Stock Price Prediction Using LSTM and Search Economics Optimization

Abba Suganda Girsang, Fernando Lioexander, and Daniel Tanjung

***Abstract*—Long Short-Term Memory (LSTM) has been one of the most popular methods in time-series forecasting. Within the LSTM architecture, there are hyperparameters present that need to be optimized in order to achieve the optimum results. In order to optimize these hyperparameters, a metaheuristic optimization method was used. A metaheuristic algorithm was used as a way to reduce both time and computational complexity. We used the Search Economics algorithm because we found the algorithm quite interesting while being unpopular. The evaluation was carried out by using Root Mean Square Error (RMSE) as the primary metric used for the optimization. The dataset being used in this experiment is the stock price of one of the most well-known financial institutes in Indonesia. The SE-Optimized LSTM was able to create a prediction that did not overfit with RMSE of 538.92, which was great compared to the unoptimized LSTM model with RMSE of 661.041 and ARIMA model with RMSE of 2809.015.**

***Index Terms*—forecasting, long short-term memory, metaheuristic optimization, time-series, search economics**

I. Introduction

STOCK market price prediction has attracted many eyes in the last couple of decades. However, for most people, stock trading could not give them a reliable source of income. For many people, stock trading is much like gambling, especially for those unfamiliar with the way it works. They could only try to guess where the price would go for the future. Even for those expert traders, they would not dare to say that they could always make a profit. It is because stocks are unpredictable and have high volatility in nature [1]. Many factors could affect the future price of a stock, both from outside and inside. Outside factors that could potentially affect the stock price are politics, trade wars, and worldwide events. On the other hand, inside factors are factors that directly affect stockholders and traders, such as the difference in disposition [2], character [3], IQ [4], and emotional state [5], [6], and [7].

Because of the unpredictability trait of the stock price, both humans and computers could not reliably predict future prices of the stock itself. In the computer science field, there are methods to predict time-series, which theoretically includes stock price prediction. However, most classical prediction models could only predict simple time-series models, not random-walk time-series models like a stock price. Deep learning becomes a viable option to try to make a more reliable model of prediction for harder time-series cases. Among them, Long Short-Term Memory (LSTM) has been given much love from people working in this field. The reason for that is because LSTM will often give better results compared to other popular time-series prediction methods such as Multi-Layer Perceptron (MLP) [8], Support Vector Regression (SVR) [9], Bayesian-Optimized RNN [10], and Auto-Regressive Integrated Moving Average (ARIMA) in [10], [11], and [12].

However, LSTM is not an all-powerful method. One thing that can drastically change the output from an LSTM model is its hyperparameters. Hyperparameters are values that need to be assigned manually before the model enters the learning state. As these hyperparameters need to be set manually, it is a job of data scientists to assign them their correct values. Be that as it may, assigning values to these parameters is not as easy as it sounds. Data scientists could have a hard time assigning values for these parameters as some of them are parameters of the LSTM model with no relation to the data itself. While data scientists can roughly estimate parameters related to the data, it would be hard for them to estimate the parameters of the LSTM model.

The most straightforward way to tune these hyperparameters is to test them one by one and compare the results manually. However, manually tuning each parameter would take way too much time as just training one LSTM model would need substantial resources and take quite a long time, not to mention to train many of them with each one having different parameters. Because of that, heuristic optimizations are often used in this kind of situation. In this experiment, we have decided to use a heuristic optimization algorithm called Search Economics.

Search Economics is proposed as a new kind of heuristic algorithm to minimize the time it takes for an optimization algorithm to search for a better answer. The concept of investing in areas with potential allows this algorithm to minimize redundancy of searching in the same repeatedly. Using a standard optimization problem, the Search Economics was proven to give better results than the Genetic Algorithm as recorded in [13], where this optimization algorithm was used to optimize the deployment of wireless sensor networks.

II. Literature Review

A. Related Works

Researchers had tried to implement many different methods for time-series forecasting in order to improve the reliability of time-series forecasting. Research about

Manuscript received February 27, 2020; revised June 20, 2020.

Abba Suganda Girsang is with the Computer Science Department, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480 (e-mail: agirsang@binus.edu).

Fernando Lioexander is with the Computer Science Department, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480 (e-mail: fernando.lioexander@binus.ac.id).

