**CRYPTOCURRENCY PRICE PREDICTION USING STATISTICAL, MACHINE LEARNING AND DEEP LEARNING MODELS**

1st Duong Truc Uyen

IS403.O21.HTCL

University of Information Technology

[21522780@gm.uit.edu.vn](mailto:21522780@gm.uit.edu.vn)

2nd Nguyen Quang Lam

IS403.O21.HTCL

University of Information Technology

[21521057@gm.uit.edu.vn](mailto:21521057@gm.uit.edu.vn)

3rd Nguyen Thi Phuong Nhi

IS403.O21.HTCL

University of Information Technology

[19520814@gm.uit.edu.vn](mailto:19520814@gm.uit.edu.vn)

**Abstract -** Cryptocurrency price prediction has garnered significant attention due to its volatile nature and potential for substantial financial gains. This study explores the application of various predictive models including Linear Regression, ARIMA, Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Seasonal Exponential Smoothing (SEMOS), Fuzzy Time Series, and CNN-LSTM for forecasting Bitcoin, Ethereum, and Binance Coin prices. Using historical price data, each model is evaluated based on performance metrics such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). Results indicate that deep learning models, particularly RNN, consistently outperform traditional statistical methods in terms of prediction accuracy.

**Key work** – Time series, Linear Regression, ARIMA, RNN, GRU, LSTM, SEMOS, FTS, CNN-LSTM

# **I. INTRODUCTION**

**The financial world is entering a new erea, where virtual currencies, also known as cryptocurrencies, dominate. The main difference from the currency trading system is that virtual currency is decentralized, not under the control of any intermediary bank or government. Like a mesmerizing dance, virtual currencies such as Bitcoin (BTC), Ethereum (ETH) and Binance Coin (BNB) are attracting the attention of millions of users and consultants globally.**

**Cryptocurrencies hold a huge source of energy but at the same time contain dangerous risks. Instead, they can apply algorithms to understand the extent to which this virtual currency revolution is advancing. Forecasting models can be of great assistance in this area, providing methods to understand the underlying trend of the “currency”.**

**In this report, our team will use algorithms such as Linear Regression, ARIMA, RNN, GRU, LSTM, SEMOS, Fuzzy Time Series and CNN-LSTM to predict the closing price of cryptocurrencies on the data. BitCoin, Ethereum and Binance data.**

**During the evaluation process, they rely on metrics such as MAPE, RMSE, and MAE to ensure the effectiveness of the forecasting models. Based on detailed information and analysis, we hope to provide an overview of the price outlook of cryptocurrencies such as Bitcoin, ETH and Binance. At the same time, it helps readers better understand the risks and opportunities when investing in this potential field of cryptocurrency.**

# **II. RELATED WORK**

Abdussalam Aljadani [1] applied machine learning and deep learning algorithms like GRU, LSTM, and BiLSTM to analyze and predict cryptocurrency prices. Results showed that GRU outperformed BiLSTM for Bitcoin, ETH, and Cardano with the lowest RMSE values of 0.01711 for Bitcoin, 0.02662 for ETH, and 0.00852 for Cardano.

Dzaki Mahadika Gunarto, Siti Sa’adah and Dody Qori Utama [2] compared RNN and LSTM algorithms for predicting Bitcoin and Ethereum prices. The study found that LSTM outperforms RNN, with an RMSE of 0.061 and a MAPE of 5.66% for Bitcoin, and an RMSE of 0.036 and a MAPE of 4.58% for ETH, indicating LSTM's superior effectiveness in cryptocurrency price prediction.

Dian Utami Sutiksno and partners [3] esearched Bitcoin historical data forecasting using the ARIMA method and the α-Sutte indicator. The study found that ARIMA did not achieve the best accuracy, with an MSE of 295797.315 and an MAE of 497.657.

Rohit Chivukula and T.Jaya Lakshmi [4] studied machine learning algorithms for predicting Bitcoin prices using live data from quindle.com. They compared 11 regression algorithms and found that Lasso regression combined with generalized linear regression outperformed others, showing a 9% improvement in accuracy.

