JAKARTA COMPOSITE INDEX FORECAST UNDERSTANDING VOLATILITY USING LONG-SHORT MEMORY

Dr. Pravin Ramesh Gundalwar[1], Dr. Amol D. Potgantwar[2], Dr. Mrs. Kamini Ashutosh Shirsath[3]

Article History: Received: 12.12.2022 **Revised:** 29.01.2023 **Accepted:** 15.03.2023

Abstract

This article covers how the model was created and the outcomes of using LSTM to forecast JCI (Jakarta Composite Index) volatility (Long Short-Term Memory). Using multiple hyperparameters, the LSTM models were evaluated in several different scenarios. The best model is the one with the lowest RMSPE and RMSE among all models when it comes to the performance of volatility prediction on LSTM. Based on test findings, it is discovered that LSTM models can accurately forecast JCI volatility. All of the models utilised have low RMSPE and RMSE values.

Keywords: Prediction, Volatility, LSTM, RMSPE, RMSE

[1]Professor, Department of Computer Science, School of Engineering and Technology, Sandip University, Nashik, Maharashtra 422213
[2]Professor, Department of Computer Engineering, Sandip Institute of Technology & Research Centre, Nashik, Maharashtra 422213
[3]Professor, Department of Computer Engineering, Sandip Institute of Engineering and Management, Nashik, Maharashtra 422213

Email: [1]pravin.gundalwar@sandipuniversity.edu.in, [2]amol.potgantwar@sitrc.org, [3]kamini.nalavade@siem.org.in

DOI: 10.31838/ecb/2023.12.s3.113

1. Introduction

Due to their potential for substantial profits, stocks are financial instruments that are highly valued in the capital market. Another point that has to be noted is that stock investments fall into the category of investments that are high risk owing to high swings, meaning that owners might suffer a very substantial financial loss. The capital loss is a loss as a result of the negative difference between the buying price and the selling price, according to (Hurri et al., n.d.). As a result, thorough research is required before choosing to purchase a stock. If the price of the acquired shares keeps falling, many factors must be taken into account in order to forecast alternative scenarios and prevent losses.

Although the movement of stock values changes extremely quickly, it is still feasible to foresee or forecast using a variety of techniques for the benefit of shareholders. Volatility shows variations in stock prices. Investors must take into account risks that may be categorised as high and low volatility. The danger increases with volatility, and vice versa. In order to improve upon earlier research and develop the most accurate approach for forecasting market volatility, more studies on stock volatility are always being conducted. The investigation is still being developed today. The accuracy of the prediction model, the length of time it is used to make predictions because some models can only be used for a short time while others need a longer period, and problems with prediction error are just a few of the issues that can arise when forecasting or predicting stock volatility. Based on some of these issues, a technique is required to forecast stock volatility with high accuracy, a substantial amount of time to apply, or a minimal amount of mistake in order to ensure that the outcomes of the prediction are accurate. The better the model is at predictions, the smaller the error value.

Deep learning is the model that is also being created and utilised for prediction. Machine learning includes deep learning (Kurniawan & Sumirat, 2020). Recent studies have demonstrated that deep learning models are more accurate in forecasting financial markets than conventional machine learning models, according to (Yun et al., 2021) .'s research. The LSTM (Long Short Term Memory) architecture is used to anticipate outcomes more precisely. LSTM features a memory cell that can correlate the memory of prior events with the input of future occurrences, making it suited for forecasting time series financial data such as stocks.(Eachempati et al., 2021) asserts that LSTM is a dependable option when great data accuracy and little volatility are required. In this study, LSTM will be utilised to forecast the volatility of the JCI (Jakarta Composite Index) shares on the IDX (Indonesia Stock Exchange) using the performance metrics of RMSE (Root Mean Square Error) and RMSPE (Root Mean Square Percentage Error). The RMSPE measurement may be used to determine if a consistent estimate took place in a certain experiment (Chai & Draxler, 2014). As outliers are present in the data collected and do not wish to be disregarded, the RMSE metric is utilised since it is outlier sensitive (Doyog et al., 2021). The goal of this volatility forecast is to reduce current risks. Because of the strong heteroscedasticity and variation of stocks, reducing mistakes to the absolute minimum will significantly aid in obtaining more accurate results. The JCI is an index that is closely watched by many individuals, including investors and other participants in the capital market, since it tracks the price performance of all stocks listed on the Main Board and Development Board of the IDX (Doyog et al., 2021). Issuers on the JCI may change as a result of the JCI's ongoing evaluation. The JCI was chosen as the data for this study due to its significant position in the stock market. The study's identification of the issue—very dynamic stock price movements that turn equities into high-risk financial instruments—will have a significant impact on how capital market players choose to proceed.

This research has been modified by using realised volatility as an LSTM input. The design of the LSTM modelling has been altered, and this involves the selection of different hyperparameters and the usage of the Lambda layer. The Lambda layer's presence serves to enable the construction of functional and sequential API (Application Programming Interface) models using arbitrary expressions as layers (Karim, 2018).

2. Design of volatility prediction using LSTM

The design for JCI volatility prediction using LSTM is shown below.

- Using data from Yahoo Finance's JCI dataset between January 31, 2011, and January 29, 2021.
- Cleaning up data, which include getting rid of useless information and finding and getting rid of NaN values.
- Just the closing price is utilised in this calculation, followed by a plot of the dataset's close price.
- Calculating the close price's log returns and creating a log return graphic.
- Visualizing the log return distribution and comparing the results to a normal distribution.
- Calculating the observation values, mean, standard deviation, minimum, median, and maximum, as well as the skewness and kurtosis of log returns.

- Using the ADF (Augmented Dickey Fuller) test to examine the stationarity of log returns. If the test's outcome is stationary, calculating daily realised volatility from log returns is the next step. If the test's outcome is nonstationary, differencing is used to assist in obtaining stationary data.
- Calculation of the past, present, and future volatility.
- Removing NaN values from volatility data, both past and present.
- Dividing data log returns into train and test data, as well as current and future volatility, in accordance with Table 1.
- Min-max scaling to normalise both the present and future volatility.
- Using the training dataset's most recent volatility as the LSTM input.
- Calculation of different hyperparameters (hidden layer neuron units, batch sizes, epochs).
- Overfitting prevention: usage of APIs (EarlyStopping and ModelCheckpoint).
- Choose the validation split option to divide the training dataset into the training and validation datasets.