Daniel Tanjung is with the Computer Science Department, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480 (e-mail: daniel.tanjung@binus.ac.id).

predicting the Shanghai Composite Index was executed using improved C-Fuzzy Decision Tree (CFDT) with the use of a stop condition to reduce time complexity [14]. A comparison between different methods of time-series forecasting such as MLP, Artificial Neural Network (ANN), and hybrid ANN which used Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) was carried out in [15] with MLP came out as the winner and hybrid ANN as the worst of the three. Various structures of MLP combined with the use of different training and transfer functions were carried out in [16], resulting in a tremendous R^2 score of 0.9622.

In predicting the movement of stock prices in the next 15 minutes, the comparison between LSTM-based models with the baseline models (Random Forest, MLP, Pseudo-random) was evaluated. LSTM can exceed the accuracy of the baseline method and statistically have a p-value less than 0.05 which shows a statistically significant difference between the data populations [17]. In the stock prediction for a span of 10 years from 2007 to 2017, the RNN method tends to be more volatile in fitting the curve model especially at the turning point than the LSTM. Also the LSTM degree of deviation is slightly smaller than the RNN showing how well the LSTM model adjusts the data [18]. Genetic Algorithm was used in [19] to optimize weights and biases of a Hybrid Artificial Neural Network model with a result of a hit ratio of 81.27%.

Another popular method in time-series forecasting is ARIMA. Auto-Regressive Integrated Moving Average (ARIMA) is popular in the field of time-series forecasting because of its ability to give better and more reliable results than most known methods. Both ARIMA and ANN are often compared to one another, and many comparative study cases are comparing the two with mixed results and no clear winner between the two. A hybrid ARIMA model combined with ANN is introduced in [20], which proved that the hybrid model gives better results than its parents. Extensive research using the ARIMA model for predicting wind speed used for generating power is performed in [21].

TABLE I
SUMMARY OF RELATED WORKS

Ref.	Dataset	Method	Result
[14]	Stock Price	CFDT Weighted CFDT	Weighted CFDT gives better results
[15]	Stock Price	MLP Dynamic ANN GARCH-ANN	MLP bested other competitors
[16]	Stock Price	MLP	High R^2 Score of 0.9622
[17]	Stock Price	MLP LSTM Pseudo-random Random Forest	LSTM exceeds the average accuracy on various datasets and gives a positive ratio return
[18]	Stock Price	LSTM RNN	compared to the RNN model, the LSTM model has the advantage of adjusting data better
[19]	Stock Price	ANN GA-ANN	GA-ANN gives a better result of a hit ratio of 81.27%

Other than ANN and ARIMA, there is one popular deep learning method for time-series forecasting, which is LSTM. As a part of deep learning, LSTM provides better flexibility in terms of forecasting. Not only flexible, but LSTM could also perform very well when carefully optimized. In the case of predicting solar irradiance, LSTM performed much better compared to Back-Propagation Neural Networks (BPNN), achieving a 42.9% reduction of Root Mean Square Error (RMSE) compared to the latter [8]. A comparative study case comparing LSTM and Support Vector Regression (SVR) was carried out in [9], and LSTM outperformed SVR. Many people made comparisons between LSTM and ARIMA, like in [10], [11], and [12], with most of them won by LSTM.

B. Time-Series

Time-series is a time-ordered list of data points. Based on the number of observed variables in a time-series, time-series could be divided into two types, univariate and multivariate. Univariate means that there is only one observed variable, while multivariate means there is more than one observed variable in the time-series. No matter whether it is a univariate or multivariate time-series, each variable will usually be presented in numerical values.

Looking at the spacing intervals between each data point in time-series, we could divide them into two types, continuous time-series or discrete time-series. A continuous time-series is obtained by sampling at the smallest metrics of time possible; in most cases, every one second. On the other hand, the sampling of a discrete time-series is taken at intervals larger than one second (e.g., every minute, every hour, or even every year). However, a time-series may be sampled at irregular intervals, meaning that the interval between two data points compared to another interval of another pair within the same time-series may not be the same (e.g., a sample is taken whenever there is a transaction being made).

In order to analyze time-series, usually, decomposition takes place. Time-series decomposition could allow for more straightforward analysis, especially for those time-series which have seasonality. In most cases, a decomposition of a time-series will output three components; they are trend, seasonality, and remainder. An example of a time-series decomposition could be seen in Figure 1.

