Harnessing Advanced Algorithms for Cryptocurrency Price Prediction

Bui Van Thai[1], Nguyen Binh Nguyen[2], and Nguyen The Vinh[3]

[1]21522577, Faculty of Information Systems, University of Information Technology, (e-mail: 21522577@gm.uit.edu.vn)
[2]21522391, Faculty of Information Systems, University of Information Technology, (e-mail: 21522391@gm.uit.edu.vn)
[3]21522794, Faculty of Information Systems, University of Information Technology, (e-mail: 21522794@gm.uit.edu.vn)

ABSTRACT

Unlocking the potential of accurate Bitcoin price predictions, this study introduces eight powerful algorithms - SES, GARCH, MLP, Linear Regression, ARIMA, RNN, GRU, and LSTM - that harness the strengths of exponential smoothing, volatility modeling, regression analysis, time series forecasting, and various neural network architectures to forecast the future price movements of the cryptocurrency. These algorithms were chosen due to their respective strengths in capturing short-term trends (SES), volatility dynamics (GARCH), linear relationships (Linear Regression), autoregressive patterns (ARIMA), and complex nonlinear patterns (MLP, RNN, GRU, LSTM). By combining these algorithms, we aim to provide a comprehensive and accurate framework for forecasting Bitcoin prices, offering valuable insights for investors in the cryptocurrency market.

Keywords: Bitcoin price prediction, cryptocurrency forecasting, SES, GARCH, Linear Regression, ARIMA, RNN, GRU, LSTM, MLP, time series analysis, neural networks, financial modeling.

I. INTRODUCTION

The prediction of cryptocurrency prices has garnered significant attention in recent years due to the increasing popularity and market impact of digital currencies. Among the various cryptocurrencies, Bitcoin stands out as the most prominent and widely traded one. The ability to accurately forecast Bitcoin prices has the potential to provide valuable insights for investors, traders, and policymakers in understanding the dynamics of this volatile market.

In this study, we apply eight forecasting algorithms, namely Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Single Exponential Smoothing (SES), Multilayer Perceptron (MLP), Linear Regression, AutoRegressive Integrated Moving Average (ARIMA), Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), to predict the price of Bitcoin. These algorithms have been widely used in financial forecasting and have demonstrated effectiveness in capturing the time-varying characteristics and nonlinear patterns present in financial data.

The GARCH model, proposed by Bollerslev (1986) [1], is a popular choice for modeling the volatility of financial time series. It takes into account the conditional heteroskedasticity, allowing for the estimation of time-varying volatility. The SES method, introduced by Brown (1956) [2], is a simple and widely used exponential smoothing technique that assigns exponentially decreasing weights to past observations. The MLP algorithm, a type of artificial neural network, has gained attention for its ability to capture complex relationships and nonlinearity in data [7]. Linear Regression is a fundamental statistical method used to model the relationship between a dependent variable and one or more independent variables [10]. ARIMA, introduced by Box and Jenkins [8], is a widely used statistical method for time series forecasting that combines autoregression, differencing, and moving averages. RNN, GRU, and LSTM are advanced neural network architectures specifically designed to handle sequential data and capture long-term dependencies [9].

To evaluate the performance of these forecasting algorithms, we utilize a comprehensive dataset of Bitcoin prices spanning multiple years. The dataset includes historical price data, as well as other relevant features such as trading volume, market capitalization, and sentiment analysis indicators. By incorporating these additional factors, we aim to enhance the predictive accuracy of the models and provide a more comprehensive analysis of Bitcoin price movements.

The primary objectives of this study are twofold. First, we aim to compare the forecasting performance of the GARCH, SES, MLP, Linear Regression, ARIMA, RNN, GRU, and LSTM algorithms in predicting Bitcoin prices. By assessing the accuracy and robustness of these models, we can gain insights into their suitability for cryptocurrency price prediction. Second, we seek to identify the key factors that contribute to the prediction accuracy, shedding light on the underlying drivers of Bitcoin price dynamics.

II. RELATED WORKS

In recent years, there has been a substantial amount of research dedicated to predicting stock prices using various machine learning and statistical models.

Linear Regression is a fundamental and widely used statistical technique in financial forecasting due to its simplicity and

interpretability. Kumar and Thenmozhi (2006) [10] applied this model to predict stock prices of major Indian companies, demonstrating its ability to capture market trends. However, Patel et al. (2015) showed that more complex models like SVM and ANN often outperform Linear Regression in predicting stock prices, especially in capturing complex patterns.
RNNs (Recurrent Neural Networks) are specialized neural networks designed for sequential data processing, particularly suitable for time series forecasting. Zhang et al. (2019) [11] employed RNNs to forecast Bitcoin prices, showcasing their ability to capture complex and nonlinear time series patterns .
GRU (Gated Recurrent Unit) networks, similar to LSTMs but with a simpler architecture, have been applied to forecast stock prices effectively. Chung et al. (2014) [12] empirically evaluated GRUs and found them comparable to LSTMs in forecasting tasks while requiring fewer computational resources.
ARIMA (The AutoRegressive Integrated Moving Average) model is a powerful statistical technique widely used for time series forecasting. Tsai et al. (2010) [13] utilized ARIMA to forecast stock prices and stock market indices, highlighting its effectiveness in capturing linear time series patterns .
GARCH models have been used to catch the price volatility of different commodities and assets. It was proved that GARCH model is a suitable approach to detect the volatility along the meat supply chain [3]. The GARCH model has also been used for forecasting the volatility of the Islamic stock index [4], Bitcoin [14], and bond yield [6].
SES (Single Exponential Smoothing) is a simple yet effective time series forecasting method. Ostertagova, Eva & Ostertag, Oskar [30] utilized SES to predict primary production of electricity Slovakia for the year 2010. Their results showed that SES performed well in capturing the trend and short-term fluctuations.
Long Short-Term Memory (LSTM) networks, a type of RNN, are designed to address the vanishing gradient problem and capture long-term dependencies. Fischer and Krauss (2018) [15] highlighted LSTM's superiority over traditional models and other machine learning algorithms in predicting stock prices, due to its ability to model both short-term fluctuations and long-term trends.
MLP (Multilayer Perceptron) is a type of artificial neural network that has been widely used in financial forecasting tasks. Zhang et al. (2019) [7] employed MLP models to predict Bitcoin prices and evaluated their performance against other machine learning algorithms. The results demonstrated that MLP models achieved high accuracy in forecasting Bitcoin prices, especially in capturing complex patterns and nonlinear relationships.