Modelling of LSTM model:

- Displaying the learning curve on the training dataset and LSTM model validation.
- Predicting each LSTM model's volatility.
- Min-max scaling for normalising predictive volatility.
- Representation of the testing dataset's comparison of prediction and futures volatility.
- RMSPE and RMSE calculations for all LSTM models.
- Compiling the top LSTM model.

Table 1: Splitting dataset

Type Data	Numbers	Percentage
Train data	2139 days	89,55%
Test data	252 days	10,45%

2. Findings and Discussion

Next, the daily close prices are shown using the cleaned-up JCI information from January 31, 2011, to January 29, 2021. Figure 1 depicts the daily close price plot. The JCI data utilised throughout the observation period displays extremely erratic price changes, as shown in Figure 1. On February 19, 2018, the JCI reached its highest price throughout this time period, reaching IDR 6,689.00. In September 2015 and March 2020, the JCI fell by quite large amounts. According to (Karmiani et al., 2019), in addition to domestic factors like Indonesia's slowing economic growth, the weakening of the rupiah, declining financial performance, and general economic conditions, negative sentiment on international issues like the Greek debt crisis, rising interest rates Fed interest rates, falling commodity prices, and the slowdown in China's economy lowered the JCI to a price of IDR 4,121.00. Politics in Indonesia are not peaceful.

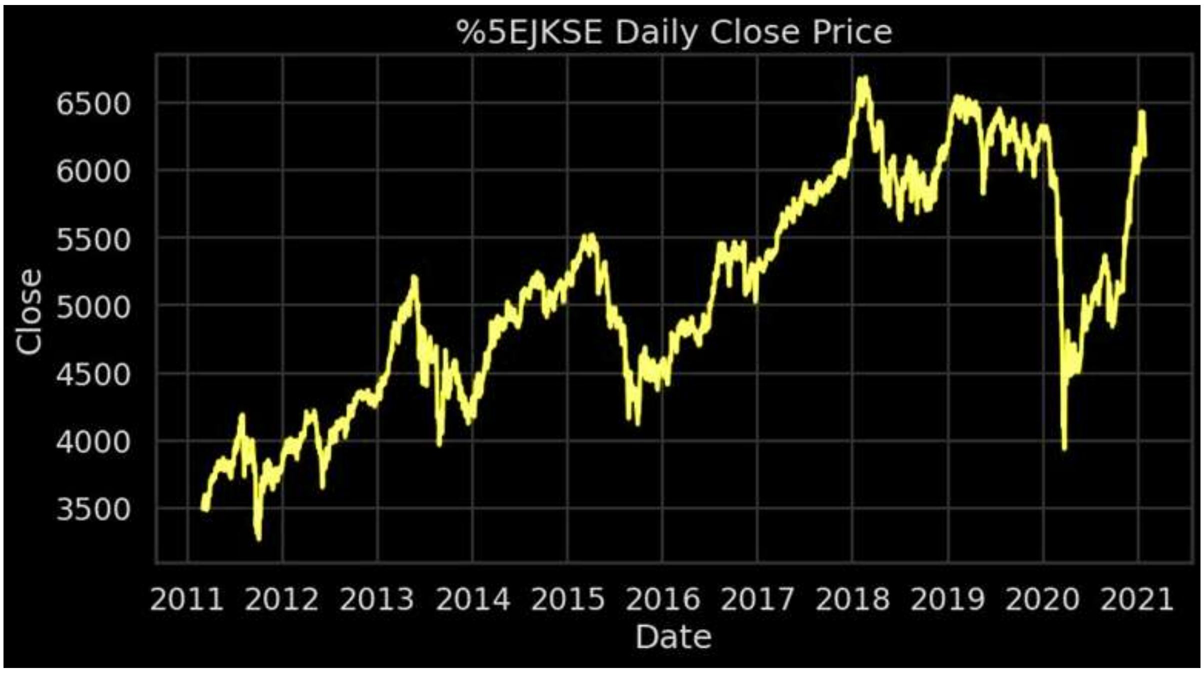


Figure 1: Daily closing price plot

Due to the COVID-19 pandemic, which occurred and significantly impacted the Indonesian economy, a temporary stoppage of trade was implemented (trading halt) in March 2020, when the JCI plummeted to a price of IDR 3,938.00 (Putra, 2018). The beginning of 2021 will mark the start of a recovery phase, which is characterised by the occurrence of stable circumstances and a rise in the number of enterprises enjoying profits, according to (Putra, 2018). The conditions in 2020 are highly unpredictable and not very favourable. The log returns may be computed using the close prices that were acquired. Figure 2 shows the plot of the log returns calculation, and Figure 3 shows the distribution of the log returns.(Thakkar, 2021)