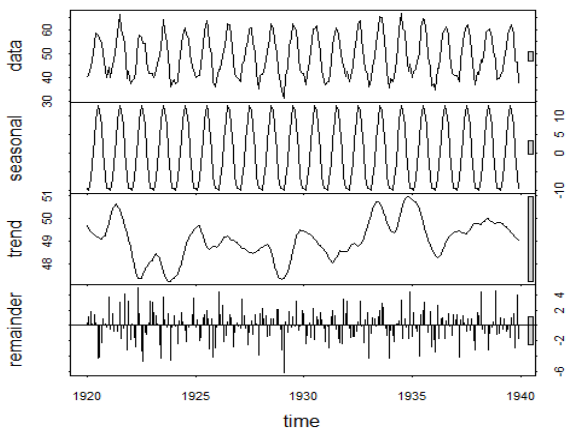

Fig. 1. An example of decomposition of time-series.

The trend in time-series describes whether the observed variables are increasing or decreasing in their direction. A trend within time-series is not constrained to be linear. As a matter of fact, most cases in real-life will have a non-linear trend. The trend is probably the most crucial part of a time-series analysis as by analyzing the trend, we could get valuable information regarding the observed variables such as the necessary information to predict the trend going forward or detect anomalies within the data.

Aside from trends, the most noticeable component in time-series is seasonality. However, not all time-series have seasonality. Seasonality in time-series refers to the repetitive pattern found on time-series within a fixed period (e.g., weekly, monthly, or yearly). Seasonality in time-series is a byproduct of a series of real-life events affecting the observed variables. An example of a yearly seasonality would be the sales of chocolate drastically rise every Valentine's Day as people would tend to buy them to gift it to their significant other.

Lastly, there is an irregular component of a time-series, which sometimes is called in different names such as remainder, residual, or random is simply a component of noise obtained from the decomposition of the time-series itself. This noise is obtained after removing all the other components from the real data.

C. Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is practically a Recurrent Neural Network (RNN) with the main concept of gates. There are three main gates in LSTM architecture, they are input gate, output gate, and forget gate. These three gates are responsible for controlling the flow of information over time within the central cell. In theory, RNN could store information from past inputs. However, in practice, RNN could only remember past information in a limited interval and could not remember information from the distant past. This problem of traditional RNN is further researched in [22] and [23], and where it is found that the existence of decaying errors is the cause of the problem.

As the number of layers in the RNN increases, the back-propagated error keeps getting smaller and smaller until it kind of vanish (*vanishing gradient*), or it keeps getting bigger and bigger until it explodes (*exploding gradient*). This problem dramatically affects the learning efficiency of the neural network. Because of this persistent problem in traditional RNN architecture, in the year of 1997, LSTM was introduced as the solution to tackle the problem mentioned above [24].

D. Search Economic

Search Economics is a newly proposed metaheuristic algorithm introduced in late 2015. Metaheuristic algorithms tremendously help in optimizing time-consuming and computational-demanding projects. The term heuristic means that the algorithm can optimize the problem faster than manual optimization even though it may not give the most optimal solution. Most metaheuristic optimization uses random selection without real knowledge of the surrounding potential. On the other hand, Search Economics offers a different approach to the selection phase.

Search Economics calculates all potential solution space before investing further into it. It also calculates the number of searchers that had been in the same solution space. It means that even though two or more solution spaces have the same objective value, each solution space may have different potential based on how many times searchers have visited it. This kind of potential is implemented so that there will be less redundancy while searching for a solution.

There are three main functions in this algorithm, which are:

1) *Resource Arrangement* (RA), which distributed every searcher into different regions to randomly invest in candidate solutions.
2) *Vision Search* (VS) is where every searcher will work together in finding the best solution.
3) *Marketing Research* (MR) is where information about each region's fitness value is saved.

As in [13], the algorithm of Search Economics can be described as follows:

Resource Arrangement

1) Divide solution space into h regions.
2) Initialize w random candidate solutions for each region, r_J.
3) Find the best solution in each region, ${r^b}_j$.
4) Assign every searcher into different regions (depending on the number of searchers and the number of regions, each region may have one or more searcher(s) assigned into it or none at all.)