III. MATERIALS

A. DATASET

Historical price data for Binance Coin (BNB), Dogecoin, and Tron from 01/03/2019 to 01/06/2024 will be applied. The data contains column such as Date, Price, Open, High, Low, Vol., Change. As the goal is to forecast close prices, only data relating to column "Price" (USD) will be processed.

B. DESCRIPTIVE STATISTICS

TABLE 1. BNB, DOGECOIN, TRON's Descriptive Statistics

	BNB	DOGECOIN	TRON
Count	1920	1920	1920
Mean	229.552	0.083	0.059
Var	33996.748	0.009	0.001
Std	184.382	0.093	0.034
Min	9.25	0.001585	0.008409
25%	27.03	0.0029	0.025
50%	247.55	0.0672	0.062
75%	332.11	0.119	0.080
Max	676.56	0.68688	0.1639

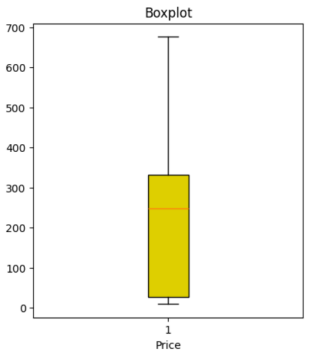

FIGURE 1. BNB coin price's boxplot

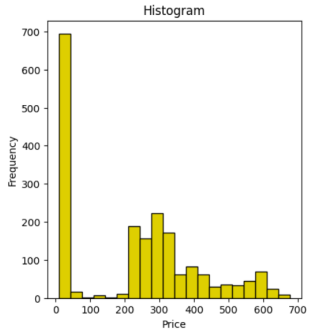

FIGURE 2. BNB coin price's histogram

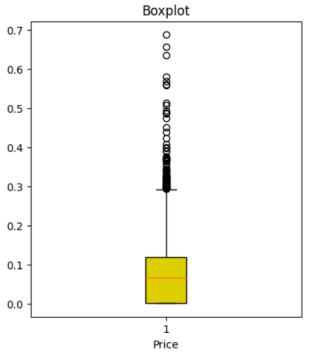

FIGURE 3. Dogecoin coin price's boxplot

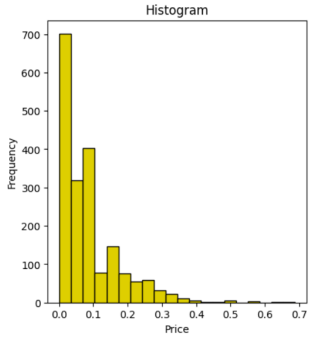

FIGURE 4. Dogecoin coin price's histogram

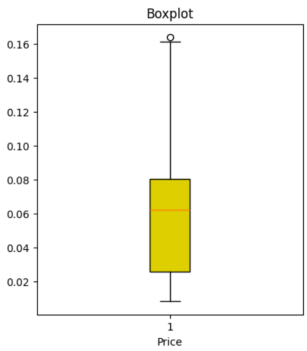

FIGURE 5. Tron coin price's boxplot

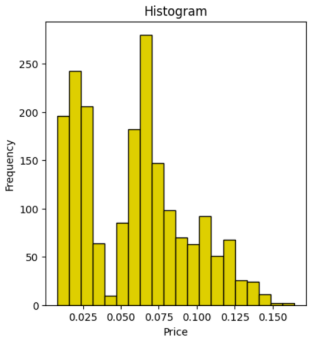

FIGURE 6. Tron coin price's histogram

IV. METHODOLOGY

A. GARCH

The GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is a crucial tool in finance for forecasting the volatility of time series. This model allows for time-varying volatility, providing a more accurate description of fluctuations in financial data. The formula of the GARCH(p,q) model is:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$$

Where:

- σ_t^2 represents the conditional variance at time t.
- ω is the constant term or intercept.
- α_i $(i = 1, \ldots, p)$ are the coefficients associated with the past squared error terms (ARCH terms).
- β_j $(j = 1, \ldots, q)$ are the coefficients associated with the past conditional variances (GARCH terms).
- ε_{t-i}^2 are the past error terms, calculated as:

$$\varepsilon_{t-i} = y_{t-i} - \mu$$

  where y_{t-i} is the observed value at time $t - i$ and μ is the mean of the time series.
- σ_{t-j}^2 are the past conditional variances, calculated recursively using the GARCH model.

In order for $\sigma_t^2 > 0$, it is assumed that $\omega > 0$ and the coefficients α_i $(i = 1, \ldots, p)$ and β_j $(j = 1, \ldots, q)$ are all non-negative. When $q = 0$, the GARCH(p,q) model reduces to the ARCH(p) model.

Usually, the GARCH(1,1) model,

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2,$$

with only three parameters in the conditional variance equation, is adequate to obtain a good model fit for daily asset returns. Indeed, Hansen and Lund (2004) provided compelling evidence that it is difficult to find a volatility model that outperforms the simple GARCH(1,1). Hence, for many purposes, the GARCH(1,1) model is the de facto volatility model of choice for daily returns [1].

