Journal of Applied and Research Computer Science and Information Systems
Vol. 2, No. 1, June 2024, pp. 149~158
ISSN: 2988-294X, DOI: 10.61098/jarcis.v2i1.180

Performance Assessment of ARIMA and LSTM Models in Prediction Using Root Mean Square Error (RMSE)

Andiani[1], Yoel Simanjuntak[2], Ninuk Wiliani[3(✉)]
[1]Pancasila University, Indonesia
[2]Cyber University, Indonesia
[1,3]Asia e University, Malaysia
andiani@univpancasila.ac.id, C70101180012@aeu.edu.my,
ninuk.wiliani@univpancasila.ac.id

Article Info

Article history:

Received May 20, 2024
Revised May 25, 2024
Accepted June 20, 2024

Keywords:

Solana
Time Series
ARIMA
LSTM
RMSE

ABSTRACT

Cryptocurrency is a digital financial asset that serves as a medium of exchange, with its ownership guaranteed using decentralized cryptographic technology, and it has become a growing investment tool. Solana is one of the highly sought-after Cryptocurrencies by investors. The market price of Solana exhibits highly volatile movements, which are considered risky for investment purposes, as it offers both high potential profits and losses. In this regard, time series data prediction models are used to analyze and forecast the price movements of Solana. By comparing the performance of ARIMA and LSTM models in predicting the closing price of Solana using RMSE as a testing metric, the aim is to determine the efficiency level of both ARIMA and LSTM models. The research results show that the ARIMA model with an order of (2,1,3) achieves an RMSE of 0.019 (1.9%) with an accuracy of 98.1%, while the LSTM model with a data training ratio of 70:30%, a batch size of 64, and 500 epochs has an RMSE of 0.075 (7.5%) with an accuracy of 92.5%. The conclusion drawn from the conducted experiments is that, in the case of using time series data samples from Solana, the ARIMA method demonstrates higher accuracy compared to the LSTM method.

Solana, Time series, ARIMA, LSTM, RMSE

Corresponding Author:

Ninuk Wiliani
Asia e University, Malaysia
Email: ninuk.wiliani@univpancasila.ac.id

1. INTRODUCTION

The development of digital technology has led to numerous innovations in the field of finance, particularly in the realm of cryptocurrency[1]. Cryptocurrency is a digital currency that does not have physical form, but is used as a media of exchange to transfer and store assets[2]. It has become a popular investment option, with Solana being a significant cryptocurrency market[3]. Investors in cryptocurrency can make better investment decisions by understanding the price of Solana[4]. Data time series models can be used to analyze and predict Solana prices, providing valuable insights[5][3]. The Autoregressive Integrated Moving Average (ARIMA) and Long Short Term Memory (LSTM) models are used for data time series prediction[6][7]. ARIMA uses a mean absolute percentage error (MAPE) to predict future price trends, while LSTM uses a combination of four gates to provide accurate data predictions[8][9].

The relationship between ARIMA and LSTM in predicting Solana prices is not well understood[10]. This study aims to implement ARIMA and LSTM models in cryptocurrency price prediction to determine the

best model for data time series prediction[11]. The research uses Root Mean Square Error (RMSE) as a comparison between ARIMA and LSTM models, which helps determine the accuracy of predictions[12]. By comparing ARIMA and LSTM, investors can gain valuable insights and make informed decisions about Solana price predictions[10].

2. METHOD

The data collection time series in this study is with scraping technique through Google Colaboratory with Python programming language using yFinance library modules yahoofinace, pandas, and files[13]. In Figure 1 describes the process of data collection with data scrapping technique by performing the installation of yFinance library, importing yaofinance, pandasa, and files, then entering the coin code and time series (time series) to be scraped, after downloading the data and converting the data into csv extensions.