Vision Search

5) Transit (crossover and mutation between searcher investment with candidate solution, ${v^i}_{jk}$.
6) Measure investment for each region, T_j, in order to reduce redundancy of search.

$$T_j = \frac{{t^b}_j}{{t^a}_j} \tag{1}$$

7) Measure the potential of the i-th searcher investing in the j-th region, ${v^i}_j$, based on the candidate solutions of the region, ${v^i}_{jk}$.

$${V^i}_j = \frac{\sum_{k=1}^{w} f({v^i}_{jk})}{w} \tag{2}$$

8) Calculate the weight of each region's best solution, M_j, using (3).

$$M_j = \frac{f({r^b}_j)}{\sum_{j=1}^{h} f(m_j)} \tag{3}$$

9) Evaluate approximately every solution's quality or potential using (4).

$${e^i}_j = T_j {V^i}_j M_j \tag{4}$$

10) The determination operator of the VS will randomly choose some solutions from other regions, v_{ij}, where $i \neq j$ to add into the temporary solutions so that the i-th searcher will less likely to get stuck on global optimum.
11) i-th searcher chooses the best solution out of all temporary solutions provided.

Marketing Research

12) Save the history of each regions' solutions.
13) Update every region's potential.

III. Proposed Method

There are five hyperparameters that we optimized in this research, namely, *sliding window*, *dropout*, *LSTM units*, *batch size*, and *column*. For each one of them, we assigned a set of values. From these sets of values, Search Economics tried to find the best combination of hyperparameters by trying random combinations. Search Economics used the fitness value of each combination to compare which one is better (higher means better). The calculation for the fitness value used in this experiment is directly impacted by the RMSE of the model as could be seen in (5)

$$fitness = \frac{1}{RMSE} \tag{5}$$

The best-so-far fitness value was updated every iteration whenever there was a better fitness value, and the better solution was saved. The iteration was stopped when it reached the maximum iteration constraint set at the beginning. Other than the maximum iteration constraint, other stop conditions may also be used (e.g., the convergence of the fitness value). A simple illustration of how the proposed SE-LSTM works could be seen in Figure 2

