9)(10), it is the forgetting rate and the weight and bias of each layer is represented by the Wi, Wc and bi, bc respectively. The activation function C is tanh, which is a fresh candidate value.

$C_t = f_t C_{t-1} + i_t C_t$ (12)

In the equation (11), where the unit state of the present hidden layer is represented by Ct, and the unit state of the hidden layer of the previous value is represented by the Ct−1.

During cell state experiencing the tanh function, a value

between [-1,1] will be generated and then multiply the output sigmoid layer and result in the final output.

$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ (13)

$h_t = o_t * tanh(C_t)$ (14)

In the equation (12)(13), o_t stands for the forgetting rate, the weight and bias of the output gate are represented by the W_o and b_o separately. Finally, the current state of the hidden layer is generated as h_t [15].

2.4 Bo-Lstm Model

BO is a method for determining suitable hyperparameters for machine learning algorithms in terms of a loss function. BO looks for a configuration space where our model will be most effective. This approach is based on Bayesian probability theory and employs Markovian assumptions throughout the optimization process, distinguishing it from other optimization techniques such as gradient descent and random search. The BO's Black-box function is seen below.

$x* = \arg\min f(x)$ (15)

$x \in X$

In the equation (14), f is the black object function which take a series of hyperparameters *x1, x2, x3, , xn* where n is the ultimate number of the hyperparameters as input and return in $x_1^*, x_2^*, x_3^*, \ldots, x_n^*$.

A data set made up of multiple arrays is represented by *H*

in the implementation. (X, y) denotes each and every pair of arrays in which X denotes a set of hyperparameters and y

denotes the result that is similar to X. A is the Acquisition function used to determine x in equation (14) and M denotes the kind of model utilized to fit the data series H. BO often use a Gaussian model.

The technique employed by BO is to treat the target function as a random function and apply a prior to it. When

Table 1 Search Space Of Hyperparameters

Hyperparameter	Search space
Number of LSTM cells	4-512
Activation function	ReLU, Linear, Sigmoid, Tanh, ELU
Optimization method	SGD, Adam, Nadam, Adamax, Adadelta, Adagrad, RMPSprop
Neurons in hidden layer	4-512
Dropout rate	0-0.8
Batch size	4-256
Epochs	5-100

3. Data Description

3.1 Data Categorization

The data set used to evaluate the previously mentioned time series prediction techniques is described in this section. From 2011 to 2018, we collect statistics for SP500 points. The stock market point for the SP500 is updated on a daily basis, and these figures are very volatile for a variety of reasons. Then, from 2013 to 2017, we gather the price per share for a number of well-known firms, including Amazon, Apple, Rio Tinto, and FedEx. We chose Apple and Rio Tinto as our test cases because Apple's stock price has been rising in recent years, while Rio Tinto's stock price has been declining. We can determine from the prediction results if these approaches can be used to practical circumstances and if BO-LSTM can beat the existing mechanisms because the stock prices for these businesses have showed distinct tendencies throughout the years.

3.2 Data Preprocessing

Normalization, standardization and regularization are com- monly used data preprocessing methods. Since the original data in this research belong to the same feature (load), standardization is a more appropriate data preprocessing method to accelerate convergence. The input standardization is conducted using equation:

$$X_{new} - \frac{x-\mu}{\sigma} \tag{16}$$

Where μ denotes input data mean, σ denotes input data standard deviation, x and xnew denote data actual value and standardized value [10]. Therefore, the data needs to be sequenced to the size of a 3-dimensional array to feed into the hidden layer, and the data set is divided into training and test sets using a 10 k cross-validation method to avoid over-fitting.

3.3 Performance Evaluation Indice

In this study, mean absolute percentage error (MAPE), root mean squared error (RMSE) and Correlation coefficient (R) are adopted to evaluate the performance of predictions [11].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{17}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} * 100\% \tag{18}$$

$$R = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{19}$$

Where n is the entire number of observations, yˆ represents the forecast value, and y represents the actual value of the tested data [10].

4. Experiments and Results

The data for testing forecasting models may be found in this area, which includes the SP-500 stock market point, Apple value per share, and Rio Tinto value per share. The experiment is divided into three instances in this section.

Case 1: SP-500 point: we set the final date to be February 2018, which is the last entry of the acquired data set. Then, we collect the market value ranging from the last entry to its previous 2500, 1000, and 500 data points, and divided these data into 90% training set and 10% test set.