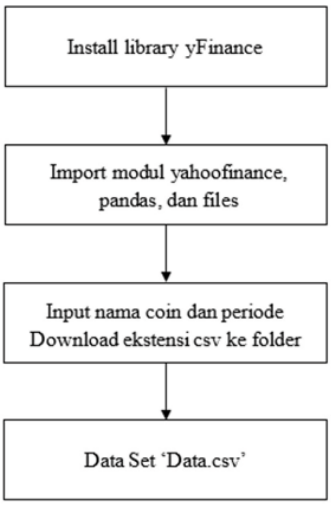

Figure 1. Scraping data solana

The data analysis technique on this compilation is implementing the CRIPS-DM model[14], in Figure 1 CRIPs-DM has 6 phase models that are implemented on the research:

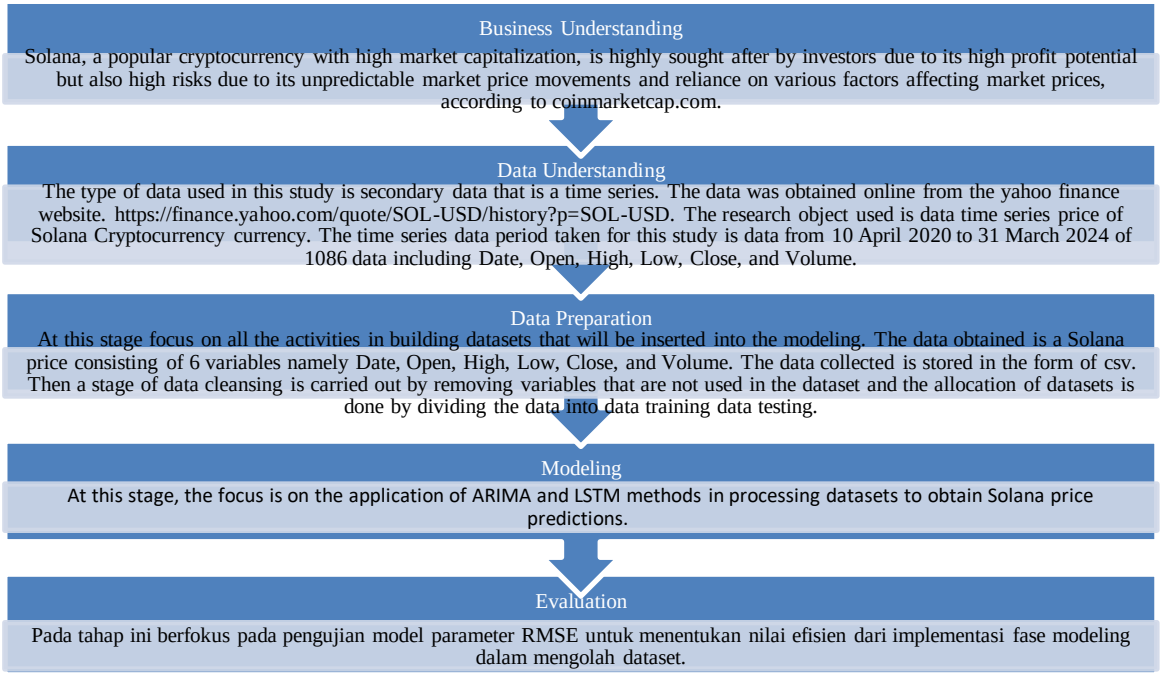

Figure 2. Research Method

3. RESULTS AND DISCUSSION

In this section, the focus is on the application of ARIMA and LSTM methods in processing datasets to obtain Solana price predictions. In Figure 3 is the source code of data scraping technique time series Solana.

```
1. !pip install yFinance
2. import yfinance as yf
3. from google.colab import files
4. import pandas as pd
5. solana = yf.Ticker("SOL-USD")
6. data = solana.history (start = '2020-04-10', end =
   '2023-03-31')
7. data.to_csv('data.csv')
8. files.download('data.csv')
```