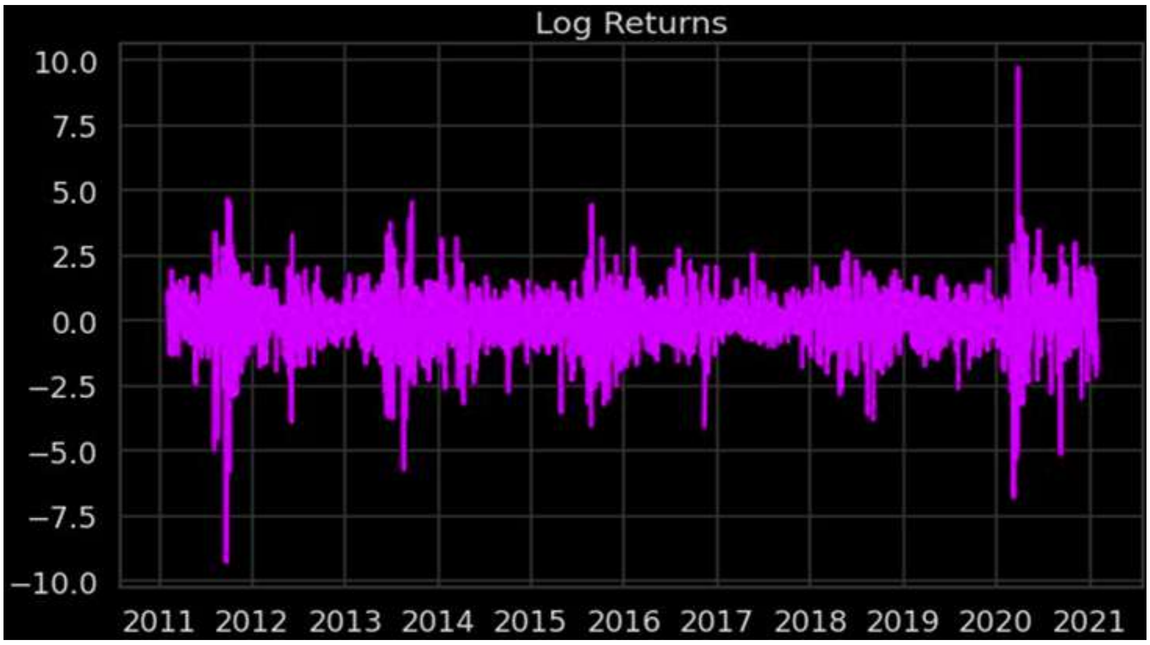


Figure 2: Log return plot

Figure 3 illustrates positive kurtosis (leptokurtic), as evident by the greater peak and fatter tail of the log returns compared to the typical normal distribution.(Liu et al., 2022) Table 2 provides a descriptive analysis of log returns across the observation period.

Table 2: Log return descriptive analysis

Observation	2433
Mean	0.023093
Standard Deviation	1.094169
Minimum	-9.299684
Median	0.086737
Maximum	9.704219
Skewness	-0.533845
Kurtosis	8.137918

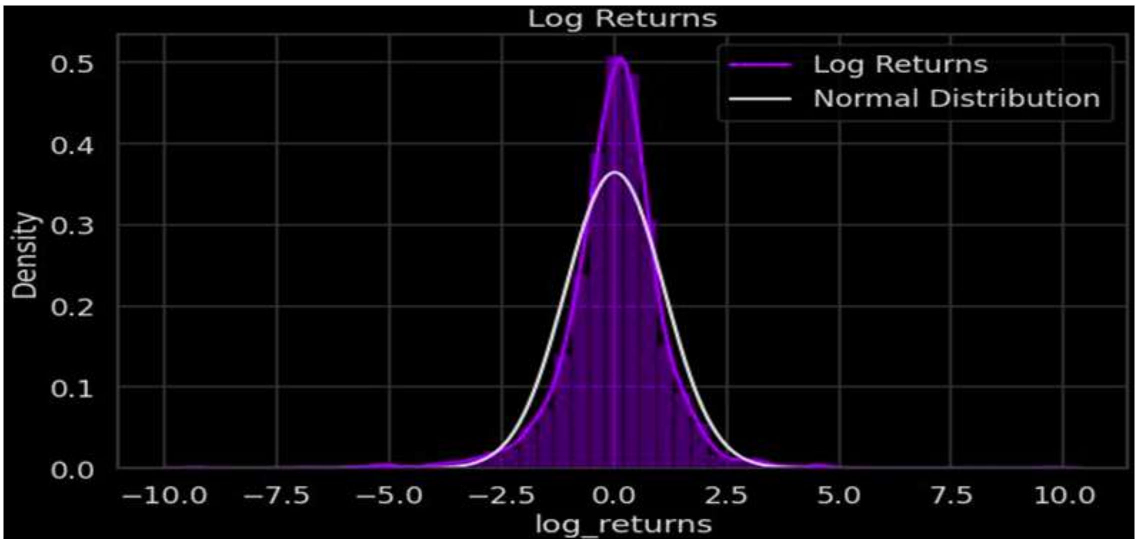


Figure 3: A plot comparing the conventional normal distribution with the log returns distribution

- **Test for Stationarity**

It is necessary to examine the stationarity of log returns. The ADF test may be used with the following hypothesis to examine stationarity.

- H_0: The time series data is not stationary and has a unit root.
- H_1: Time series data is steady and has no unit root.

If the data are not stationary, a differencing procedure must be performed until the data are stationary. Table 3 displays the outcomes of the ADF test performed on the log returns data.

Table 3: Log return data from the ADF test

ADF Statistics	11.9235843
p-value	$4.9665607 \times 10^{-22}$
Critical Value 5%	-2.8627

The log returns do not have a unit root and are stationary, as may be inferred from Table 3's p-value of 0.05, which indicates that the hypothesis is both rejected and accepted.

- **Calculating realized volatility**

Scaling is done as follows to get the daily realised volatility at specific intervals, which in this example might be weekly (5 days), monthly (21 days), or annual (252 days).

$$\mathrm{RV_{daily}} = \sqrt{\sum_{i=1}^{T} r_i^2} \times \sqrt{\frac{1}{n-1}}$$

where n is the number of days in the interval that were utilised. The daily realised volatility is plotted using various window intervals in Figure 4 below.

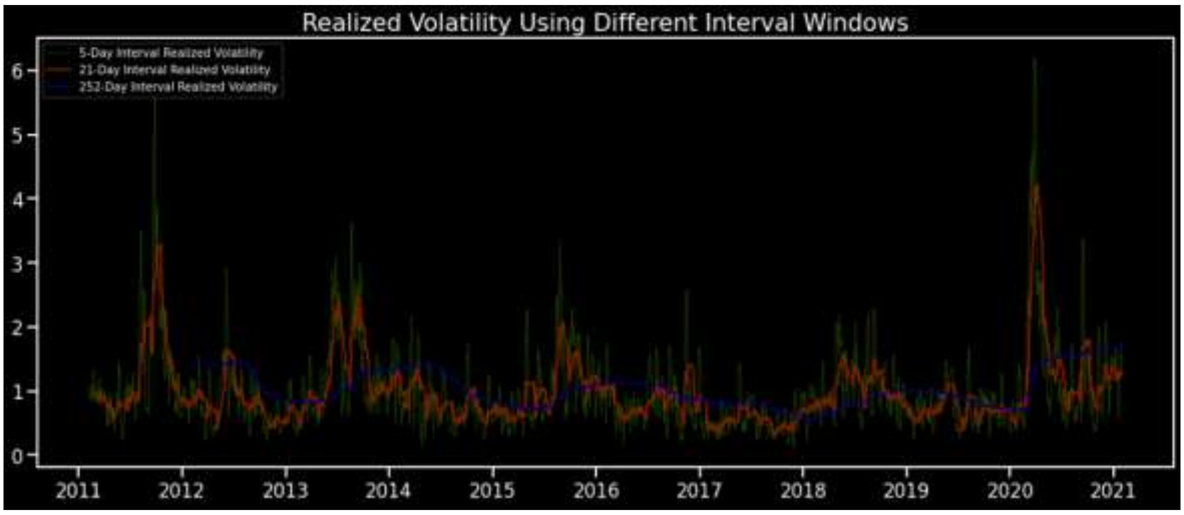


Figure 4: The realised volatility for windows of 5, 21, and 252 days

The study's window period was 21 days, or roughly one month's worth of stock activity. The 5-day period was too busy to detect important patterns or information, while the 252-day interval lowered volatility too much. Hence, the 21-day interval was chosen. Time series prediction models forecast future values using values that have already been seen. In this study, the realised volatility value is converted to the current volatility value, which is then shifted backward by one index to provide the target future volatility value. It may be claimed that today is the future for yesterday, thus if today's volatility is moved back one day, it can be utilised as the output future intended for yesterday. This value is then used for model performance testing and training. Figure 5 provides a visualisation of past, present, and future volatility.