Mohammad J. Hamayel and Amani Yousef Owda [5] presented a method to predict cryptocurrency prices using machine learning algorithms like GRU, LSTM, and bi-LSTM, showing promising results. For GRU, the RMSE and MAPE were 174.129 and 0.2454% for BTC, 26.59 and 0.8267% for ETH, and 0.825 and 0.2116% for LTC. GRU outperformed LSTM and bi-LSTM, but all algorithms provided excellent predictions overall.

# **III. MODELS**

## **A. Linear Regression**

Linear regression is a statistical method used to model the relationship between a dependent variable (response variable) and one or more independent variables (predictors or features). The goal is to find the best-fitting linear equation that describes how the dependent variable changes as the independent variables change. [6].

A graph of a slope and a random error

Description automatically generated

*Figure 1. Linear Regression*

**Simple Linear Regression:**

In the case of a single independent variable, the relationship can be expressed with the equation:

y =

In there: y is the dependent variable; x is the independent variable; is the y-intercept of the regression line; is the slope of the regression line; is the error term.

**Multiple Linear Regression:**

When there are multiple independent variables, the equation extends to:

y =

In there: are the independent variables; is the y-intercept; are the coefficients for each independent variable; is the error term.

## **B. ARIMA**

ARIMA (AutoRegressive Integrated Moving Average) [7] is a statistical analysis model that uses time series data to gain a better understanding of the dataset or to forecast future trends. It can be understood by outlining each of its components as follows:

Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

Integrated (I): represents the differencing of raw observations to allow the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).

Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

**AutoRegressive (AR)**:

The autoregressive part of the model specifies that the output variable depends linearly on its own previous values. The AR part of order *p* is written as:

In there: is the value of time t; are the parameters of the AR part; is the white noise error term.

**Integrated (I)**:

The integrated part involves differencing the series times to make it stationary. For a time series , the differenced series is given by:

If differenced times, the series becomes:

where is the backshift operator such that .

**Moving Average (MA)**:

The moving average part specifies that the output variable depends linearly on the current and various past error terms. The MA part of order *q* is written as:

are the parameters of the MA part.

is the white noise error term.

**Together**:

## **C. Recurrent Neural Network (RNN)**

Recurrent Neural Network (RNN) is a type of neural network where the output from the previous step is used as input for the current step, making it ideal for tasks where current output depends on previous inputs, such as predicting the next word in a sentence. RNNs achieve this through a Hidden Layer that retains information about previous inputs, functioning as a Memory State. This hidden state allows the network to remember important details from earlier in the sequence. A significant advantage of RNNs is that they use the same parameters across all steps, reducing the network's complexity compared to other architectures. [8].

A diagram of a machine

Description automatically generated

*Figure 2. A simple representation of recurrent neural networks* [9]

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

*Figure 3. Working of Recurrent Neural Network* [9]

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.

The input layer 'x' receives the input to the neural network, processes it, and passes it to the middle hidden layer.

The hidden layer 'h' can comprise multiple hidden layers, each with its own activation functions, weights, and biases. If the neural network's various hidden layers' parameters are not influenced by the previous layers, meaning the network lacks memory, a recurrent neural network can be employed.

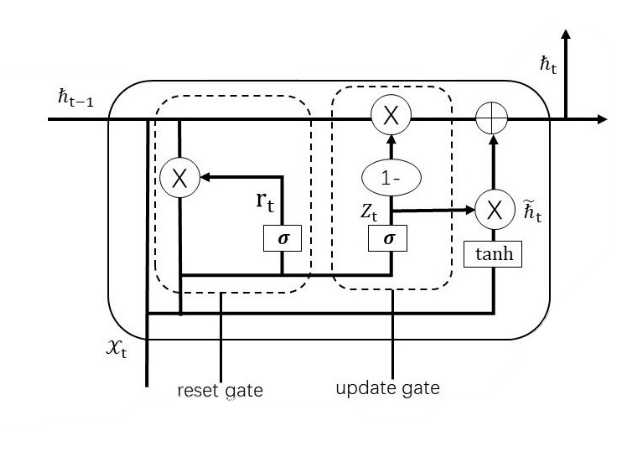
An RNN standardizes the different activation functions, weights, and biases so that each hidden layer has the same parameters. Instead of creating multiple distinct hidden layers, an RNN creates a single hidden layer and loops over it as many times as needed.