Given its effectiveness in modeling daily asset returns, the GARCH(1,1) model is also suitable for forecasting the volatility of cryptocurrencies such as Bitcoin. Cryptocurrencies often exhibit high volatility, and the GARCH(1,1) model's ability to adapt to changing volatility over time makes it a practical choice for this application.

B. RECURRENT NEURAL NETWORKS (RNNS)

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for sequence modeling, commonly used in tasks such as time series prediction, natural language processing, and speech recognition. Unlike traditional feedforward neural networks, RNNs have connections that form directed cycles, allowing them to maintain a form of memory of previous inputs through their internal state. The general form of an RNN can be described by the following equations:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

Where:

- h_t represents the hidden state at time t.
- x_t is the input at time t.
- y_t is the output at time t.
- W_h is the weight matrix for the input to hidden state.
- U_h is the weight matrix for the hidden state to hidden state (recurrent connection).
- W_y is the weight matrix for the hidden state to output.
- b_h and b_y are the bias vectors for the hidden state and output, respectively.
- σ_h and σ_y are the activation functions for the hidden state and output, respectively.
- h_{t-1} is the hidden state at the previous time step $t - 1$.

[16], [18], [19]

C. ARIMA

ARIMA (Autoregressive Integrated Moving Average) is a time series forecasting model that combines autoregression (AR), differencing (I), and moving average (MA) components. It is commonly used to predict future values based on past observations.

- AR (Autoregressive): Autocorrelation property of the data with itself at previous time points. The ARIMA model uses parameter ρ to determine the number of previous values to compute.
- I (Integrated): The integration step is to make the data easily predictable by removing trends in the data. This is often done by taking the difference between consecutive data points. The ARIMA model uses parameter $\bar{d}$ to determine the number of differencing steps.
- MA (Moving Average): Used to model random fluctuations in the data. The ARIMA model uses parameter $\bar{q}$ to determine the number of moving average values [8], [9], [13].

D. GATED RECURRENT UNIT(GRU)

GRU is a type of recurrent neural network (RNN) designed to handle and predict sequence data, such as time series data. GRU was first introduced by K. Cho et al. in 2014 as an improvement over Long Short-Term Memory (LSTM), another RNN model, with the aim of simplifying the architecture and reducing the number of parameters. GRU consists of two main gates:

- Update Gate: Determines how much of the previous state needs to be retained.
- Reset Gate: Determines how much of the previous state needs to be forgotten.

The computational formulas for GRU are:

- Update Gates:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

- Reset Gate:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

- Candidate State:

$$\tilde{h} = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

- New State:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Where:

- x_t: is the input at time t.
- h_t: is the hidden state at time t.
- W_z, W_r, W: are weight matrices.
- σ: is the sigmoid function. [12], [20], [21]

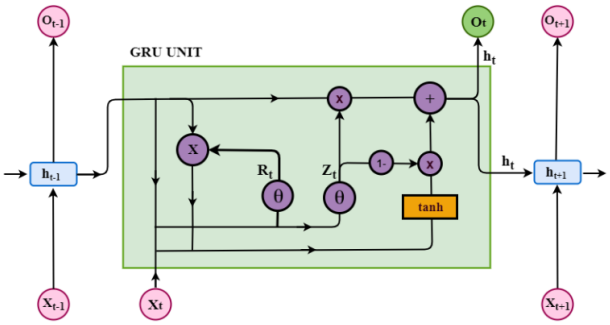

FIGURE 7. The Architecture of basic Gated Recurrent Unit (GRU).

E. MULTILAYER PERCEPTRON(MLP)

MLP is a type of neural network with a simple structure and forms the basis of many other neural network architectures. An MLP consists of at least three layers: an input layer, one or more hidden layers, and an output layer. Each layer in an MLP is a set of perceptrons, and each perceptron is a computational unit that performs matrix multiplication and applies an activation function.The structure of a typical MLP includes:

- Input Layer: Receives input data.
- Hidden Layer(s): Processes and extracts features from the input data through weights and activation functions.
- Output Layer: Produces the final prediction result.

Each perceptron in a layer operates according to the formula:

$$y = \phi(W \cdot x + b)$$

Where:

- x is the input.
- W is the weight vector.
- b is the bias.
- ϕ is the activation function (commonly sigmoid, tanh, or ReLU). [16], [22], [23]

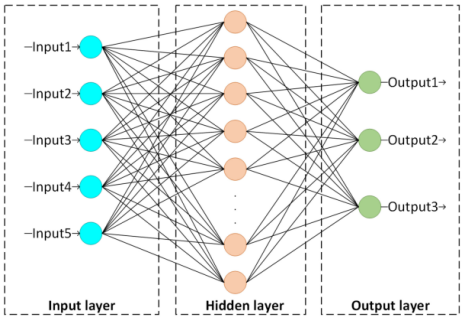

FIGURE 8. A basic MLP model framework.

F. LINEAR REGRESSION(LR)

Regression analysis is a tool for building mathematical and statistical models that characterize relationships between a dependent variable and one or more independent, or explanatory, variables, all of which are numerical. This statistical technique is used to find an equation that best predicts the y variable as a linear function of the x variables. A multiple linear regression model has the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

Where:

• Y is the dependent variable (Target Variable).

• $X_1, X_2, \ldots, X_k$ are the independent (explanatory) variables.

• β_0 is the intercept term.

• $\beta_1, ..., \beta_k$ are the regression coefficients for the independent variables.