Figure 3. Source code scraping data time series Solana

Line 1, use "!pip" to install the "yfinance" library using the pip installer.
Line 2-4, is a source code importing the yfinance library to get stock data, files to download files, and a hint to work with tabular data.
Line 5, ("SOL-USD") is a Solana utility token on the yahoo finance website, which means on this source code it marks that Solana's time series data will be scraped.
Line 6, using the history method on the Solana object to obtain historical data with the date range of the history data specified with start 10 April 2020 and end 31 March 2024. Line 7-8, is the source code for downloading the data of the solana time series, the data will be automatically stored in a device with a data format.csv

```
#Eksprolasi Dataset
data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-04-10	0.832005	1.313487	0.694187	0.951054	0.951054	87364276
1	2020-04-11	0.951054	1.049073	0.765020	0.776819	0.776819	43862444
2	2020-04-12	0.785448	0.956670	0.762426	0.882507	0.882507	38736897
3	2020-04-13	0.890760	0.891603	0.773976	0.777832	0.777832	18211285
4	2020-04-14	0.777832	0.796472	0.628169	0.661925	0.661925	16747614
...	...	...	...	...	...	...	...
1081	2023-03-27	20.976086	20.979824	19.508055	19.921675	19.921675	359607077
1082	2023-03-28	19.922195	20.764208	19.665567	20.470697	20.470697	347656506
1083	2023-03-29	20.465441	21.505144	20.458122	21.111467	21.111467	378726667
1084	2023-03-30	21.113840	21.590513	20.232969	20.551046	20.551046	472449267
1085	2023-03-31	20.552515	21.441999	20.155243	21.170496	21.170496	352864960

1086 rows × 7 columns

Figure 4. Solana data set scraping results

Figure 4 represents the data scraping result of the data set that will be used in this study with a total of 1086 rows with 7 columns covering Date as a time variable from the period April 10, 2020 to March 31, 2024, Open as the opening price variable on a day of price movement of Solana, High as the highest price variability on a Day of Price movement of solana, Close the closing price on one Day of price movements ofsolana, Adj Close closing prices adjusted on one day of prices movement ofolana, Volume volume of transactions on a single day of the price movement.

The processing stage of the index formation with the function "pd.to_datetime" to change the "Date" column in the DataFrame. "data" becomes the datetime data type. This is useful to ensure that the column "DATE" is considered as a single data that can be properly used in subsequent processing.

```
1. plt.figure(figsize=(20,10))
2. sns.set_style('darkgrid')
3. plt.xlabel('Date')
4. plt.ylabel('Close Price')
5. plt.title(' Market Close Price')
6. plt.plot(data['Close'])
```

Figure 5. Source code visualisasi Close coloumn

In Figure 5 is the stage of visualizing or creating a plot line from the "Close" column in the DataFrame. In the function plt is a module of the "matpltlib" aimed at carrying the size of the plot image on the line 1, giving the axis labels x and y on the lines 3-4, giving the title to the plot picture in the line 5, making the line change price over time on the plot photo from the column "Closed" in the DatFrame "data" in line 6. There is also the sns function of the seaborn library on line 2 to set the plot grid style.

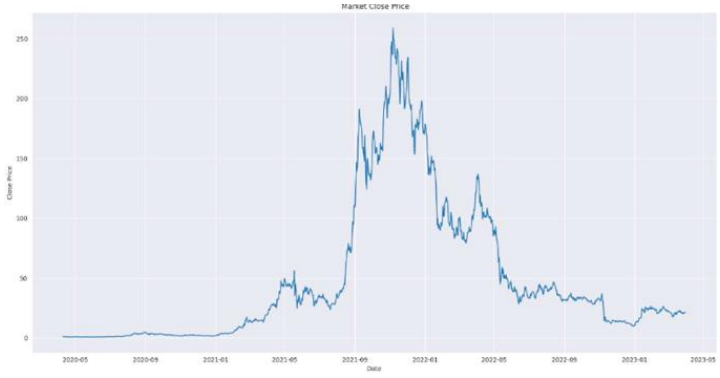

Figure 6. Column Visualization Close

3.1. ARIMA Implementation

In this section, researchers will test the effectiveness of ARIMA models to predict Solana timeseries data. Using the ARIMA model in this study to determine the best order value of ARIMA (p,d,q), the parameter p determines the number of previous value lags that will be used as values to predict the current value, the parameters d are describing the amount of differentiation tests that have been applied to the data, and the q parameter determines how many residual lags (the difference between actual observation values and predicted values) will be considered in the model. (Latif et al., 2023). The ARIMA order value (p,d,q) is obtained from the implementation of the auto_arima() function, which is a module of the pmdarima. The auto _arima () function uses an automatic search approach to find the best order parameter combination.

```
import pmdarima as pm
auto_arima = pm.auto_arima(ts,stepwise=False, seasonal=False)
auto_arima

output:
ARIMA(2,1,3)(0,0,0)[0] intercept
```

Gambar 5. ARIMA Modeling

In Figure 5 is a source code modeling ARIMA using the auto_arima() function of the pdmarima module. On the source code of the ts object as the data argument that will be tested auto_ arima() with stepwise=False means the search used non-stepwise, which means will search order by considering all possible combinations. For seasonal=False shows that the selected ARIMA model has no seasonal components. From the modeling results using the auto_arima() function obtain the best output order that is ARIMA (2,1,3) p=2 the ARIMA model takes into account two previous values to predict the current value, d=1 the Arima model applies one differentiating test to the data and q=3 the ARima model will use three lag of the residual value.