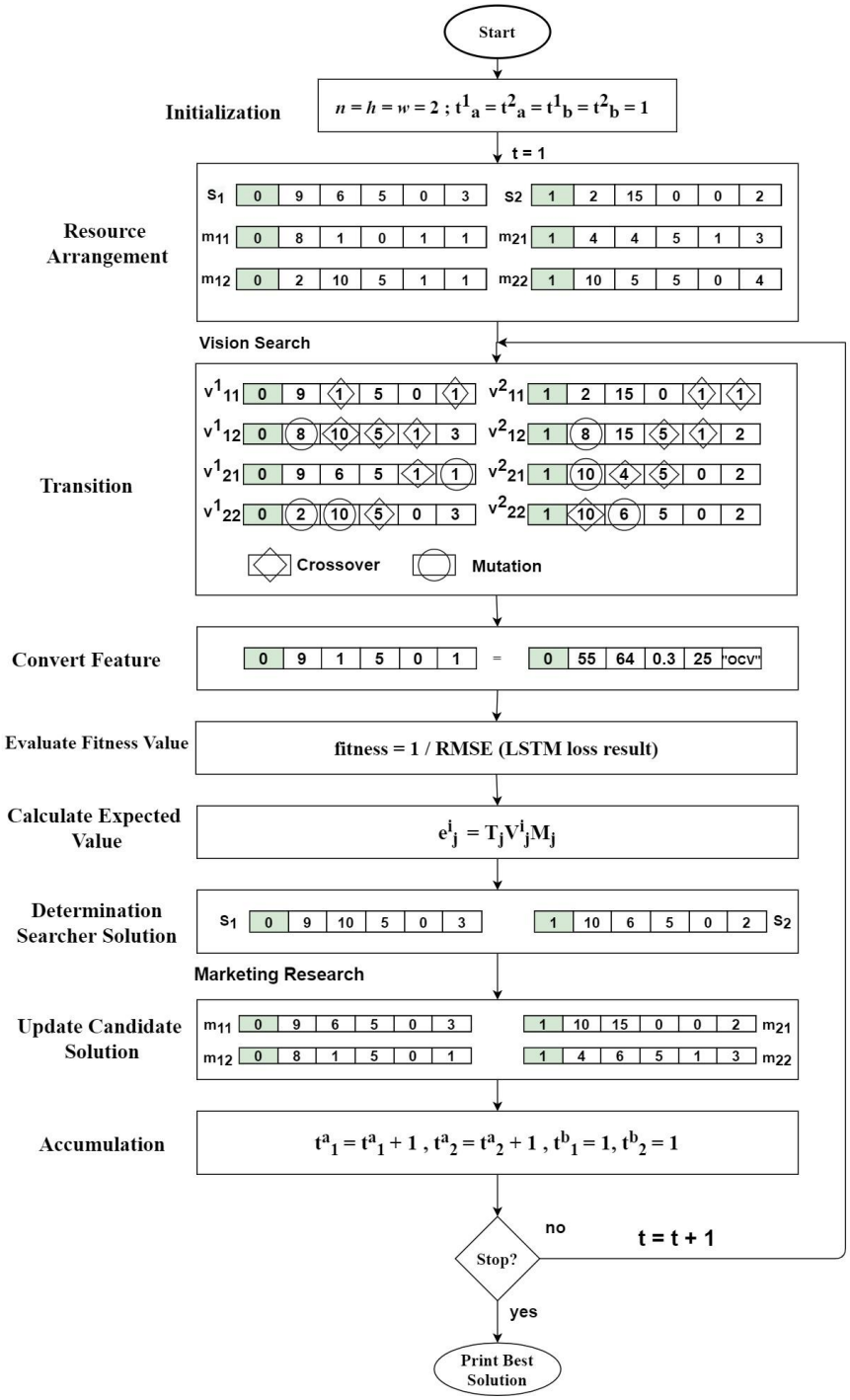

Fig. 2. Simple SE-LSTM illustration.

IV. Experimental Results

A. Dataset

In this case, we used a dataset of the stock price of one of the most well-known financial institutes in Indonesia. The dataset contains daily records of the stock price from the year 2006 to 2019, except for weekends and holidays. We used closing price, opening price, highest price, lowest price, and volume of transactions of the day as observed variables. Some of the combinations of these five columns were included for optimization, so we could know what variables are great for this case of stock price prediction and what variables we should avoid. Table II describes overall dataset values ranging from January 2006 to December 2019.

TABLE II
Dataset Description

Features	Min	Max	Mean	Standard Deviation
Open	1700	25100	9191.513	5935.624
High	1700	25475	9285.968	5976.618
Low	1675	24675	9092.838	5896.598
Close	1700	25075	9193.526	5936.223
Volume	0	2.171140e+08	1.862130e+07	1.793164e+07

B. Data Scaling

After the data successfully imported, we scaled them using the MinMaxScaler function with a range of 0 to 1. The function shrunk the range of value of each column based on their lowest and highest values. After getting both the lowest and the highest value, it scaled the rest of the data in the same column appropriately. The process was done for every column we used.

C. Data Splitting

Scaled data then split into three sets, training set, validation set, and test set (holdout set). The proportion we used in splitting the dataset into training, validation, and test sets is 80%, 10%, and 10%, respectively. We used training data to train the LSTM model and evaluate the model by using the validation set while keeping the test set quarantined. The horizon was calculated by multiplying the Holdout value (Ho) by the total of rows of the dataset. With the holdout value of 0.1 and the total of rows of the dataset being 3473, the horizon of the model was 347 days.

D. Hyperparameters

For this experiment, we set the epoch of the model as 100, while the other five hyperparameters were optimized in the later stage. Sets of values for all hyperparameters optimized in this experiment could be seen in Table III.

TABLE III
List of Optimized Hyperparameters

Hyperparameter	Range Values	Explanation
Sliding Window	10 to 60	interval of 5
Dropout	0.05 to 0.50	interval of 0.05
LSTM Units	25 to 100	interval of 5
Batch Size	32, 64, 128	-
Columns	[Open, Close, High, Low, Volume]	every possible combination

E. Environment and Parameter Setting

Because of the high time-complexity and resource-demand of LSTM models, we used the Compute Unified Device Architecture (CUDA) platform offered by Nvidia to assist us in processing the models by using a CUDA-enabled GPU made by Nvidia.