• ε is the error term. [**?**], [**?**]

G. SIMPLE EXPONENTIAL SMOOTHING(SES)

Exponential smoothing is a forecasting method for univariate time series data. This method produces forecasts that are weighted averages of past observations where the weights of older observations exponentially decrease. The simplest of the exponentially smoothing methods is naturally called simple exponential smoothing (SES). Use simple exponential smoothing for univariate time series data that do not have a trend or seasonal cycle. Forecasts are calculated using weighted averages, where the weights decrease exponentially

as observations come from further in the past — the smallest weights are associated with the oldest observations:

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$$

Where:

- $\hat{y}_{t+1|t}$ the forecasted value at time based on $t+1$ information up to time t.
- $\hat{y}$ the observed value at time t.
- $\hat{y}_{t|t-1}$ the forecasted value at time t based on information up to time $t-1$.
- α the smoothing factor, which is a value between 0 and 1 that determines the influence of the current observation on the forecast. [26], [27]

H. LONG SHORT-TERM MEMORY(LSTM)

LSTM (Long Short-Term Memory) is a specialized type of Recurrent Neural Network (RNN) designed to handle long-term dependencies in sequential data. LSTM units consist of a cell state and three gates (input, forget, and output) that regulate the flow of information. The cell state maintains long-term information, while the gates control what information is added, retained, or output at each time step. This structure allows LSTMs to effectively mitigate the vanishing gradient problem common in traditional RNNs, making them well-suited for applications like speech recognition, time series prediction, and natural language processing. A basic LSTM unit consists of the following main components:

- Cell State: Carries information across time steps, the cell state has the ability to maintain and regulate information over long periods.
- Forget Gate: Decides what information from the previous cell state should be forgotten.
- Input Gate: Decides what new information will be added to the cell state.
- Output Gate: Determines which part of the current cell state will be output.
- Cell State Update: Combines old and new information to update the cell state.

The computational formulas for LSTM are:

- Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- Output Gate:

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

- Cell Update:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Where:

- f_t: Forget gate activation vector at time step t
- i_t: Input gate activation vector at time step t
- o_t: Output gate activation vector at time step t
- $\tilde{C}_t$: Candidate cell state at time step t
- C_t: Cell state vector at time step t
- h_t: Hidden state vector at time step t
- x_t: Input vector at time step t
- W_f, W_i, W_C, W_o: Weight matrices for the forget, input, cell, and output gates, respectively
- b_f, b_i, b_C, b_o: Bias vectors for the forget, input, cell, and output gates, respectively
- σ: Sigmoid activation function
- $\tanh$: Hyperbolic tangent activation function. [**?**], [**?**]

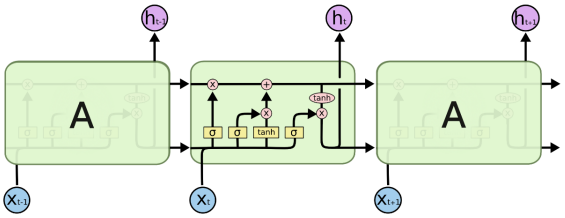

FIGURE 9. The repeating module in an LSTM.

V. RESULT

A. EVALUATION METHODS

Mean Percentage Absolute Error (MAPE): is the average percentage error in a set of predicted values.

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

Where:

• n is the number of observations in the dataset.
• y_i is the true value.
• $\hat{y}_i$ is the predicted value.

Root Mean Squared Error (RMSE): is the square root of the average value of the squared error in a set of predicted values.

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

Where:

• n is the number of observations in the dataset.
• y_i is the true value.
• $\hat{y}_i$ is the predicted value. **Mean Absolute Error** (MSE):is a commonly used performance metric in machine learning and statistics. It measures the average of the squared differences between the predicted values and the true/actual values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\log(1+\hat{y}_i) - \log(\log(1+y_i))\right)^2$$

Where:

• n is the number of observations in the dataset.

- y_i is the true value.
- $\hat{y}_i$ is the predicted value.

B. TRON DATASET

TRON Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSE
LR	7:3	0.0365	27.53	0.0013
	8:2	**0.0365**	**27.53**	**0.0013**
	9:1	0.0200	13.42	0.0004
RNN	7:3	0.003	2.82	6.96
	8:2	0.0024	1.8	5.59
	9:1	**0.0026**	**1.74**	**6.87**
GRU	7:3	0.002	1.681	0.000004
	8:2	0.002	1.509	0.000004
	9:1	**0.002**	**1.355**	**0.000005**
ARIMA	7:3	0.035	26.805	0.001
	8:2	0.036	27.547	0.001
	9:1	**0.02**	**13.423**	**0.0004**
GARCH	**7:3**	**0.039**	**30.93**	**0.002**
	8:2	0.063	57.03	0.004
	9:1	0.079	63.97	0.006
SES	7:3	0.0365	27.53	0.0013
	8:2	0.0365	27.53	0.0013
	9:1	**0.0200**	**13.42**	**0.0004**
LSTM	7:3	0.0038	3.29	0.000014
	8:2	**0.0029**	**2.00**	**0.000008**
	9:1	0.0034	2.30	0.000012
MLP	7:3	0.002	0.017	0.000004
	8:2	**0.002**	**0.014**	**0.000004**
	9:1	0.003	0.018	0.000007