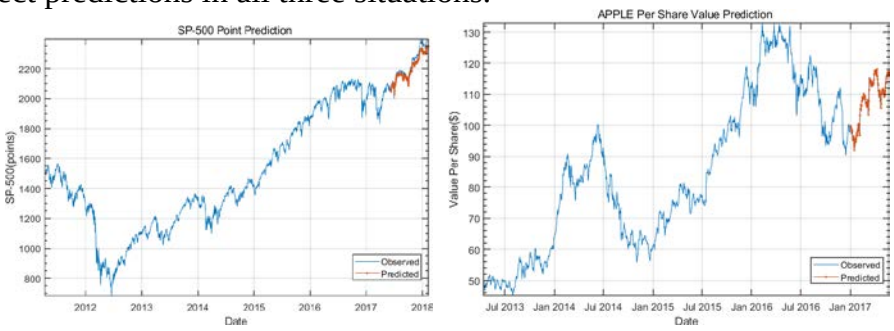
Case 2: Apple value per share: Similar to case 1, we use 1500, 1000 and 500 data points and the same strategy for training and test set division (90% training and 10% test).

Case 3: Rio Tinto value per share: Similar to case 1, we use 1500, 1000 and 500 data points and the same strategy for training and test set division (90% training and 10% test).

The efficacy of BO-LSTM is demonstrated in this section. The suggested model's performance is assessed and compared to that of various current prediction models such as ARIMA, LSTM, and CNN [10]. All experiments are performed in MATLAB 2021a.

4.1 Effectiveness of Bo-Lstm

The longest forecast results from all three examples are presented in this section, along with the findings. We have 2250 data points for model training and 250 data points for testing in Case 1. We have 1350 data points for model training and 150 data points for testing in instance 2. We have 1350 data points for model training and 150 data points for testing in instance 3. After effective tuning of hyperparameters of the LSTM neural network, the BO-LSTM is able to generate correct predictions in all three situations.



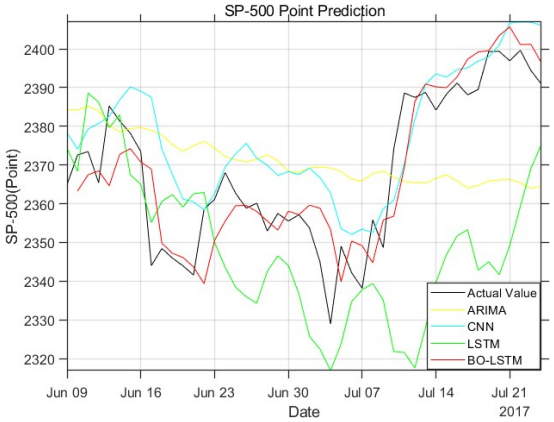
(a)Apple

(b)Rio

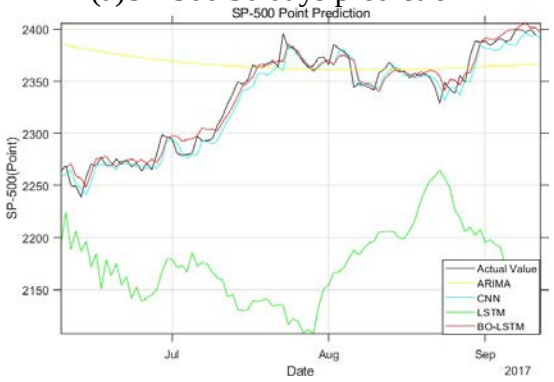
Fig.1 Comparison between Observed and Predicted Value for the Three Cases.

4.2 Model Comparison

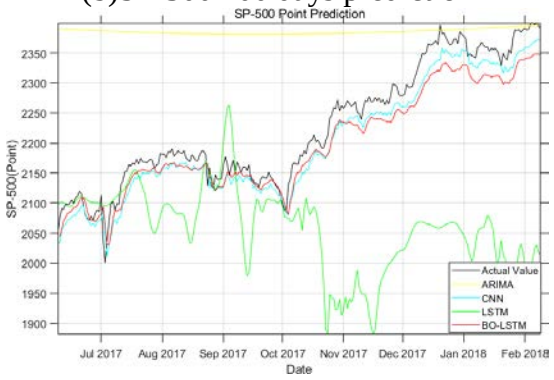
We go over the differences between these instances in great depth. In this case study, we collect various amounts of data points for short-term, medium-term, and long-term forecasts. We've run several tests on conventional CNN and LSTM to determine the best hyperparameters for the most accurate prediction. LSTM is difficult to adjust in general, and the results vary substantially (this is also the main motivation of developing BO-LSTM). After including the Bayesian Op- timization method, BO-predictive LSTM's performances are equivalent to, if not better than, CNN's.