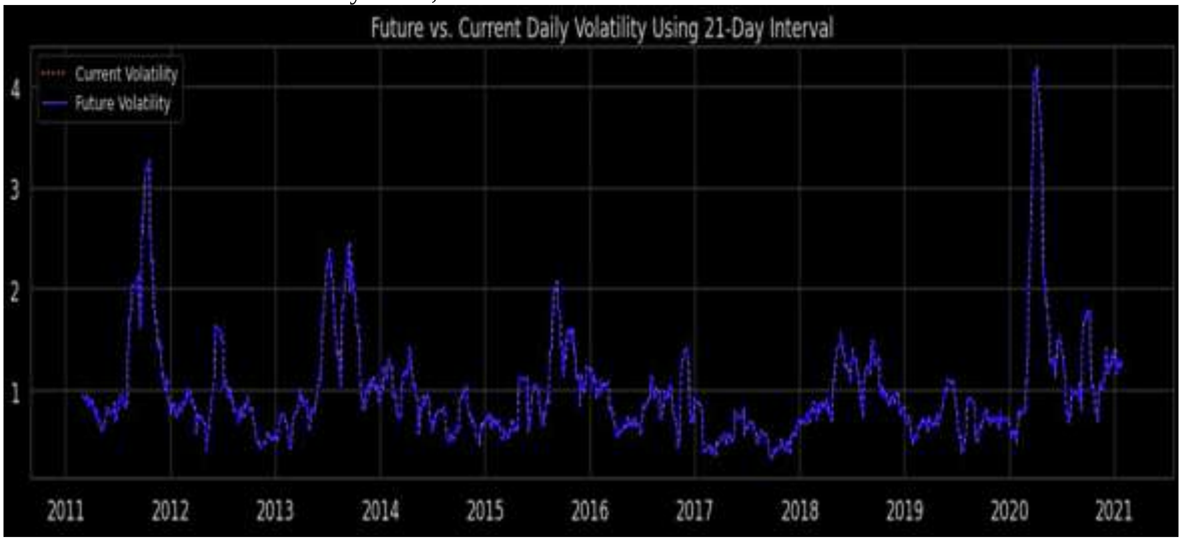


Figure 5: Future and current volatility with intervals of 21 days

- **Tuning for hyperparameters**

As there is no set formula for how many hyperparameters should be used, the hyperparameters in this study were evaluated first, and the smallest RMSPE and RMSE values were used to determine which hyperparameters provided the best model for forecasting stock volatility of JCI data. The specifics of the hyperparameters employed in this study's LSTM are provided in Table 4.

Table 4: lists specifics of the applied hyperparameters

No.	Hyperparameter	Numbers
1	Unit neurons in hidden	16, 32, 64
2	Batch sizes	16, 32, 64
3	Epochs	100, 200, 500, 1000

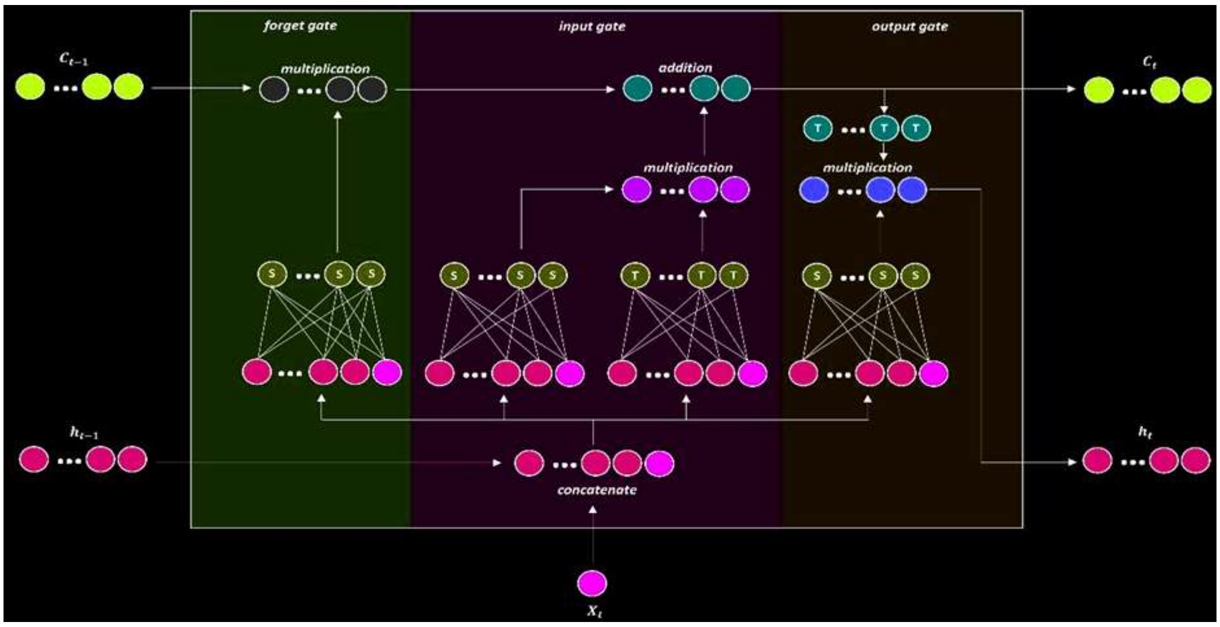
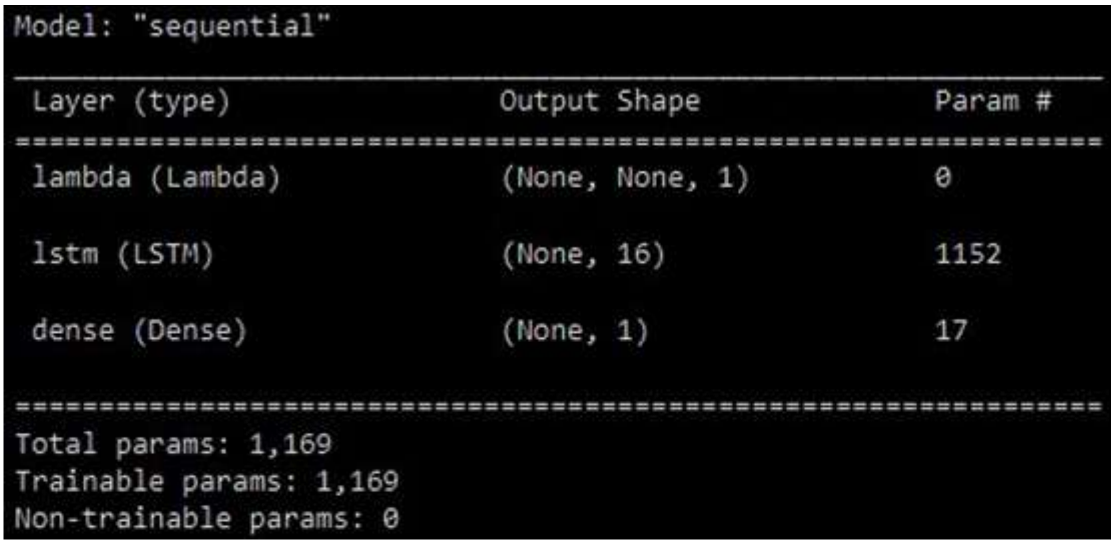