TABLE 2. TRON Dataset's Evaluation

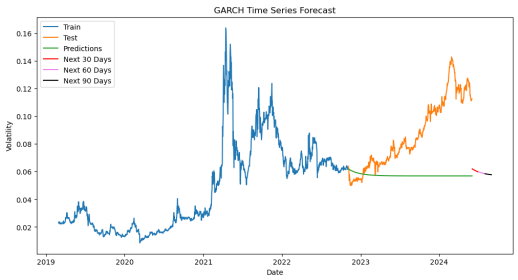

FIGURE 10. GARCH model's result with 7:3 splitting proportion

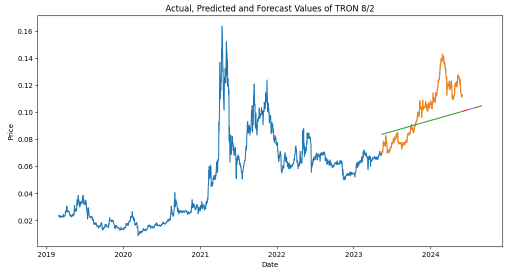

FIGURE 11. LR model's result with 8:2 splitting proportion

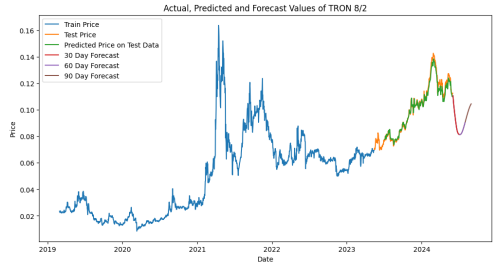

FIGURE 12. LSTM model's result with 8:2 splitting proportion

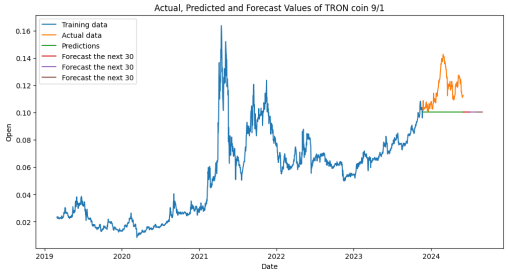

FIGURE 13. SES model's result with 9:1 splitting proportion

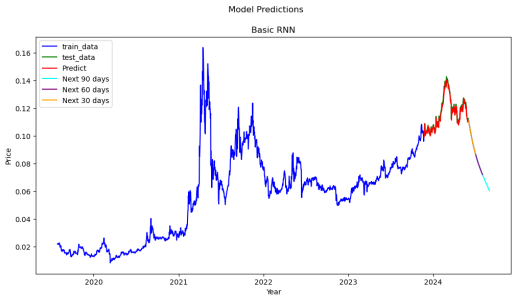

FIGURE 14. RNN model's result with 9:1 splitting proportion

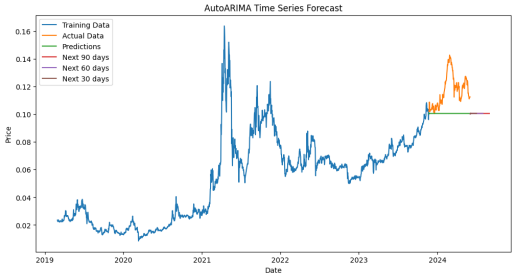

FIGURE 15. ARIMA model's result with 9:1 splitting proportion

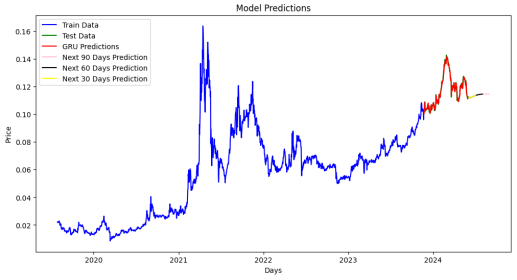

FIGURE 16. GRU model's result with 9:1 splitting proportion

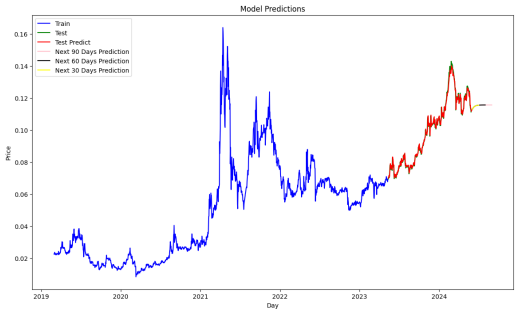

FIGURE 17. MLP model's result with 8:2 splitting proportion

C. BNB DATASET

BNB Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSE
LR	7:3	250.279	86.40	62639.603
	8:2	196.348	69.72	38552.750
	9:1	**131.961**	**32.73**	**17413.947**
RNN	7:3	20.28	5.19	411.22
	8:2	**12.02**	**1.9**	**144.498**
	9:1	16.68	2.95	278.129
GRU	**7:3**	**13.270**	**2.160**	**176.090**
	8:2	14.592	2.425	212.925
	9:1	15.475	2.175	239.472
ARIMA	**7:3**	**116.788**	**26.463**	**13639.48**
	8:2	141.783	30.427	20102.409
	9:1	242.385	38.723	58750.813
GARCH	**7:3**	**116.42**	**26.75**	**13553.28**
	8:2	141.23	30.42	19944.69
	9:1	239.61	38.07	57411.59
SES	**7:3**	**116.764**	**26.36**	**13633.898**
	8:2	141.8623	30.32	20124.9178
	9:1	240.6164	38.27	57896.2456
LSTM	**7:3**	**13.473**	**2.38**	**181.539**
	8:2	16.371	2.91	268.015
	9:1	17.907	2.40	320.663
MLP	**7:3**	**10.894**	**0.020**	**118.682**
	8:2	11.968	0.020	143.245
	9:1	15.366	0.022	236.119