(a)SP-500 50 days prediction



(b)SP-500 100 days prediction



(c)SP-500 250 days prediction

Fig.2 Sp-500 Point Prediction Using Different Forecasting Methods.

From the experiments and figures shown, we can also reach the following conclusions for the different prediction methods other than BO-LSTM.

When the data is not stationary, ARIMA fails to generate meaningful findings, and it only finds linear relation- ships in the training set. We experimented with several ARIMA hyperparameters (p,d,q) and found that the results were unsatisfactory. It can be utilized for short- term prediction in specific instances, but it is not ideal for predicting stock market information in general.

The hyperparameters utilized in LSTM, such as the initial learning rate, drop rate, number of cells, and so

Table 2 Sp-500 Stock Market Point Prediction Evaluation Statistics

SP500		ARIMA	LSTM	CNN	BO-LSTM
50 days	R	-0.15661	0.28444	0.83002	0.86939
	MAPE	8.30%	6.74%	0.49%	0.29%
	RMSE	2372.757	28.9465	14.368	0.86939
100 days	R	-0.8057	-0.10089	0.98017	0.97523

	MAPE	20.34%	6.60%	0.32%	0.62%
	RMSE	2367.388	166.8857	10.4018	10.997
250 days	R	0.53888	0.49077	0.98969	0.98117
	MAPE	7.59%	7.07%	0.58%	2.73%
	RMSE	2385.694	219.8035	9.6757	16.8485

on, are highly sensitive. It is hard to know how to change these hyperparameters while utilizing the stock market data set, and the predictive results vary greatly. We show that LSTM can be utilized to predict short-term values in most instances, but the cumulative effect of LSTM mistakes renders it incapable of medium- and long-term forecasting.

Although CNN is often employed in image processing, we leverage its unique convolutional kernel structure to understand the data's potential information. In this work, we discovered that in time series prediction, CNN beats classical RNN.

Our suggested model's performance is compared to that of other well-known classic models including ARIMA, LSTM, and CNN using R, MAPE, and RMSE. We can see from the tables that ARIMA and LSTM fail to perform well in projecting the stock market point and the value per share of firms. The accuracy (MAPE, RMSE) and correlation (R) of the CNN and BO-LSTM are comparable, and we can see that our proposal outperforms both in these statistical assessments. Overall, the case study findings show that the data prediction method's accuracy has considerably increased since the automated hyperparameter modification mechanism was implemented.

5. Conclusion

The BO-LSTM neural network model is demonstrated to be effective in long-term prediction in this study. The findings of the case study indicate consistent performance as well as the ability to detect unexpected data deviation patterns. Furthermore, when compared to other contemporary models in terms of MAPE, RMSE, and R at various time intervals, BO-LSTM outperforms them, proving its superiority. Rather of relying on the unpredictability and uncertainty of manual tuning, Bayesian optimization aids in the optimization of hyperparameters. Adaptive optimization procedures in the tuning of machine learning hyperparameters is a promising future development route in this respect.

Future efforts will be focused on developing hybrid models by combining various neural networks to improve prediction accuracy; adding feature selection approaches into the models; and developing multi-task learning to cope with multi- dimensional, linked data.

Table 3 Apple Value Per Share Prediction Evaluation Statistics

Apple		ARIMA	LSTM	CNN	BO-LSTM
50 days	R	-0.20074	-0.35603	0.92883	0.92662
	MAPE	3.87%	11.17%	0.85%	0.40%
	RMSE	109.5252	13.4461	1.2568	0.4198
100 days	R	0.46858	0.16456	0.93991	0.93816
	MAPE	16.92%	4.30%	0.82%	0.97%
	RMSE	130.8601	7.1823	1.2586	0.3816
150 days	R	0.84885	0.5839	0.98385	0.98341
	MAPE	12.77%	3.93%	0.88%	0.58%
	RMSE	121.0103	8.6278	1.3033	0.90088

Table 4 Rio Tinto Value Per Share Prediction Evaluation Statistics

Rio Tinto		ARIMA	LSTM	CNN	BO-LSTM
50 days	R	-0.76295	0.29143	0.94879	0.91923
	MAPE	17.68%	15.50%	1.70%	0.30%

	RMSE	30.866	6.4336	0.83106	0.399
100 days	R	0.76295	0.29143	0.94879	0.91923
	MAPE	17.68%	15.50%	1.70%	1.90%
	RMSE	30.866	6.4336	0.83106	1.399
150 days	R	-0.83176	0.23312	0.97716	0.97544
	MAPE	14.18%	17.21%	2.50%	2.48%
	RMSE	33.2447	6.9947	1.0169	0.95442

References

[1] K. A. Althelaya, S. A. Mohammed and E. -S. M. El-Alfy, "Com- bining Deep Learning and Multiresolution Analysis for Stock Market Forecasting," in IEEE Access, vol. 9, pp. 13099-13111, 2021, doi: 10.1109/ACCESS.2021.3051872.