Figure 6: LSTM cell

In this study, validation split=0.2 is employed, such that the training dataset is split automatically by the fit() method into training and validation datasets, with the proportion of training datasets used for validation datasets of 0.2. RMSPE validation is the performance metric that is utilised as a monitor to determine when training should cease, therefore training will end if RMSPE validation stops rising. Model Checkpoint, a separate callback that can save the best model during training, is utilised because the final model after training is ended by Early Stopping might not be the best model in the validation dataset. Model Checkpoint's monitor also uses RMSPE validation. Lambda layer modelling is utilised in LSTM.

The LSTM layer is utilised in LSTM modelling, and the LSTM layer contains LSTM cells. The LSTM cell utilised in this investigation is shown in Figure 6 and is based on (Yang et al., 2020). In Figure 6, the LSTM cell receives three inputs: the input value at the current time step t-1 ($C_{(t-1)}$), the hidden state value from the previous time step ($h_{(t-1)}$), and the cell state value from the previous time step (t-1). Figure 6 demonstrates the LSTM's 4 FFNNs, which are located on the forget gate, two input gates, and one output gate. A circle with a distinct hue represents the dimensions of the input and output tensors. The number of neurons' hyperparameter units in the hidden layer of the LSTM cell determines the size of the hidden state ($h_{(t-1)}$) and cell state ($C_{(t-1)}$) vectors in Figure 6. For the purposes of this study, d may be worth 16, 32, or 64. The vector($h_{(t-1)}$) and ($C_{(t-1)}$) should have the same dimensions. The vectors ($h_{(t-1)}$) and ($C_{(t-1)}$), as well as ($h_{(t-1)}$) and ($C_{(t-1)}$), must have the same dimensions. A vector with one dimension, which is defined by the number of features, is the input to Figure 6. In this study, realised volatility is the only characteristic used as an input to the LSTM.



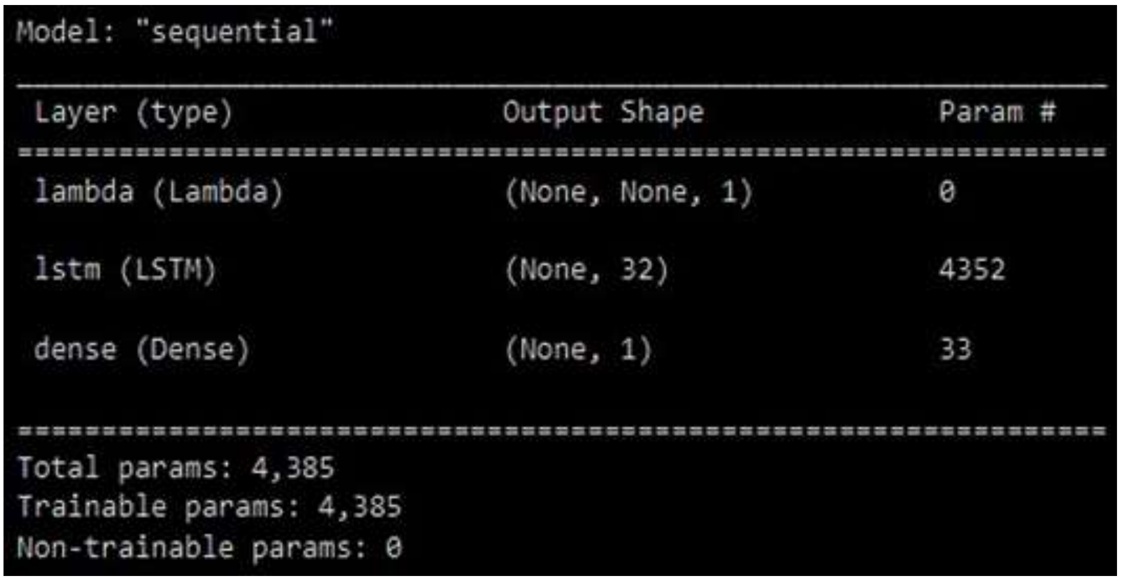
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, None, 1)           0