The configuration for the Search Economics algorithm itself could be seen in Table IV.

TABLE IV
SE Parameters Settings

Parameter	Setting
Dimension	5
Number of Searcher	4
Number of Region	4
Number of Samples	4

F. Evaluation

For the evaluation, we used a few performance metrics, such as Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and R^2 Score. However, we mainly focused on minimizing the RMSE of the model. RMSE is a metric used in measuring the spread and concentration of the predicted data from the actual data. MAE measures the average deviation of the predicted data from the actual data. MAPE is just like MAE in implementation, aside from the fact that MAPE uses the percentage of deviation instead of the value of the deviation itself. On the other hand, the R^2 Score is very much different from error calculation metrics like the RMSE, MAE, and MAPE.

In error calculation metrics, the value ranges from 0 to $+\infty$, with a lower value being better than a higher value. However, R^2 Score value ranges from $-\infty$ to 1, and the closer it is to 1, the better it is. R^2 Score tells how well a model fits the actual data. An R^2 Score of 1 means that there is no error or deviation, and the model fits the actual data perfectly. An R^2 Score of 0 means that the model fits the actual data the same as a horizontal straight line fits the actual data, while an R^2 Score of negative means that the model fits worse than a horizontal straight line.

A comparison was made with two other methods using the same training and testing datasets. An LSTM model was built without the use of any optimization method. The other method being used, ARIMA, was built with the help of auto_arima function which decided the values for *p*, *d*, and *q*. The auto_arima function decided to use the configuration ARIMA(1,1,1), meaning that $p = 1$, $d = 1$, and $q = 1$. However, the ARIMA model could only be univariate, compared to the other two which was multivariate. This restriction was the core of the method itself, so it is not something that can be modified easily.

G. Experiment Result

The point of convergence in optimizing this LSTM model by using Search Economics was achieved at the 45th iteration, as could be seen in Fig. 5. There was no further improvement attained from the 46th iteration to the 100th iteration, thus ending the optimization process.

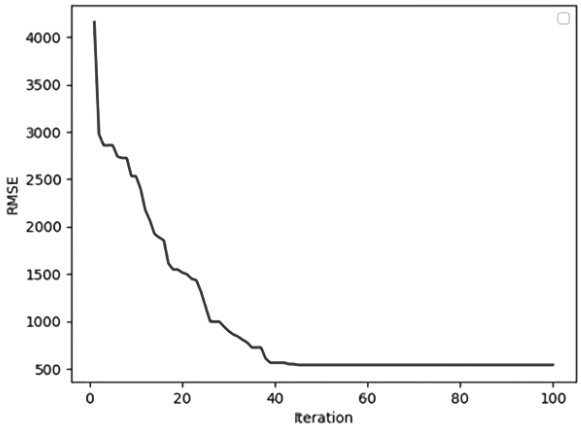

Fig. 3. The convergence of the SE-LSTM model.

By the end of the optimization, an output file containing the best configuration of hyperparameters was produced along with two other files. One of them describes a list of the best-so-far RMSE in each iteration of optimization. This allows the draw of Figure 3. Meanwhile, the other one describes a list of the predicted values performed on the holdout set which enables us to see how well the prediction produced compared to the actual data itself drawn in a graph as can be seen in Figure 4.

The final configuration of the hyperparameters obtained from the optimization process could be seen in Table V.

TABLE V
BEST HYPERPARAMETERS CONFIGURATION

Hyperparameter	Value
Sliding Window	60
Dropout	0.25
LSTM Units	50
Batch Size	32
Columns	['Close', 'Volume']

Using the configuration given in Table V, the prediction was performed on the quarantined holdout set with a Ho value of 0.1 and a horizon of 347. The performance of the model yield excellent results with no overfitting problem, as can be seen in Figure 4.