TABLE 3. BNB Dataset's Evaluation

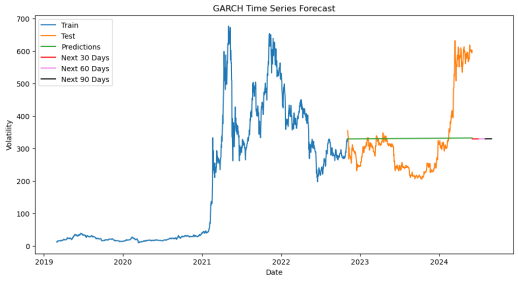

FIGURE 18. GARCH model's result with 7:3 splitting proportion

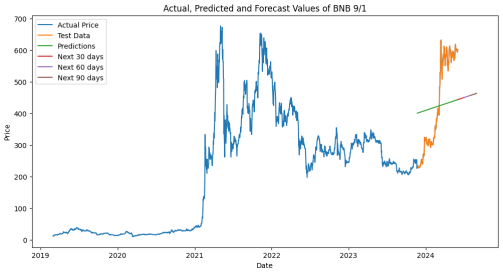

FIGURE 19. LR model's result with 9:1 splitting proportion

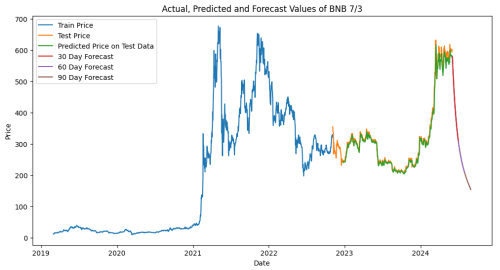

FIGURE 20. LSTM model's result with 7:3 splitting proportion

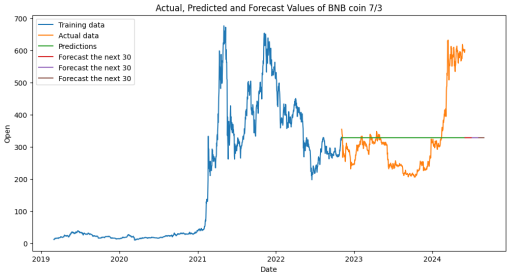

FIGURE 21. SES model's result with 7:3 splitting proportion

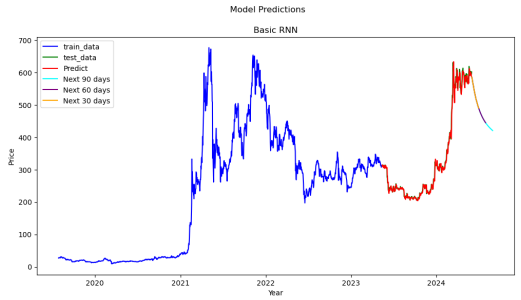

FIGURE 22. RNN model's result with 8:2 splitting proportion

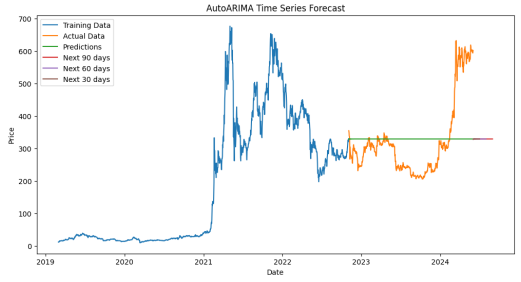

FIGURE 23. ARIMA model's result with 7:3 splitting proportion

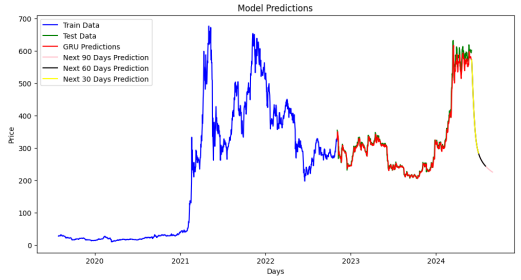

FIGURE 24. GRU model's result with 7:3 splitting proportion

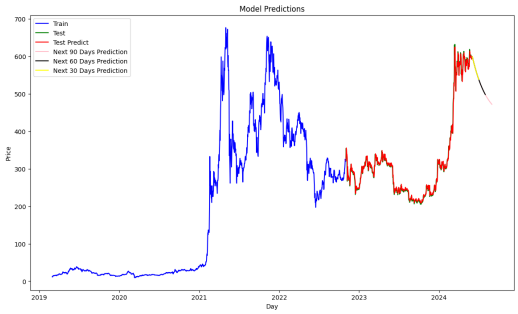

FIGURE 25. MLP model's result with 7:3 splitting proportion

D. DOGECOIN DATASET

DOGECOIN Coin Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSE
LR	7:3	0.1336	163.61	0.0179
	8:2	0.0872	108.02	0.0076
	9:1	**0.0428**	**37.72**	**0.0018**
RNN	7:3	0.008	7.37	5.82
	8:2	**0.0096**	**9.547**	**9.144**
	9:1	0.01	7.07	0.0001
GRU	7:3	0.006	4.060	0.000033
	8:2	**0.006**	**3.083**	**0.000035**
	9:1	0.008	3.777	0.000064
ARIMA	7:3	0.0198	17.739	0.0003
	8:2	0.013	15.159	0.0001
	9:1	**0.008**	**3.881**	**0.000069**
GARCH	7:3	0.05	66.25	0.003
	8:2	0.047	56.21	0.002
	9:1	**0.037**	**25.25**	**0.001**
SES	7:3	0.0469	56.26	0.0022
	8:2	**0.0471**	**22.39**	**0.0022**
	9:1	0.0635	32.07	0.0040
LSTM	**7:3**	**0.0064**	**4.32**	**0.000040**
	8:2	0.0071	4.47	0.000051
	9:1	0.0099	4.88	0.000098
MLP	**7:3**	**0.005**	**0.031**	**0.000028**
	8:2	0.006	0.037	0.000037
	9:1	0.009	0.057	0.000088