[2] Zhang, Mingfang & Li, Huajian & Wang, Li & Wang, Pangwei & Tian, Shun & Feng, Yue. (2020). Overtaking Behavior Prediction of Rear Vehicle via LSTM Model. 3575-3586. 10.1061/9780784482933.308.

[3] Sezer, Omer & Gudelek, Ugur & Ozbayoglu, Murat. (2020). Fi- nancial time series forecasting with deep learning : A systematic literature review: 2005–2019. Applied Soft Computing. 90. 106181. 10.1016/j.asoc.2020.106181.

[4] R. M. Kapila Tharanga Rathnayaka, D. M. K. N. Seneviratna, W. Jianguo and H. I. Arumawadu, "A hybrid statistical approach for stock market forecasting based on Artificial Neural Network and ARIMA time series models," 2015 International Conference on Behavioral, Economic and Socio-cultural Computing (BESC), 2015, pp. 54-60, doi: 10.1109/BESC.2015.7365958.

[5] Riswan Efendi, Nureize Arbaiy, Mustafa Mat Deris, A new procedure in stock market forecasting based on fuzzy random auto-regression time series model,Information Sciences,Volume 441,2018,Pages 113- 132,ISSN 0020-0255, https://doi.org/10.1016/j.ins.2018.02.016.

[6] Ebadati E., Omid Mahdi & Mortazavi T., Mohammad. (2018). An effi- cient hybrid machine learning method for time series stock market fore- casting. Neural Network World. 28. 41-55. 10.14311/NNW.2018.28.003.

[7] Banerjee, Debadrita. (2014). Forecasting of Indian stock market us- ing time-series ARIMA model. 2014 2nd International Conference on Business and Information Management, ICBIM 2014. 131-135. 10.1109/ICBIM.2014.6970973.

[8] S. Mehrmolaei and M. R. Keyvanpour, "Time series forecasting using improved ARIMA," 2016 Artificial Intelligence and Robotics (IRA- NOPEN), 2016, pp. 92-97, doi: 10.1109/RIOS.2016.7529496.

[9] Ladda, Matthew & Champagne, Trevor. (2019). Utilizing convolutional neural networks to detect skin cancer: a review of initial trials. University of Toronto medical journal. 96. 22 - 24.

[10] Munem, Mohammad & Bashar, T. & Roni, Mehedi & Shahriar, Munem & Shawkat, Tasnim & Rahaman, Habibur. (2020). Electric Power Load Forecasting Based on Multivariate LSTM Neural Network Using Bayesian Optimization. 1-6. 10.1109/EPEC48502.2020.9320123.

[11] Oliveira, Fernanda & Carvalho, Luciene & Teixeira, Leonardo & Fontes, Cristiano & Lima, Kassio & Caˆmara, Anne & Medeiros, Heloise & Sales, Rafael. (2017). Predicting cetane index, flash point and content sulfur of diesel-biodiesel blend using an artificial neural network (ANN) model. Energy & Fuels. 31. 10.1021/acs.energyfuels.7b00282.

[12] Kumar, Raghavendra & Kumar, Pardeep & Kumar, Yugal. (2021). Multi- step time series analysis and forecasting strategy using ARIMA and evo- lutionary algorithms. International Journal of Information Technology. 10.1007/s41870-021-00741-8.

[13] Fan, Dongyan & Sun, Hai & Yao, Jun & Zhang, Kai & Yan, Xia & Zhixue, Sun. (2020). Well production forecasting based on ARIMA- LSTM model considering manual operations. Energy. 220. 119708. 10.1016/j.energy.2020.119708.

[14] Liwei, Tian & Li, Feng & Yu, Sun & Yuankai, Guo. (2021). Fore- cast of LSTM-XGBoost in Stock Price Based on Bayesian Opti- mization. Intelligent Automation & Soft Computing. 29. 855-868. 10.32604/iasc.2021.016805.

[15] Li, Hui & Hua, Jinjin & Li, Jinqiu & Li, Geng. (2020). Stock Forecasting Model FS-LSTM Based on the 5G Internet of Things. Wireless Com- munications and Mobile Computing. 2020. 1-7. 10.1155/2020/7681209.

[16] Qiu, Jiayu & Wang, Bin & Zhou, Changjun. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLOS ONE. 15. e0227222. 10.1371/journal.pone.0227222