```
model_ARIMA=ARIMA(ts,order=(2,1,3))
results_ARIMA=model_ARIMA.fit()
predictions_ARIMA=pd.Series(results_ARIMA.fittedvalues,copy=True)
```

Figure 6. ARIMA Model

In Figure 6 is the stage for building an ARIMA model with source code using the ARIMA class of the module statsmodels.tsa.arima.model on the import library with the ts argument is the data time series that will be used to train the model and order=(2,1,3) shows the best order obtained from testing the model ARIMA with the auto_arima() function. Then results_ARIMA.fit() is used for training the AREMA model that is stored on the results_Arima object.

3.2. LSTM Implementation

In this section, the researchers will test the effectiveness of the LSTM model in making predictions on Solana's time-related data sets. In modeling testing, LSTM will use the training data ratio to divide the data into train and testing data, then batch size to determine the number of samples to be processed in each training iteration, in addition to determining the amount of epochs that will run the entire data set in the training process[15][16]. Here's a list of the training data ratio, batch size and epoch that will be used on the compilation for the LSTM model.

Table 1. Data Ratio, batch Size and Epoch for LSTM

Data Ratio	Batch Size	Epoch
60%:40%	64	50
		100
		500
	72	50
		100
		500
70%:40%	64	50
		100
		500
	72	50
		100
		500
80%:20%	64	50
		100
		500
	72	50
		100
		500
90%10%	64	50
		100
		500
	72	50
		100
		500

List Table 1 is a combination of the training data ratio, batch size and epoch in LSTM modeling testing. The table is used to organize experiments and find the best combinations of parameters to obtain the optimal model. Exercise data ratio is a method of dividing data into two parts, batch size is the number of data samples to be given to the LSTM model and epoch is the total number of experiments of the entire data set to be used throughout the modeling process[17]. Here's the source code for the implementation.

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 64, epochs=50)

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 64, epochs=100)

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 64, epochs=500)

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 72, epochs=50)

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 72, epochs=100)

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 72, epochs=500)
```

Figure 7. Source code trains the LSTM model with Batch Size and Epoch

On Figure 7 is a source code to train a model by receiving the training data ratio set on the source code of Figure 4.0 i.e. x_train as the input data used to train the model and y_trains as the output corresponding to x_trein. Then determine the size of the batch_size that will be processed in each training which is 64 and 72 and determine the epochs of 50,100,500 which shows how many times the entire data will be trained according to the number of epoches specified. After training on the process model, the next step is to use the LSTM model to make predictions of the results of the training data of the L STM model.

```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

Figure 8. Source code for develop LSTM prediction model

Figure 8 The source code builds the predictive model of LSTM. In Fig. 8, it is the source code for building the prediction of the model of lSTM using x_test as input data in the model testing. The code of this prediction model is used throughout the test on the list of combinations of the training data ratio, batch size and epoch that are set in Table 1

3.4. Evaluation

At the evaluation stage, the performance analysis of ARIMA and LSTM models will be performed. RMSE parameters will be used to measure the performance of models in predicting Solana data[18]. In addition, visualization of the predictions of ARIMA and LSTM models will be performed to understand how the model performs in predicting Solana data[19].

Results of evaluation and visualization of ARIMA

To measure the performance of the ARIMA model performed testing using RMSE parameters.

```
rmse = np.sqrt(np.mean(ts-predictions_ARIMA)**2)
rmse
```

Figure 9. Source code RMSE ARIMA

Figure 9 The source code of RMSE ARIMA is the RMSE source code in the ARIMA model with the calculation of the value between the actual value (ts) and the predictions value (predictions_ARIMA) and squared with the operator **2. Then take the average value of the entire difference using the function np.mean(). Finally take the square root of the mean value using the np.sqrt() function and store it in the rmse variable[20].