In there: : hidden state at the time t; : activation function at a hidden state, usually using a tanh function; : weight matrix for input-to-hidden connections; : input at the time t; : Weight matrix for the connection from the hidden state at time t−1 to the hidden state at time 𝑡; : Hidden state at time t – 1; : hidden state bias.

## **D. Gated Recurrent Unit (GRU)**

GRU models are a type of recurrent neural network of RNN to make predictions in time series, allowing them to model sequential data effectively. Gru models were developed to address the limitations of conventional RNN.

The LSTM model is an advanced RNN, it consists of forget gate, input gate and output gate, which can control the retention or discard of information in the sequence. The GRU model is a modified version of the LSTM model, it merges the forget gate and the input gate into an update gate but also drops the cell state, achieved reduction of amount of parameters. A GRU unit is composed of reset gate, update gate and a candidate hidden state [10].



*Figure 4 . The structure of the GRU*

Reset gate and update gate are calculated using both the hidden state from the previous time step and the input data at the current time step, it be reserved by applying a sigmoid function :

In there: , : Reset Gate, Update Gate; : Input data with time step *t;* : Hidden state of the previous time step; , , , : Weight parameters; , : Bias parameters.

The retained from the previous time steps along with the new inputs. Firstly, the previous hidden state is multiplied by the reset gate and then by a trainable weight. Secondly, the input data at the current time step is also multiplied by a trainable weight. Finally, the results obtained from summing the values from the first and second steps are passed through the tanh function.

The resultant value, derived from the *tanh* function, represents the candidate hidden state. If the value of is 1, it signifies that all information from the previous hidden state is being retained. Conversely, if the value of is 0, it indicates that the information from the previous hidden state is entirely disregarded.

The update gate's role is to assist the model in deciding how much past information from the previous hidden state should be kept for future use. In the final step, the updated hidden state is obtained by applying element-wise multiplication to the previous hidden state and (1 - update gate). This is then added to the product of the update gate and the candidate hidden state to produce the output.

## **E. Long Short-Term Memory (LSTM)**

LSTM (Long Short-Term Memory) is a specialized type of RNN designed to handle long-term dependencies in time series data. It replaces traditional hidden layer neurons with memory cells that maintain crucial cell states. LSTM uses a gating mechanism involving input, forget, and output gates to selectively filter and update information. These gates are governed by sigmoid and tanh functions, enabling effective management of information flow and storage within the network. [11].

A diagram of a tank

Description automatically generated

*Figure 5. The structure of the LSTM*

Three fully connected layers with sigmoid activation functions calculate the values for the input, forget, and output gates. Due to the sigmoid activation, the values for these gates range from 0 to 1. Moreover, an input node is needed, usually calculated using a tanh activation function. The input gate decides the extent to which the input node's value should be added to the current state of the memory cell. The forget gate chooses whether to retain or discard the current memory value. Lastly, the output gate determines if the memory cell should affect the output at the current timestep [12].

In there : Input gate; : Forget gate; : Output gate; : Input data with time step *t;* : Hidden state of the previous time step; , , , : Weight parameters; , : Bias parameters

The input gate determines how much new information is added to the neuron state. First, the input layer containing the sigmoid activation function determines which information needs to be updated, and then a tanh layer generates candidate vectors , an update is made to the state of the neuro

: Input node

In LSTMs, the input gate governs how much we take new data into account via  and the forget-gate addresses how much of the old cell internal state retain

If the forget gate is always 1 and the input gate is always 0, the memory cell internal state will remain constant forever, passing unchanged to each subsequent time step. However, input gates and forget gates give the model the flexibility of being able to learn when to keep this value unchanged and when to perturb it in response to subsequent inputs

To define how to compute the output of the memory cell, the hidden state as seen by other layers. This is where the output gate comes into play. In LSTMs, we first apply *tanh* to the memory cell internal state and then apply another point-wise multiplication, this time with the output gate. This ensures that the values of are always in the interval (-1, 1)

## **F. SEMOS**

SEMOS is a sophisticated statistical technique that refines ensemble weather forecasts by smoothing and correcting the raw model outputs. This results in more accurate and reliable weather predictions, helping meteorologists and decision-makers better understand and anticipate weather conditions.