 lstm (LSTM)                 (None, 16)                1152

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 1,169
Trainable params: 1,169
Non-trainable params: 0
```

Figure 7: LSTM parameters (16 units)



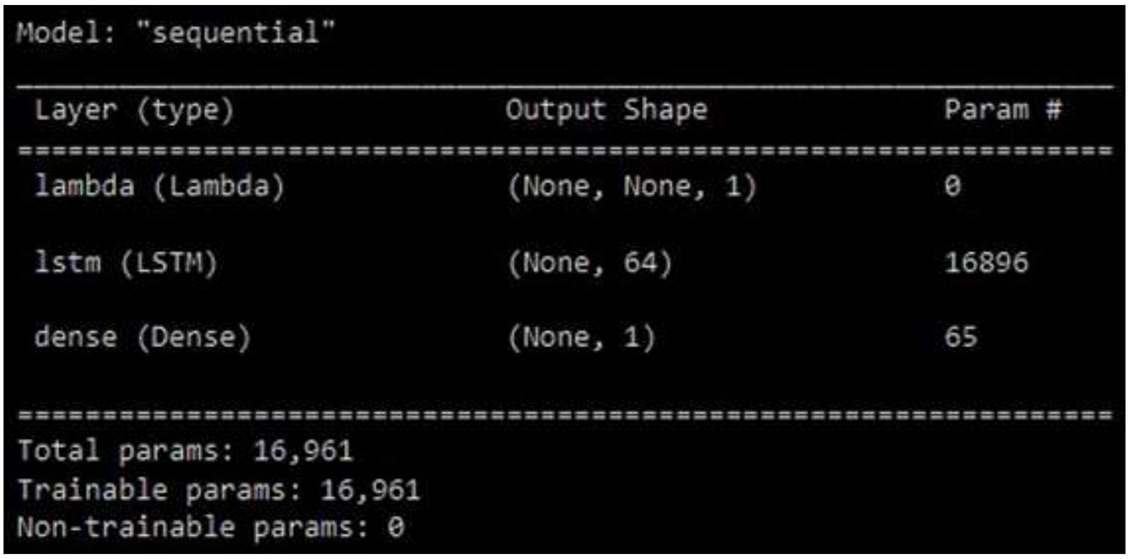
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, None, 1)           0

 lstm (LSTM)                 (None, 32)                4352

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
```

Figure 8: LSTM parameters (32 units)