```
Output:
RMSE: 0.0192
```

Figure 10. Output of ARIMA RMSE

Output from the RMSE test resulting output for ARIMA order(2,1,3) is 0.0192, RMSE value for Arima order (2,1,3) appears to have a very small and accurate error rate in modeling the data. Because, the lower the value of RMSE, the better the model in making accurate predictions close to the actual value[21]. Here's the output of the ARIMA model visualization:

```
plt.figure(figsize=(10, 5))
plt.plot(ts, label='Aktual')
plt.plot(predictions_ARIMA, color='red', label='Prediksi ARIMA (2,1,3)')
plt.legend()
plt.title('Model')
plt.xlabel('Tanggal')
plt.ylabel('Nilai')
plt.show()
```

Figure 11. Source code visualization of ARIMA prediction results

From the visualization seen in Figure 12 it can be seen that the model prediction with RMSE 0.0192 shows good results. The red line represents the ARIMA order(2,1,3), and the blue line represent the actual data.

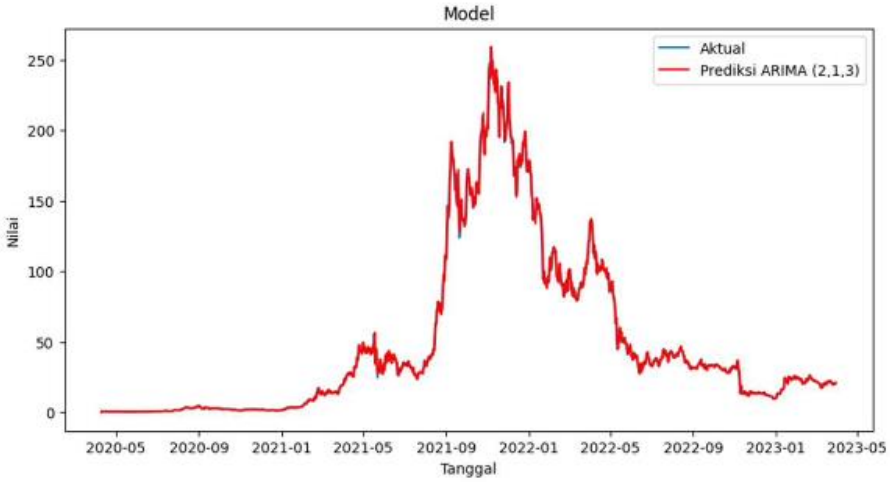

Figure 12. Visualization of ARIMA model prediction results

From the visualization seen in Figure 12 it can be seen that the model prediction with RMSE 0.0192 shows good results. The red line represents the ARIMA order(2,1,3), and the blue line represent the actual data[8].

3.5. Results of LSTM evaluation and visualization.

To measure the performance results of the LSTM model, testing is conducted using the RMSE parameter. Here is the source code for implementing the RMSE parameter in the LSTM model.

```
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse
```

Figure 13. Source code RMSE LSTM

In Figure 13, the source code for RMSE in the LSTM model is presented, which calculates the value between the predictions generated (predictions) and the actual values (y_test) by squaring the differences using the operator **2. Then, it finds the average of all the differences using the np.mean() function[22]. Finally, it takes the square root of that average using the np.sqrt() function and stores it in the variable rmse. After testing the LSTM model to predict Solana data, the researchers obtained evaluation results with varying RMSE parameter values[23]. In Table 2, the lowest RMSE parameter value is found in the 70:30% data ratio list, with a batch size of 64 and 500 epochs, resulting in an RMSE of 0.075. This indicates that the training data is very accurate for LSTM modeling. A low RMSE value shows that the model has a low prediction error rate, with the ability to produce predictions that closely match the actual values[24].

```
data = data.filter(['Close'])
train = data[:training_data_len]
validation = data[training_data_len:]
validation['Predictions'] = predictions
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date')
plt.ylabel('Close Price USD ($)')
plt.plot(train)
plt.plot(validation[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```

Figure 14. Source code for visualizing LSTM prediction results.

In Figure 14, the source code is presented to visualize the prediction results with an RMSE of 0.075 on the LSTM model, using a line plot for the validation data with the Close and Predictions columns as input. The aim is to compare the prediction performance with the actual values and to observe how well the model can learn patterns in the Solana data.