**Ensemble Forecast Bias Correction** [13]:

Y^t, i =Yt,i + β0 + β1 Xt,i

In there: Y^t, i is the bias-corrected ensemble forecast for member iat time t; Yt, i is the raw ensemble forecast for member i at time t; Xt, i represents predictors (e.g., temperature, pressure) from the model output; β0 and β1 are coefficients determined through regression analysis using historical data.

**EnsembleSmoothing** [14]:

The smoothing process can involve techniques like kernel smoothing or local regression. A common approach is the Gaussian kernel smoothing:

Y~t, i =

In there: Y~t, i is the smoothed forecast for member i at time t; N is the number of ensemble members; Kh is the Gaussian kernel function with bandwidth h, which controls the degree of smoothing.

**Calibration of Ensemble Spread** [15]:

Calibration adjusts the spread of the ensemble to better match observed variability. This can involve techniques like quantile mapping:

In there: is the inverse cumulative distribution function (CDF) of the smoothed forecast; is the inverse CDF of the observed data; p is the probability level.

**Continuous Updating and Verification**:

The statistical model parameters , and others are updated periodically using new observations and ensemble forecasts to ensure the model remains accurate over time.

## **G. Fuzzy Time Series**

Fuzzy Time Series (FTS) is a time series forecasting method using fuzzy set theory. In FTS, temporal data is represented by fuzzy sets. This allows the model to naturally handle uncertain and ambiguous data.

Fuzzy set: is an extension of classical set theory used in fuzzy logic. Unlike classical set theory, where elements either belong or do not belong to a set in a binary manner, fuzzy set theory allows for gradual membership evaluation using a membership function valued between 0 and 1. This generalizes classical sets, as their characteristic functions are special cases of fuzzy set membership functions, taking only the values 0 or 1 [16].

Fuzzy Logic: is a type of many-valued logic where, unlike the conventional true or false values, variables can have any truth value between 0 and 1. It is a mathematical technique for modeling vagueness and uncertainty in decision-making, useful for handling imprecise or uncertain information [17].

The definition of a fuzzy set A on a background space X is as follows:

A functional function measures how much an element x is part of the base set X. An element is not included in the supplied set if the function returns 0 for it; result 1 describes a whole member of the set. Fuzzy members have values that fall between 0 and 1.

A diagram of a fuzzy curve

Description automatically generated

*Figure 6. Fuzzy set and crisp set*

Dependent function 𝜇𝐴(𝑥) satisfy the following conditions:

Song and Chissom [18] define fuzzy time series as follows:

***Definition 1***: Let Y(t) R (t = …, 0, 1, 2, …). If fuzzy sets fi(t) (i = 1, 2, …) are defined and F(t) is a collection of f1(t), f2(t), … Then F(t) is called a fuzzy time series defined on Y(t) (t = …, 0, 1, 2, …).

F(t) is a set of fuzzy sets and is a function of time t since its value may vary at different times.

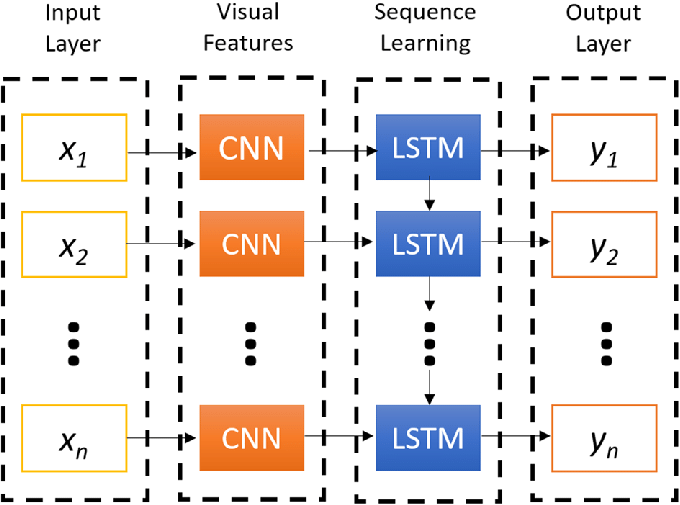
***Definition 2***: Suppose F(t) is caused by F(t-1) only, that is F(t-1) -> F(t). This relation can therefore be written as follows: F(t) = F(t-1) ○ R(t, t-1) where R(t, t-1) is the fuzzy relationship between F(t-1) and F(t), and F(t) = F(t-1) ○ R(t, t-1) is called the first-order model of F(t).