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, None, 1)           0

 lstm (LSTM)                 (None, 64)                16896

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
```

Figure 9: LSTM parameters (64 units)

Figures 7, 8, and 9 show an overview of the design and the parameters of the LSTM (16 units), LSTM (32 units), and LSTM (64 units) models, respectively. As there are no neuron units in the Lambda layer, the parameter is 0. Figure 10 depicts the LSTM modelling architecture in detail.

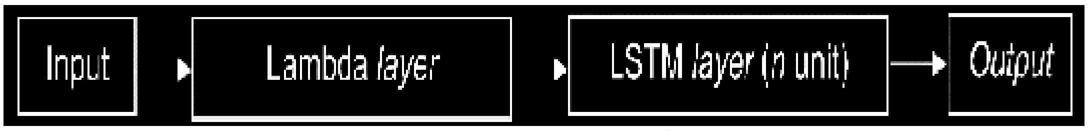


Figure 10: The LSTM modelling framework.

Table 5: RMSPE and RMSE LSTM models with different hyperparameters

	Epochs = 100		
Batch Size	**Model (Neuron Hidden)**	**RMSPE Testing**	**RMSE Testing**
16	LSTM 16 unit	0.117218	0.05981
	LSTM 32 unit	0.112257	0.045326
	LSTM 64 unit	0.111903	0.044433
32	LSTM 16 unit	0.116204	0.057198
	LSTM 32 unit	0.11368	0.045036
	LSTM 64 unit	0.114863	0.042206
64	LSTM 16 unit	0.115225	0.052383
	LSTM 32 unit	0.113061	0.043457
	LSTM 64 unit	0.112155	**0.041049**
	Epochs = 200		
Batch Size	**Model (Neuron Hidden)**	**RMSPE Testing**	**RMSE Testing**
16	LSTM 16 unit	0.116252	0.057017
	LSTM 32 unit	0.112624	0.046369
	LSTM 64 unit	0.111746	0.044081
32	LSTM 16 unit	0.113567	0.047196
	LSTM 32 unit	0.112563	0.045633
	LSTM 64 unit	0.111696	0.042764
64	LSTM 16 unit	0.114321	0.051041
	LSTM 32 unit	0.112774	0.047073
	LSTM 64 unit	0.111572	0.042945
	Epochs =500		
Batch Size	**Model (Neuron Hidden)**	**RMSPE Testing**	**RMSE Testing**
16	LSTM 16 unit	0.112865	0.046437
	LSTM 32 unit	0.112281	0.04578
	LSTM 64 unit	0.11211	0.044225
32	LSTM 16 unit	0.115828	0.056293
	LSTM 32 unit	0.112503	0.045414
	LSTM 64 unit	0.111777	0.043474
64	LSTM 16 unit	0.115952	0.056139
	LSTM 32 unit	0.11246	0.04575
	LSTM 64 unit	0.111642	0.043214
	Epochs = 1000		
Batch Size	**Model (Neuron Hidden)**	**RMSPE Testing**	**RMSE Testing**
16	LSTM 16 unit	0.114435	0.049963
	LSTM 32 unit	0.1129	0.046128
	LSTM 64 unit	0.11195	0.043569
32	LSTM 16 unit	0.113284	0.048368
	LSTM 32 unit	0.112285	0.044674
	LSTM 64 unit	0.111746	0.043285
64	LSTM 16 unit	0.11456	0.052073
	LSTM 32 unit	0.11216	0.044984

	LSTM 64 unit	**0.11154**	0.042801

It will be challenging to analyse all 2160 datasets learned in LSTM modelling at once because of their size. The procedure is separated into batches with sizes that have been specified in Table 4 in order to process the full dataset. By using a batch size of 16, each process inputs 16 data one at a time into the LSTM architectures as seen in Figures 10 until the full dataset has been processed. Because this study employs time series data, the chosen data are sequential (not random). This is often referred to as 1 epoch if all of the data has been processed. Iteration is the term for the number of processes required to complete one epoch; in this example, there are 2160: 16 = 135 iterations. The same processing is applied to the other hyperparameters listed in Table 4.

- **LSTM Model**

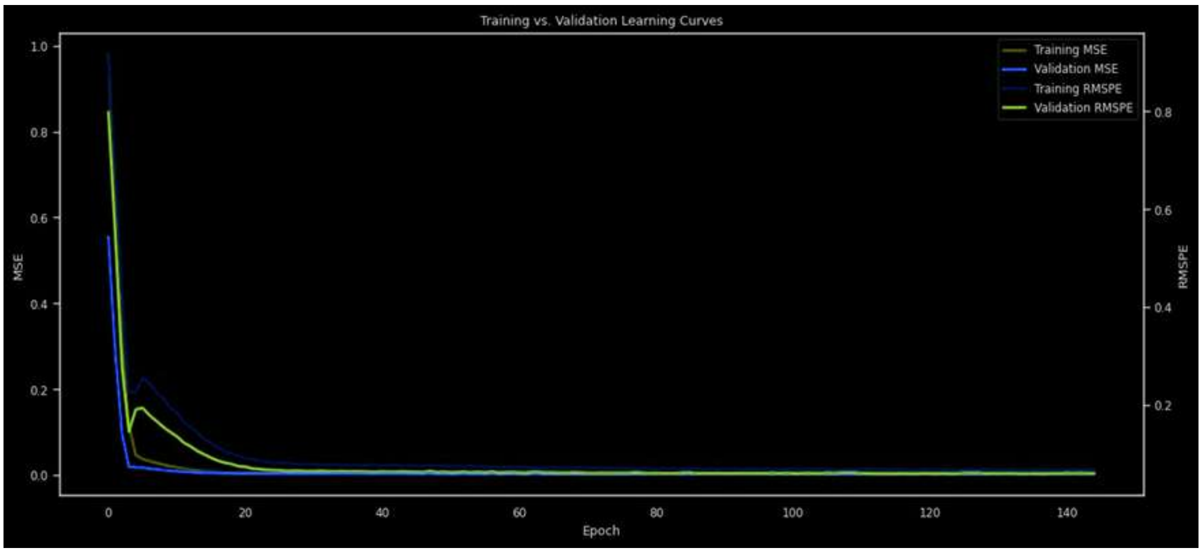


Figure 11: Learning curve LSTM (64 units), 64-unit batch size, and 1000 epochs

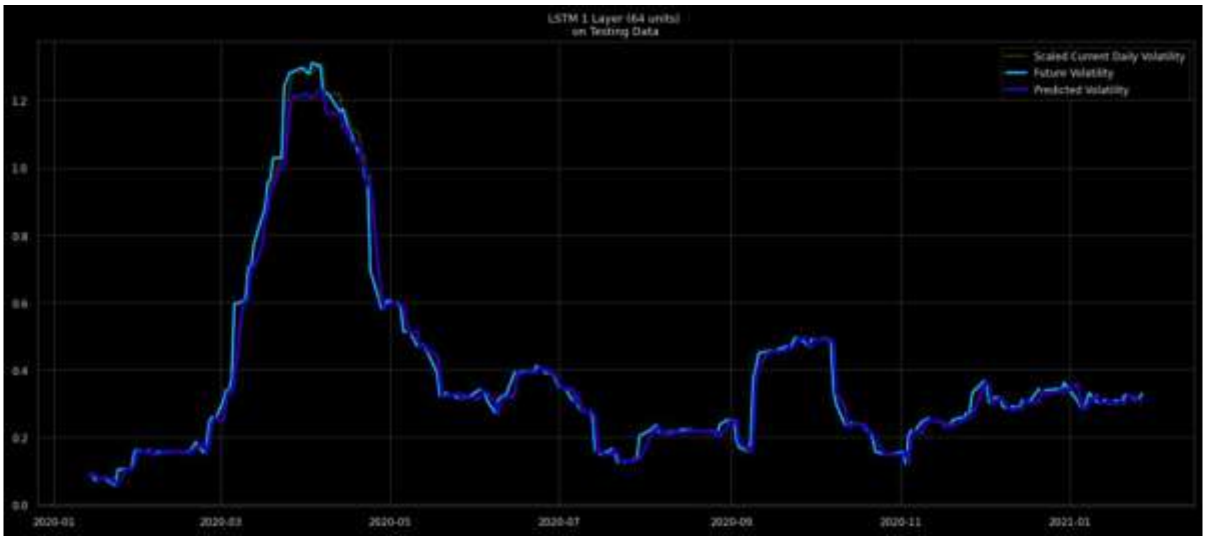


Figure 12: Batch size 64 epoch 1000 for LSTM volatility (64 units)

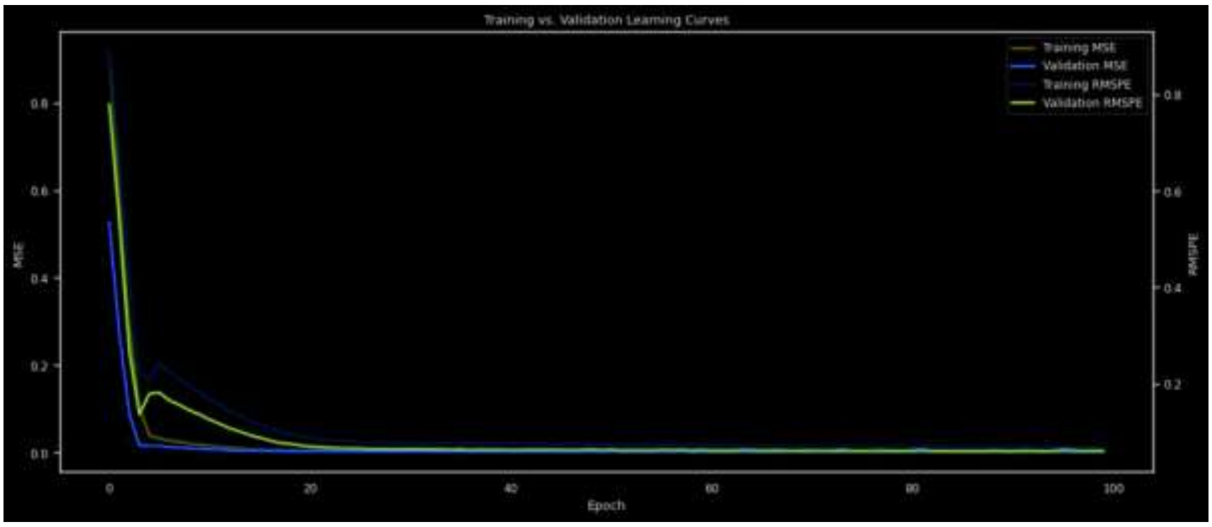


Figure 13: LSTM learning curve (64 units), 64-unit batch size, 100-epoch

Identification of the optimal model for the prediction may be observed from the minimum RMSPE and RMSE values in the testing data. Table 5 provides the RMSPE and RMSE LSTM models of various hyperparameters on the testing data.

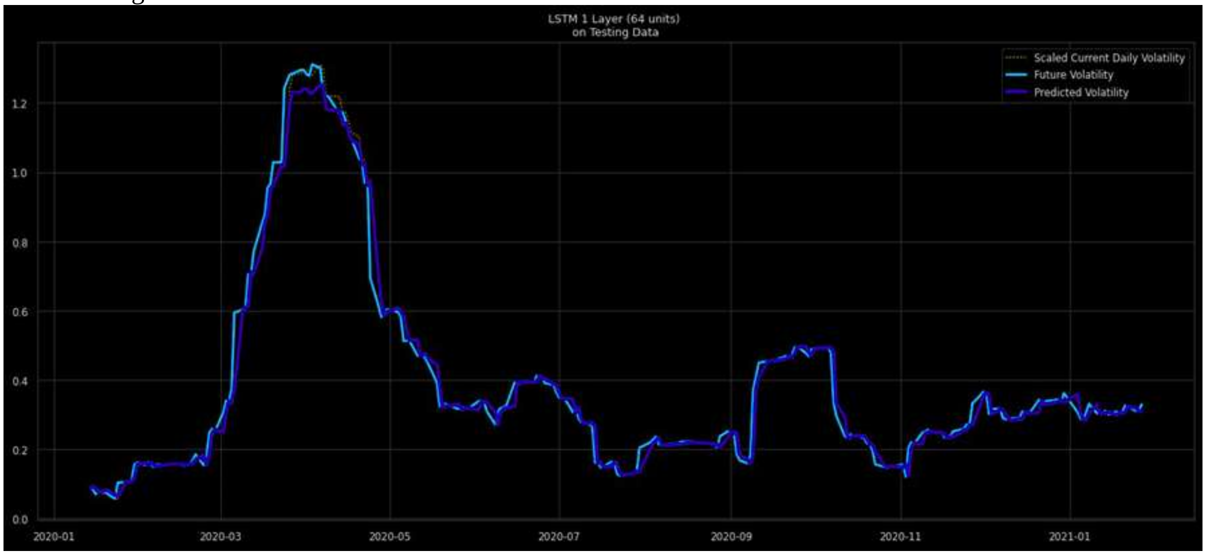


Figure 14: Volatilitas LSTM batch size 64 epoch 100 (64 units)