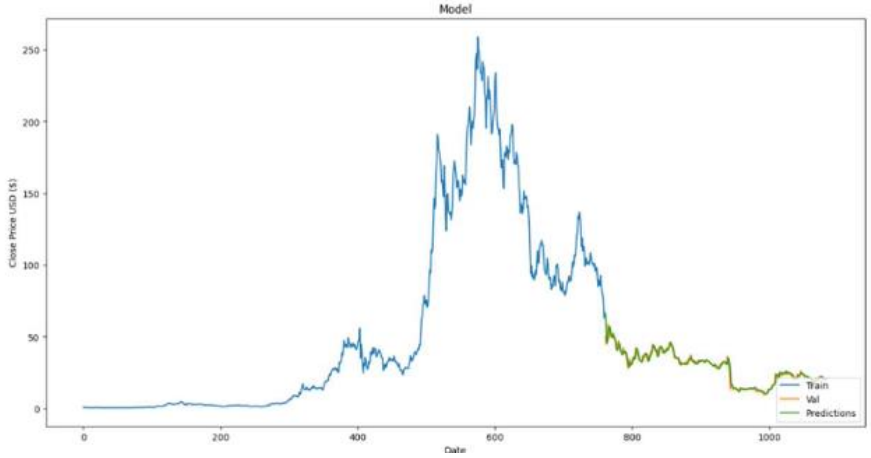

Figure 15. Visualization of the prediction results of the LSTM model.

From Figure 15, the LSTM model prediction with an RMSE of 0.075 shows good results. The blue line indicates the training data, the orange line indicates the test data, and the green line indicates the model's predictions.

Table 2 Results of RMSE Testing

LSTM			
Data Ratio	Batch Size	Epoch	RMSE
60% : 40%	64	50	0.923
		100	1.856
		500	3.847
	72	50	3.594
		100	2.317
		500	2.093
70% : 30%	64	50	2.827
		100	1.668
		500	0.075
	72	50	3.013
		100	2.502
		500	0.621
80% : 20%	64	50	0.609
		100	0.136
		500	0.993
	71	50	0.546
		100	0.321
		500	0.893
90% : 10%	64	50	2.072
		100	1.037
		500	0.937
	71	50	0.930
		100	0.598
		500	0.560

Table 3. Comparison table of models with RMSE values

Perbandingan Model		
	RMSE	Accuration
ARIMA (2,1,3)	0.019 (1,9%)	98.1 %
LSTM	0.075 (7.5%)	92.5 %

Table 3 shows a comparison of the ARIMA and LSTM models based on RMSE values. The analysis results indicate that the ARIMA model has an RMSE of 0.019 (1.9%) with an accuracy rate of 98.1%, while the LSTM model has an RMSE of 0.075 (7.5%) with an accuracy rate of 92.5%. This indicates that the ARIMA model is significantly more accurate in predicting data compared to the LSTM model[25].

4. CONCLUSION

Thus, the conclusion that can be drawn from the experiment conducted by the researchers is that in the case of using Solana timeseries data samples, the ARIMA method shows a higher accuracy level compared to the LSTM method, as evidenced by an ARIMA RMSE of 0.019 (1.9%) with an ARIMA order of (2,1,3) and

an accuracy level of 98.1%. Meanwhile, the lowest RMSE for LSTM from several tests was 0.075 (7.5%) with a training data ratio of 70:30, a batch size of 64, and 500 epochs, resulting in an accuracy level of 92.5%. In previous research conducted by (Latif et al., 2023), using a different dataset but applying the same methods, namely ARIMA and LSTM, the research results showed an ARIMA accuracy of (3,1,3) at 98.21%, while the LSTM accuracy was 99.73%. When comparing these research results with the current study, there is a reversal of those findings. Thus, it means that the ARIMA method does not always have a good level of accuracy, and conversely, the LSTM method does not always have a good level of accuracy either. The accuracy results from the ARIMA and LSTM methods are highly dependent on the characteristics of the dataset used and other factors that influence model performance, as seen from the comparison of studies. Although the test results show a significant difference between the ARIMA and LSTM methods, it is important to remember that. This conclusion only applies to the cases with the samples taken by the researcher and cannot be generally applied to all situations.

REFERENCES

[1] R. Auer and R. Böhme, "Central bank digital currency: the quest for minimally invasive technology," *BIS Work. Pap.*, no. 948, 2021, [Online]. Available: www.bis.org.

[2] I. Amsyar, E. Christopher, A. Dithi, A. N. Khan, and S. Maulana, "The challenge of cryptocurrency in the era of the digital revolution: A review of systematic literature," *Aptisi Trans. Technopreneursh.*, vol. 2, no. 2, pp. 153–159, 2020.

[3] X. Li, X. Wang, T. Kong, J. Zheng, and M. Luo, "From bitcoin to solana–innovating blockchain towards enterprise applications," in *International Conference on Blockchain*, 2021, pp. 74–100.