***Definition 3***: Suppose R(t, t-1) is a first-order model of F(t F(t) is referred to be a time-invariant fuzzy time series if R(t, t-1) = R(t-1, t-2) for any t, meaning that R(t, t-1) is independent of t. If not, it is referred to as a time-variant fuzzy time series.

## **H. CNN – LSTM**

CNN-LSTM is a deep learning network architecture that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). It is commonly used in processing data of spatial and temporal nature, such as images and videos.

CNN consists of two main parts: Convolutional layer and pooling layer. Time series input data will be viewed as a one-dimensional matrix, processed by convolutional layers and activation functions. The output data is then used by pooling layers to extract data features. Combining a large number of layers gives CNN an advantage in extracting features of the past and continues to add them to the layers of the LSTM model to train the model.



*Figure 7. Structure of the CNN-LSTM model*

The general implementation steps of a CNN-LSTM can be represented as follows:

CNN process (Convolutional Neural Networks):

+ Input data is passed through one or more Convolutional layers to extract spatial features.

+ Convolutional layers apply filters (kernels) that slide across the image to create feature maps.

+ Then, the feature map will be passed through Pooling layers (e.g. Max Pooling) to reduce the spatial dimension and extract important information.

LSTM (Long Short-Term Memory) process:

+ Feature maps from the CNN process are converted into time series by dividing them along the time dimension.

+ This time series is fed into the LSTM network to learn a prediction model and analyze sequential data.

+ LSTM uses gates to adjust information during computation and store important information from the past.

# **IV. METHOD**

***Introduction***

The input data is cryptocurrency time series historical data. After preprocessing, the data is divided into training and testing datasets. Models such as Linear Regression, ARIMA, GRU, RNN, LSTM, SEMOS, Fuzzy Time Series, and CNN-LSTM will be trained using the prepared dataset.

During training, the models learn patterns and relationships in the data by adjusting parameters to minimize the difference between predicted and actual outputs. After training, the models are evaluated to assess performance and generalization.

If the performance is unsatisfactory, hyperparameters will be adjusted and the models retrained.

***Implementation steps***

*Step 1*: Collect time series historical data.

*Step 2*: Split the data into 2 parts: training data, and testing data with different ratios 8:2, 7:3, 6:4.

*Step 3*: Choose a model and calibrate hyperparameters to be ready for training.

*Step 4*: Train the model.

*Step 5*: Predict based on the model.

*Step 6*: If the returned results are error values, go back to step 3, otherwise perform performance estimation and save the result.

*Step 7*: Select another model and return to step 2.

A diagram of a product model

Description automatically generated

*Figure 8. Implementation steps*

# **V. EXPERIMENT**

## **A. Dataset**

### **1. Introducing dataset**

Dataset collected using the yfinace library. BTC, ETH and BNB datasets taken between March 1, 2019 and May 16, 2024.

- **Date**: date of observation.

- **Open**: opening price on the given day.

- **High**: highest price on the given day.

- **Low**: lowest price on the given day.

- **Close**: closing price on the given day.

- **Adj close**: the closing price after adjustments for

all applicable splits and dividend distributions.

- **Volume**: volume of transactions on the given day.