TABLE 4. DogeCoin Dataset's Evaluation

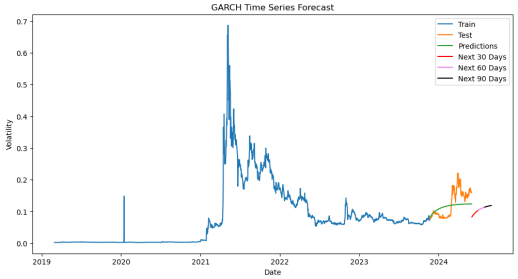

FIGURE 26. GARCH model's result with 9:1 splitting proportion

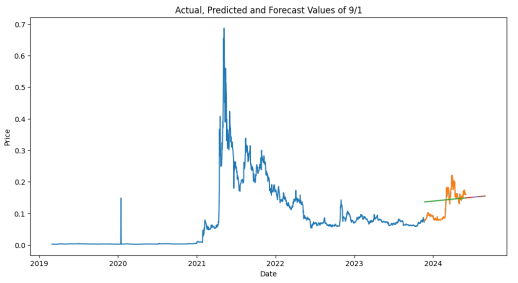

FIGURE 27. LR model's result with 9:1 splitting proportion

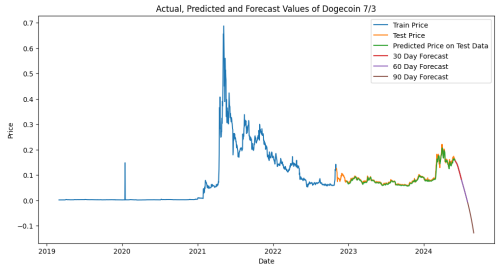

FIGURE 28. LSTM model's result with 7:3 splitting proportion

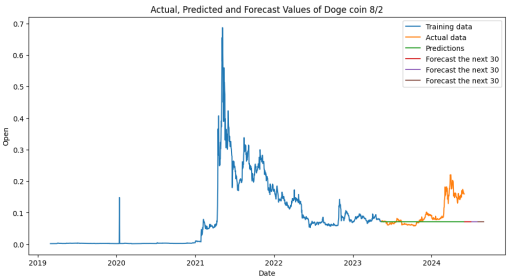

FIGURE 29. SES model's result with 8:2 splitting proportion

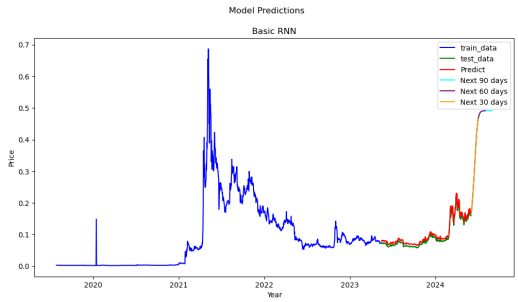

FIGURE 30. RNN model's result with 8:2 splitting proportion

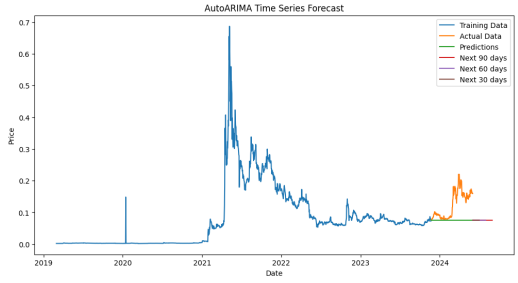

FIGURE 31. ARIMA model's result with 9:1 splitting proportion

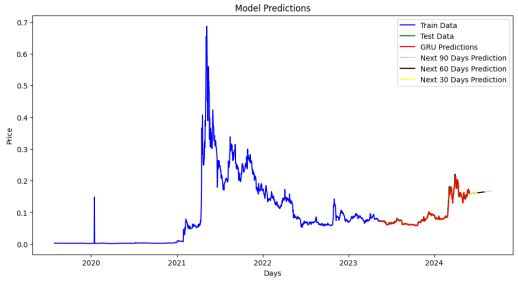

FIGURE 32. GRU model's result with 8:2 splitting proportion

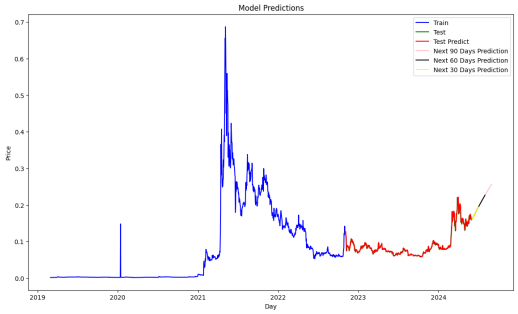

FIGURE 33. MLP model's result with 7:3 splitting proportion

VI. CONCLUSION

A. SUMMARY

In the pursuit of predicting cryptocurrency prices, various methodologies have been explored, spanning from conventional statistical models to sophisticated machine learning algorithms. Among these, MLP (Multi-Layer Perceptron) and GRU (Gated Recurrent Unit) algorithms generally outperform other methods such as SES (Simple Exponential Smoothing), Linear Regression, RNN (Recurrent Neural Network), ARIMA (AutoRegressive Integrated Moving Average), LSTM (Long Short-Term Memory) and GARCH (Generalized Autoregressive Conditional Heteroskedasticity). MLP and GRU are highly effective due to their capability to understand and adapt to intricate temporal patterns in time-series data. They excel in handling non-linear trends, long-term relationships, and erratic fluctuations commonly seen in cryptocurrency markets. LSTM networks are particularly adept at retaining information across extended sequences, which is crucial for modeling price trends that evolve over time amid varying levels of market volatility.

In contrast, simpler methods like SES and Linear Regression may struggle with the inherent complexities and nonlinearities inherent in cryptocurrency price data. While RNNs offer more flexibility, they can encounter challenges such as gradient vanishing and difficulty in learning dependencies over extended periods. ARIMA models are suitable for stationary series but may not adequately capture the dynamic and irregular patterns of cryptocurrency prices. MLPs are robust in learning complex relationships but may require extensive tuning and regularization for effective performance on time-series data. GARCH models are suitable for volatility modeling but do not directly forecast price levels.

In conclusion, GRU and MLP models stand out in cryptocurrency price prediction due to their ability to handle complex data patterns and effectively capture long-term dependencies. These models demonstrate their suitability for achieving accurate and reliable forecasts in the specialized domain of cryptocurrency price prediction.