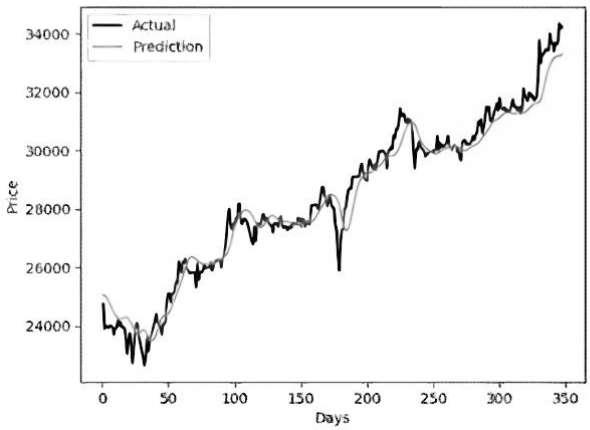

Fig. 4. The prediction result of the best SE-LSTM model.

The SE-LSTM model performed better than both the LSTM model with no optimization involved and the auto ARIMA model. The non-optimized LSTM was modeled manually with the experience the researchers have. The model produced good enough results close to the SE-optimized LSTM, the graph can be seen in Figure 5.

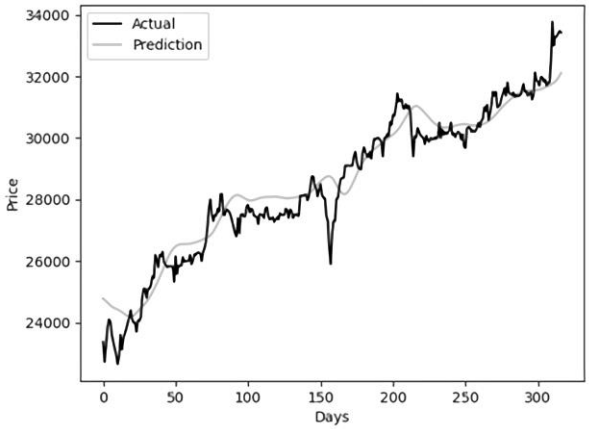

Fig. 5. The prediction result of the LSTM model.

However, the ARIMA model turns out to be inferior in which the model produced much worse results compared to the other two LSTM models. Although, the researchers have anticipated it to perform worse, the researchers did not think that it will be that much worse. Figure 6 shows the results given by the ARIMA model.

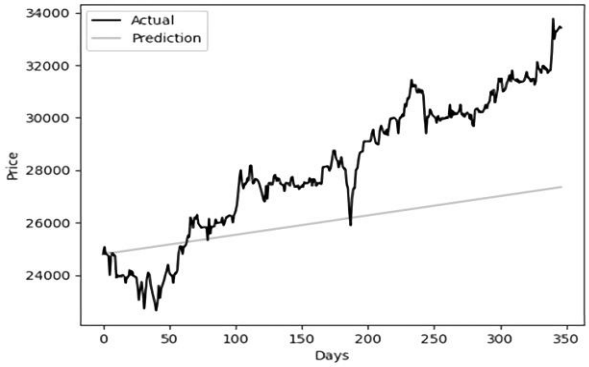

Fig. 6. The prediction result of the ARIMA(1,1,1) model.

Table VI lists the final results of the experiment in different performance metrics. The SE-LSTM gave the best results of all methods used. Meanwhile, the ARIMA model performed much worse than the other two methods used.

TABLE VI
END RESULTS IN DIFFERENT METRICS

Methods	Performance Metrics			
	RMSE	MAE	MAPE	R^2 Score
SE-LSTM	538.914	402.977	1.437%	0.961
LSTM	661.041	510.662	1.841%	0.927
ARIMA(1,1,1)	2809.015	2402.041	9.867%	-0.127

V. CONCLUSION

This experiment was implemented using the combination of the LSTM deep learning model and a novel metaheuristic optimization algorithm, Search Economics. The SE-LSTM as

the best model acquired gives RMSE around 538.914, MAE around 402.977, MAPE around 1.437%, and R^2 Score around 0.961. There was no overfitting found when comparing the prediction to the actual data, which is a good thing. The SE-LSTM provided better results when compared to an non-optimized LSTM model and the ARIMA model using auto_arima function. From what was given as the results, it was proven that ARIMA is not suitable to predict random walk time-series such as stock prices. The ARIMA produced results much worse when compared to the other two models.

Even though we already used a metaheuristic optimization algorithm, optimizing an LSTM model still requires a lot of resources and time. There is simply no way around it, as all deep learning methods need massive computational power. However, compared to manually optimizing things, using heuristic optimization algorithms surely provides much faster execution. While deciding which algorithms to use, we found a new novel heuristic optimization algorithm called Search Economics. It turns out that the algorithm was doing a great job at optimizing our LSTM model. The results however is not that much of a difference when compared to an LSTM model designed by a slightly experienced data scientist.