### **2. Statistical description and data visualization**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Bitcoin** | **Ethereum** | **Binance** |
| **Mean** | 27593.79 | 1563.66 | 226.52 |
| **Median** | 25832.79 | 1613.61 | 246.37 |
| **Standard Deviation** | 403.54 | 1195.49 | 182.07 |
| **Sample Variance** | 310055164.1 | 1429186.4 | 33147.76055 |
| **Kurtosis** | -0.697 | -0.686 | -0.891 |
| **Skewness** | 0.5689 | 0.473 | 0.326 |
| **Range** | 69321.943 | 4701.482 | 666.298 |
| **Min** | 3761.557 | 110.606 | 9.386 |
| **Max** | 73083.5 | 4812.087 | 675.684 |
| **Sum** | 52538584.54 | 2977200.039 | 431301.579 |
| **Count** | 1904 | 1904 | 1904 |

*Table 1. Statistical description of Bitcoin, ETH, and Binance dataset*

A screenshot of a graph

Description automatically generated

*Figure 9. Histogram and Boxplot Chart of Bitcoin*

A screenshot of a graph

Description automatically generated

*Figure 10.* ***Histogram and Boxplot Chart of Ethereum***

A screenshot of a graph

Description automatically generated

*Figure 11.* ***Histogram and Boxplot Chart of Binance***

## **B. Measure**

**Mean Absolute Error (MAE):**

MAE measures the average error between the prediction and the actual value as an absolute value.

**Mean Absolute Percentage Error (MAPE):**

MAPE measures the average error between prediction and actual value in absolute percentage terms.

**Root Mean Square Error (RMSE):**

RMSE measures the average error between the prediction and the actual value as the squared error.

## **E. Result**

**1. Evaluate model accuracy**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bitcoin |  | MAE | MAPE | RMSE |
| LN | **8:2** | 11356 | 0.265 | 13107.5 |
| **7:3** | 20774.8 | 0.4 | 22331 |
| **6:4** | 39392.4 | 0.562 | 40545.8 |
| ARIMA | **8:2** | 113173 | 0.469 | 19292.5 |
| **7:3** | 15599.5 | 0.811 | 21344.4 |
| **6:4** | 14959.8 | 0.367 | 16660 |
| RNN | **8:2** | 948.55 | 0.02 | 1428.28 |
| **7:3** | 1034.89 | 0.028 | 1368.15 |
| **6:4** | **697.51** | **0.021** | **1110.48** |
| GRU | **8:2** | 1009.69 | 0.021 | 1531.42 |
| **7:3** | 795.22 | 0.0206 | 1180.26 |
| **6:4** | 720.75 | 0.022 | 1085.42 |
| LSTM | **8:2** | 1514.71 | 0.034 | 1861.62 |
| **7:3** | 1213.79 | 0.033 | 1507.77 |
| **6:4** | 1233.25 | 0.041 | 1504.85 |
| SEMOS | **8:2** | 25794.5 | 0.822 | 28893.9 |
| **7:3** | 22822.2 | 0.894 | 24621.5 |
| **6:4** | 62900.2 | 2.411 | 63878.9 |
| FTS | **8:2** | 1536.69 | 0.042 | 1802.54 |
| **7:3** | 1457.74 | 0.049 | 1695.99 |
| **6:4** | 2077.74 | 0.071 | 2470.4 |
| CNN-LSTM | **8:2** | 1105.68 | 0.023 | 1630.29 |
| **7:3** | 986.16 | 0.025 | 1479.21 |
| **6:4** | 2116.38 | 0.079 | 2369.68 |

*Table 2. Results of model accuracy of Bitcoin dataset*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ethereum |  | MAE | MAPE | RMSE |
| LN | **8:2** | 854.85 | 0.444 | 935.74 |
| **7:3** | 1726.86 | 0.966 | 1768.05 |
| **6:4** | 2862.84 | 1.627 | 2936.19 |
| ARIMA | **8:2** | 517.12 | 0.185 | 778.044 |
| **7:3** | 728.112 | 0.308 | 960.787 |
| **6:4** | 1165.33 | 0.731 | 1267.47 |
| RNN | **8:2** | 57.63 | 0.023 | 83.64 |
| **7:3** | **45.77** | **0.02** | **67.77** |
| **6:4** | 58.49 | 0.033 | 78.63 |
| GRU | **8:2** | 57.86 | 0.023 | 83.39 |
| **7:3** | 2160.40 | 0.99 | 2245.54 |
| **6:4** | 1924.67 | 0.99 | 2019.25 |
| LSTM | **8:2** | 53.56 | 0.021 | 79.82 |
| **7:3** | 50.97 | 0.023 | 73.77 |
| **6:4** | 65.03 | 0.034 | 83.70 |
| SEMOS | **8:2** | 2001.6 | 1 | 2083.171 |
| **7:3** | 1558.9 | 0.907 | 1639.72 |
| **6:4** | 4996.89 | 2.871 | 5022.02 |
| FTS | **8:2** | 156.44 | 0.069 | 191.96 |
| **7:3** | 136.48 | 0.064 | 175.95 |
| **6:4** | 171.85 | 0.088 | 197.61 |
| CNN-LSTM | **8:2** | 65.17 | 0.025 | 97.56 |
| **7:3** | 48.16 | 0.021 | 70.03 |
| **6:4** | 160.24 | 0.093 | 191.74 |