[4] U. A. Haruna, "Developing and Implementing a Cryptocurrency For Gombe State University," 2024.

[5] G. Minotti, "Cryptocurrencies Price Prediction using LSTM Neural Network model," 2023.

[6] S. M. Shariff, "Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) network models for forecasting energy consumptions," *Eur. J. Electr. Eng. Comput. Sci.*, vol. 6, no. 3, pp. 7–10, 2022.

[7] H. Mou and J. Yu, "CNN-LSTM Prediction Method for Blood Pressure Based on Pulse Wave," *Electronics*, vol. 10, no. 14. 2021, doi: 10.3390/electronics10141664.

[8] F. M. Siemers, C. Feldmann, and J. Bajorath, "Minimal data requirements for accurate compound activity prediction using machine learning methods of different complexity," *Cell Reports Phys. Sci.*, vol. 3, no. 11, 2022.

[9] A. Lawi, H. Mesra, and S. Amir, "Implementation of Long Short-Term Memory and Gated Recurrent Units on grouped time-series data to predict stock prices accurately," *J. Big Data*, vol. 9, no. 1, p. 89, 2022.

[10] A. S. Girsang, "Cryptocurrency Price Prediction Based Social Network Sentiment Analysis Using LSTM-GRU and FinBERT," *IEEE Access*, 2023.

[11] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence*, 2019, pp. 49–55.

[12] K. Albeladi, B. Zafar, and A. Mueen, "Time Series Forecasting using LSTM and ARIMA," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 1, pp. 313–320, 2023.

[13] G. Abdoli, "Comparing the prediction accuracy of LSTM and ARIMA models for time-series with permanent fluctuation," *Periódico do Núcleo Estud. e Pesqui. sobre Gênero e DireitovCentro Ciências Jurídicas-Universidade Fed. da Paraíba*, vol. 9, 2020.

[14] O. O. Okesola, "Data Mining Methodolgy and its application," 2021.

[15] E. Tuncer and E. D. Bolat, "Classification of epileptic seizures from electroencephalogram (EEG) data using bidirectional short-term memory (Bi-LSTM) network architecture," *Biomed. Signal Process. Control*, vol. 73, p. 103462, 2022.

[16] W. Zhang *et al.*, "LSTM-based analysis of industrial IoT equipment," *IEEE Access*, vol. 6, pp. 23551–23560, 2018.

[17] O. Barut, L. Zhou, and Y. Luo, "Multitask LSTM model for human activity recognition and intensity estimation using wearable sensor data," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8760–8768, 2020.

[18] E. Persson, "Forecasting Efficiency in Cryptocurrency Markets: A machine learning case study." 2022.

[19] T. R. Noviandy *et al.*, "Deep learning-based bitcoin price forecasting using neural prophet," *Ekon. J. Econ.*, vol. 1, no. 1, pp. 19–25, 2023.

[20] J. I. Piovani, M. E. Rausky, and J. A. Santos, " Qualitative Research in Issues of the American Journal of Sociology during the Hegemony of The Chicago School ," *Méthod(e)s African Rev. Soc. Sci.*

Methodol., vol. 1, no. 1–2, pp. 111–123, 2015, doi: 10.1080/23754745.2015.1017279.

[21] Ü. Ağbulut, A. E. Gürel, and Y. Biçen, "Prediction of daily global solar radiation using different machine learning algorithms: Evaluation and comparison," *Renew. Sustain. Energy Rev.*, vol. 135, p. 110114, 2021.

[22] J. M. Oliva-Lozano, I. Martín-Fuentes, V. Fortes, and J. M. Muyor, "Differences in worst-case scenarios calculated by fixed length and rolling average methods in professional soccer match-play," *Biol. Sport*, vol. 38, no. 3, p. 325, 2021.

[23] Y. A. Ünvan and C. Ergenc, "A COMPREHENSIVE COMPARATIVE STUDY OF MACHINE LEARNING MODELS FOR PREDICTING CRYPTOCURRENCY," *Facta Univ. Ser. Electron. Energ.*, vol. 37, no. 1, pp. 211–227, 2024.

[24] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *Peerj Comput. Sci.*, vol. 7, p. e623, 2021.

[25] M. Elsaraiti and A. Merabet, "A comparative analysis of the arima and lstm predictive models and their effectiveness for predicting wind speed," *Energies*, vol. 14, no. 20, p. 6782, 2021.