B. FUTURE CONSIDERATIONS

In our future research endeavors, it is paramount to prioritize further refining the models previously discussed. This optimization initiative should focus on several key areas

• Focus on interpretability and explainability, as models grow in complexity, ensuring they remain interpretable is crucial for stakeholders to trust and act upon forecasting insights. Techniques such as model explainability tools and transparent methodologies should be integrated to enhance trustworthiness and usability.

• Enhancing model accuracy , while the aforementioned algorithms have shown promise in predicting cryptocurrency prices, there remains a pressing need to elevate their accuracy for more precise forecasting outcomes. This involves not only improving existing algorithms but also exploring advanced techniques such as feature engineering and hyperparameter tuning to maximize predictive performance.

• Innovating new forecasting models, the landscape of forecasting is continuously evolving, necessitating ongoing exploration of cutting-edge models and methodologies. This includes researching novel forecasting techniques that integrate alternative data sources (e.g., sentiment analysis from social media or macroeconomic indicators) to capture nuanced market behaviors more effectively.

By continually exploring new avenues for improvement, integrating cutting-edge technologies, and expanding the scope of data inputs, we aim to continuously optimize forecasting models. This proactive approach not only enhances their precision and reliability in predicting cryptocurrency prices but also equips us to navigate the dynamic landscape of financial markets with greater confidence and foresight.

ACKNOWLEDGMENT

Above all, we deeply appreciate the guidance and expertise provided by **Assoc. Prof. Dr. Nguyen Dinh Thuan** and **Mr. Nguyen Minh Nhut** throughout our research journey. Their mentorship and invaluable feedback have played a pivotal role in shaping the direction and quality of this study. Their extensive knowledge, critical insights, and meticulous attention to detail have significantly contributed to the achievements of this research.

We are sincerely grateful to our mentors for their unwavering support and contributions, without which this research would not have been possible. We extend heartfelt thanks to everyone involved for their invaluable assistance, encouragement, and belief in our work.

REFERENCES

[1] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307-327.

[2] Brown, R. G. (1956). Exponential smoothing for predicting demand. Management Science, 6(3), 324-342.
[3] Ben Abdallah, M., Fekete Farkas, M., & Lakner, Z. (2020). Analysis of meat price volatility and volatility spillovers in Finland. Agricultural Economics, 66, 84–91.
[4] Saleem, A., Bárczi, J., & Sági, J. (2021). COVID-19 and Islamic Stock Index: Evidence of Market Behavior and Volatility Persistence. Journal of Risk and Financial Management, 14, 389.
[5] Hansen, P. R., & Lund, T. (2004). A Forecast Comparison of Volatility Models: Does Anything Beat a GARCH(1,1)?. Journal of Applied Econometrics, 20(7), 873-889.
[6] Kim, J.-M., Kim, D. H., & Jung, H. (2021). Estimating yield spreads volatility using GARCH-type models. The North American Journal of Economics and Finance, 57, 101396.
[7] Zhang, H., et al. (2019). Multilayer Perceptron Models for Bitcoin Price Forecasting. Neural Computing and Applications, 32(7), 1987-2000.
[8] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control (5th ed.). John Wiley & Sons.
[9] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts.
[10] Kumar, R., & Thenmozhi, M. (2006). Predicting Stock Prices Using Linear Regression. [Details missing].
[11] Zhang, Y., Zhang, G., & Zhang, J. (2019). Forecasting Bitcoin Prices Using RNN.
[12] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555.
[13] Tsai, C.-F., Lin, Y.-C., Yen, D. C., & Chen, Y.-M. (2010). Predicting Stock Prices Using Hybrid ARIMA and Genetic Algorithms.
[14] Aras, S. (2021). Stacking hybrid GARCH models for forecasting Bitcoin volatility. Expert Systems with Applications, 174, 114747.
[15] Fischer, T., & Krauss, C. (2018). Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions.
[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[17] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv preprint arXiv:1506.00019.
[18] Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, 385, Springer.
[19] Cryer, J. D., & Chan, K. S. (2008). Time series analysis: with applications in R. Springer Science & Business Media.
[20] Cho, K., et al. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078.
[21] Bibi, I., Akhunzada, A., Malik, J., Iqbal, J., Musaddiq, A., & Kim, S. (2020). A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware. IEEE Access, PP. 1-1. 10.1109/ACCESS.2020.3009819.
[22] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
[23] Pan, H., Ye, Z., He, Q., Yan, C., Yuan, J., Lai, X., Su, J., & Li, R. (2022). Discrete Missing Data Imputation Using Multilayer Perceptron and Momentum Gradient Descent. Sensors, 22, 5645. 10.3390/s22155645.
[24] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
[25] Olah, C. (2015). Understanding LSTM Networks. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[26] Hyndman, R. J., & Athanasopoulos, G. (n.d.). 7.1 Simple Exponential Smoothing. In Forecasting: Principles and Practice (2nd ed.). Retrieved from https://otexts.com/fpp2/simple-smoothing.html.
[27] Hyndman, R. J., & Athanasopoulos, G. (n.d.). Chapter 8 Exponential Smoothing. In Forecasting: Principles and Practice (3rd ed.). Retrieved from https://otexts.com/fpp3/expsmooth.html.
[28] Frost, J. (n.d.). Linear regression. Statistics By Jim. Retrieved from https://statisticsbyjim.com/regression/linear-regression/.
[29] Taylor, S. (2023). Regression analysis. Corporate Finance Institute. Retrieved from https://corporatefinanceinstitute.com/resources/data-science/regression-analysis/.
[30] Ostertagova, Eva & Ostertag, Oskar. (2011). The Simple Exponential Smoothing Model.