*Table 3. Results of model accuracy of Ethereum dataset*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binance |  | MAE | MAPE | RMSE |
| LN | **8:2** | 184.848 | 0.38 | 202.157 |
| **7:3** | 240.251 | 0.435 | 258.157 |
| **6:4** | 346.976 | 0.522 | 366.365 |
| ARIMA | **8:2** | 105.114 | 0.33 | 129.321 |
| **7:3** | 72.610 | 0.193 | 115.277 |
| **6:4** | 138.027 | 0.502 | 145.514 |
| RNN | **8:2** | 7.387 | 0.021 | 12.331 |
| **7:3** | **6.465** | **0.019** | **10.69** |
| **6:4** | 11.91 | 0.041 | 14.49 |
| GRU | **8:2** | 7.927 | 0.021 | 13.312 |
| **7:3** | 7.917 | 0.023 | 11.882 |
| **6:4** | 14.412 | 0.0493 | 16.879 |
| LSTM | **8:2** | 7.035 | 0.019 | 11.931 |
| **7:3** | 7.565 | 0.021 | 12.526 |
| **6:4** | 13.43 | 0.046 | 15.96 |
| SEMOS | **8:2** | 412.774 | 1.561 | 429.433 |
| **7:3** | 315.092 | 1.171 | 330.438 |
| **6:4** | 631.829 | 2.261 | 638.329 |
| FTS | **8:2** | 19.239 | 0.062 | 23.362 |
| **7:3** | 19.578 | 0.064 | 23.385 |
| 6:4 | 20.47 | 0.069 | 23.888 |
| CNN-LSTM | 8:2 | 9.57 | 0.025 | 15.45 |
| 7:3 | 8.88 | 0.027 | 13.42 |
| 6:4 | 31.53 | 0.117 | 36.12 |

*Table 4. Results of model accuracy of Binance dataset*

**The best model for each dataset**

**For Bitcoin dataset**

A screenshot of a computer screen

Description automatically generated

*Figure 11. Prediction results for the next 30, 60, 90 days of the RNN model Bitcoin dataset (6:4)*

**For Ethereum dataset**

A screenshot of a computer screen

Description automatically generated

F *Figure 12. Prediction results for the next 30, 60, 90 days of the RNN model Ethereum dataset (7:3)*

**For Ethereum dataset**

A graph showing a line of orange and white

Description automatically generated with medium confidence

*Figure 13. Prediction results for the next 30, 60, 90 days of the RNN model Binance dataset (7:3)*

)**VI. CONCLUSION**

The results of this study show that the RNN model was the most suitable for predicting the Cryptocurrency prices of Bitcoin, Ethereum, and Binance. RNN is a deep learning model that fits best to Time Series data. We can rely on this model to recommend or give advice on the cryptocurrency market.

It is crucial to remember that the cryptocurrency market is extremely unpredictable and volatile, influenced by a wide range of outside influences. Thus, enhancing the precision of cryptocurrency price forecasts requires constant observation, model improvement, and the addition of new elements or methodologies.

This research enhances understanding of cryptocurrency price prediction through a detailed analysis of forecasting models. The findings aid market participants in making informed decisions, managing risks, and optimizing investment strategies in the dynamic cryptocurrency market. Further research is encouraged to advance forecasting capabilities and adapt models to the rapidly evolving landscape.

**ACKNOWLEDGMENTS**

First of all, we would like to express our sincere thanks to all of the teachers and students of the University of Information Technology - Vietnam National University Ho Chi Minh City and the teachers of the Information System Faculty for helping us to have the basic knowledge that is the basis for implementing this topic.