The test reveals that the LSTM model (64 units) with hyperparameter batch size 64 and epochs 1000 produced the least RMSPE value of 0.11154, while the LSTM model (64 units) with hyperparameter batch size 64 and epochs 100 produced the smallest RMSE value of 0.041049. Figures 11 and 12 show the learning curve visualisation and volatility visualisation for the LSTM model (64 units) with hyperparameter batch size 64 and epochs 1000. Figures 13 and 14 show the learning curve visualisation and volatility visualisation for the LSTM model (64 units) with hyperparameter batch size 64 and epochs 100.

3. Conclusion

Based on the study's findings, it can be said that LSTM input is determined by computing the log returns, which are then utilised to calculate realised volatility. The JCI's close prices are used to calculate the log returns. The best model is the one with the lowest RMSPE and RMSE among all models when it comes to volatility prediction performance on LSTM. Testing of the LSTM model was done utilising a variety of situations and hyperparameters. According to the test findings, the LSTM can accurately forecast the volatility of the JCI based on the RMSPE and RMSE. The least value for each model employed is 0.11154 for the smallest RMSPE, which was achieved from the LSTM model (64 units) batch size 64 epochs 1000, and 0.041049 for the smallest RMSE, which was acquired from the LSTM model (64 units) batch size 64 epochs 100.
Given the findings and the study's limitations, it would be preferable to use a method that can

choose the ideal hyperparameter automatically. Additionally, hybrid LSTM can be used in conjunction with other techniques to improve predictions on time series data, such as stock prices, weather, gold prices, and other variables.

4. Reference

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247–1250.

Doyog, N. D., Lin, C., Lee, Y. J., Lumbres, R. I. C., Daipan, B. P. O., Bayer, D. C., & Parian, C. P. (2021). Diagnosing pristine pine forest development through pansharpened-surface-reflectance Landsat image derived aboveground biomass productivity. Forest Ecology and Management, 487, 119011.

Eachempati, P., Srivastava, P. R., Kumar, A., Tan, K. H., & Gupta, S. (2021). Validating the impact of accounting disclosures on stock market: A deep neural network approach. Technological Forecasting and Social Change, 170, 120903.

Hurri, I., Munajat, A., Santoso, M. A., di Indonesia, P. K., & Isra, S. (n.d.). Daftar Istilah. PUSAT PENDIDIKAN SDM KESEHATAN BADAN PENGEMBANGAN DAN PEMBERDAYAAN SDM KESEHATAN KEMENTERIAN KESEHATAN TAHUN 2020, 46.

Karim, R. (2018). Animated rnn, lstm and gru.

Karmiani, D., Kazi, R., Nambisan, A., Shah, A., & Kamble, V. (2019). Comparison of predictive algorithms: backpropagation, SVM, LSTM and Kalman Filter for stock market. 2019 Amity International Conference on Artificial Intelligence (AICAI), 228–234.

Kurniawan, W. M., & Sumirat, E. (2020). Evaluation of Risk Parity Asset Allocation Strategy in Indonesia Mutual Fund During 2010–2019 Period. European Journal of Business and Management Research, 5(5).

Liu, X., Zhou, R., Qi, D., & Xiong, Y. (2022). A Novel Methodology for Credit Spread Prediction: Depth-Gated Recurrent Neural Network with Self-Attention Mechanism. Mathematical Problems in Engineering, 2022.

Putra, J. W. G. (2018). Pengenalan Pembelajaran Mesin dan Deep Learning. Wiragotama, Tokyo [JP], 167–172.

Thakkar, P. (2021). Adaptive multi-column multi-stage machine learning pipeline for predicting stock price by solving a nonlinear optimization program. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(11), 6806–6813.

Yang, C., Zhai, J., & Tao, G. (2020). Deep learning for price movement prediction using convolutional neural network and long short-term memory. Mathematical Problems in Engineering, 2020.

Yun, K. K., Yoon, S. W., & Won, D. (2021). Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. Expert Systems with Applications, 186, 115716.