REFERENCES

[1] Y.-F. Huang and R. Startz, "Improved recession forecasts considering stock market volatility," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.3297949.

[2] B. Shiv, G. Loewenstein, A. Bechara, H. Damasio, and A. R. Damasio, "Investment behavior and the negative side of emotion," *Psychol. Sci.*, 2005, doi: 10.1111/j.0956-7976.2005.01553.x.

[3] J. S. Lerner and D. Keltner, "Fear, anger, and risk," *J. Pers. Soc. Psychol.*, 2001, doi: 10.1037/0022-3514.81.1.146.

[4] M. Grinblatt, M. Keloharju, and J. Linnainmaa, "IQ and stock market participation," *J. Finance*, 2011, doi: 10.1111/j.1540-6261.2011.01701.x.

[5] E. Gilbert and K. Karahalios, "Widespread worry and the stock market," in *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, 2010.

[6] R. Tumarkin and R. F. Whitelaw, "News or noise? Internet postings and stock prices," *Financ. Anal. J.*, 2001, doi: 10.2469/faj.v57.n3.2449.

[7] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *J. Comput. Sci.*, 2011, doi: 10.1016/j.jocs.2010.12.007.

[8] X. Qing and Y. Niu, "Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM," *Energy*, 2018, doi: 10.1016/j.energy.2018.01.177.

[9] G. Chniti, H. Bakir, and H. Zaher, "E-commerce time series forecasting using LSTM neural network and support vector regression," in *ACM International Conference Proceeding Series*, 2017, doi: 10.1145/3175684.3175695.

[10] S. McNally, J. Roche, and S. Caton, "Predicting the Price of Bitcoin Using Machine Learning," in *Proceedings - 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2018*, 2018, doi: 10.1109/PDP2018.2018.00060.

[11] A. Saxena and T. R. Sukumar, "Predicting bitcoin price using LSTM and compare its predictability with ARIMA model," *Int. Journa Pure Appl. Math.*, 2018.

[12] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 2019, doi: 10.1109/ICMLA.2018.00227.

[13] C. W. Tsai, "An effective WSN deployment algorithm via search economics," *Comput. Networks*, 2016, doi: 10.1016/j.comnet.2016.01.005.

[14] W. Qiu, X. Liu, and L. Wang, "Forecasting shanghai composite index based on fuzzy time series and improved C-fuzzy decision trees," *Expert Syst. Appl.*, 2012, doi: 10.1016/j.eswa.2012.01.051.

[15] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Syst. Appl.*, 2011, doi: 10.1016/j.eswa.2011.02.068.

[16] A. H. Moghaddam, M. H. Moghaddam, and M. Esfandyari, "Stock market index prediction using artificial neural network," *J. Econ. Financ. Adm. Sci.*, 2016, doi: 10.1016/j.jefas.2016.07.002.

[17] M. Qiu and Y. Song, "Predicting the direction of stock market index movement using an optimized artificial neural network model," *PLoS One*, 2016, doi: 10.1371/journal.pone.0155133.

[18] D. M. Q. Nelson, A. C. M. Pereira, and R. A. D. Oliveira, "Stock markets price movement prediction with LSTM neural networks," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

[19] Q. Jiang, C. Tang, C. Chen, X. Wang, and Q. Huang, "Stock Price Forecast Based on LSTM Neural Network," *Proceedings of the Twelfth International Conference on Management Science and Engineering Management Lecture Notes on Multidisciplinary Industrial Engineering*, pp. 393–408, 2018.

[20] P. G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, 2003, doi: 10.1016/S0925-2312(01)00702-0.

[21] P. Chen, T. Pedersen, B. Bak-Jensen, and Z. Chen, "ARIMA-based time series model of stochastic wind power generation," *IEEE Trans. Power Syst.*, 2010, doi: 10.1109/TPWRS.2009.2033277.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, 1994, doi: 10.1109/72.279181.

[23] J. F. Kolen and S. C. Kremer, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Networks*, 2010.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, 1997, doi: 10.1162/neco.1997.9.8.1735.