In particular, the authors would like to send our deepest gratitude and gratitude to Ph.D. Nguyen Dinh Thuan – theoretical instructor of Data Analysis In Business; Ms. Trinh Thi Thanh Truc, Ms. Dang Vu Phuong Uyen, and Mr. Nguyen Minh Nhut – practical instructor of Data Analysis In Business. They have guided directly, corrected and contributed many valuable comments to help the authors successfully complete the course report.

# **REFERENCES**

|  |  |
| --- | --- |
| [1] | A. Aljadani, "DLCP2F: a DL-based cryptocurrency price prediction framework," *Discover Artificial Intelligence,* vol. 2, pp. 1-19, 2022. |
| [2] | Dzaki Mahadika Gunarto, Siti Sa'adah, Dody Qori Utama, "Predicting Cryptocurrency Price Using RNN and LSTM Method," *Jurnal Sisfokom,* vol. 12, pp. 1-8, 2023. |
| [3] | Dian Utami Sutiksno, Ansari Saleh Ahmar, Nuning Kurniasih, Eko Susanto, Audrey Leiwakabessy, "Forecasting Historical Data of Bitcoin using ARIMA and α-Sutte Indicator," in *2nd International Conference on Statistics, Mathematics, Teaching, and Research*, 2018. |
| [4] | Rohit Chivukula, T. Jaya Lakshmi, "Cryptocurrency Price Prediction: A Machine Learning Approach," *Sensors & Transducers,* vol. 224, pp. 44-47, 2020. |
| [5] | Mohammad J. Hamayel, Amani Yousef Owda, "A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms," vol. 2, no. 4, pp. 477-496, 2021. |
| [6] | K. MALI, "What is Linear Regression?," 13 Jun 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/. [Accessed 18 Jun 2024]. |
| [7] | A. Hayes, "Autoregressive Integrated Moving Average (ARIMA) Prediction Model," 05 April 2024. [Online]. Available: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp. [Accessed 18 Jun 2024]. |
| [8] | "Introduction to Recurrent Neural Network," 5 Jun 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/. [Accessed 18 Jun 2024]. |
| [9] | A. Biswal, "Recurrent Neural Network(RNN) Tutorial: Types, Examples, LSTM and More," 14 Feb 2023. [Online]. Available: https://www.simplilearn.com.cach3.com/tutorials/deep-learning-tutorial/rnn.html. [Accessed 18 Jun 2024]. |
| [10] | Chien-Jung Chen, Fu-I Chou, Jyh-Horng Chou, "Temperature Prediction for Reheating Furnace by Gated Recurrent Unit Approach," vol. 10, pp. 33362 - 33368, 2022. |
| [11] | Jiayu Qiu, Bin Wang, Changjun Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," pp. 1-15, 2020. |
| [12] | "10.1. Long Short-Term Memory (LSTM)," [Online]. Available: https://d2l.ai/chapter\_recurrent-modern/lstm.html. [Accessed 18 Jun 2024]. |
| [13] | D. S. Wilks, "Statistical Methods in the Atmospheric Sciences," vol. 100, no. 3, 2011. |
| [14] | B. W. Silverman, "Density Estimation for Statistics and Data Analysis," 1998. |
| [15] | Thordis L. Thorarinsdottir, Tilmann Gneiting, "Probabilistic Forecasts of Wind Speed: Ensemble Model Output Statistics by using Heteroscedastic Censored Regression," *Journal of the Royal Statistical Society Series A: Statistics in Society,* vol. 173, no. 2, pp. 371-388, 2010. |
| [16] | L. Zadeh, "Fuzzy sets," *Information and Control,* vol. 8, p. 338, 1965. |
| [17] | Surya Priy, Abhishek rajput, "Fuzzy Logic | Introduction," 24 Jun 2023. [Online]. Available: https://www.geeksforgeeks.org/fuzzy-logic-introduction/. [Accessed 18 Jun 2024]. |
| [18] | Qiang Song, Brad S. Chissom, "Forecasting enrollments with fuzzy time series — Part I," *Fuzzy Sets and Systems,* vol. 54, pp. 1-9, 1993. |