Using Forecasting Models in Cryptocurrency Price Prediction

HOANG NHAT MINH[1], LE NGUYEN NHAT MINH[2], AND TRAN THI LUYEN[3]

[1]Faculty of Information Systems, University of Information Technology, (e-mail: 21522336@gm.uit.edu.vn)
[2]Faculty of Information Systems, University of Information Technology, (e-mail: 21522338@gm.uit.edu.vn)
[3]Faculty of Information Systems, University of Information Technology, (e-mail: 21521107@gm.uit.edu.vn)

ABSTRACT Predicting cryptocurrency price has become a central focus for investors and traders in the rapidly evolving landscape of digital assets. Cryptocurrencies like Bitcoin, Ethereum, and other altcoins have captured global attention, with their prices subject to fluctuations influenced by various factors, from market sentiment to technological developments. With the unprecedented volatility of these markets, the need for accurate forecasting models has never been more crucial. In this research, we conduct an in-depth analysis using various models including Linear regression (LR), Autoregressive Integrated Moving Average (ARIMA), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Seasonal Autoregressive Integrated Moving Average with eXogenous factors (SARIMAX), Neural Basis Expansion Analysis Time Series (NBEATS), Random Forest (RF) to predict cryptocurrency price and determine the effectiveness of each model.

INDEX TERMS Cryptocurrency, price prediction, Bitcoin, Ethereum, Linear regression, ARIMA, RNN, GRU, LSTM, SARIMAX, NBEATS, RF.

I. INTRODUCTION

Cryptocurrency, a digital form of currency, has transformed the traditional concept of money. Unlike cash, cryptocurrencies exist purely in electronic form, utilizing encryption techniques for secure transactions. Bitcoin, the pioneering cryptocurrency introduced in 2009 by Satoshi Nakamoto, paved the way for a myriad of digital currencies that followed. The decentralized nature of cryptocurrencies means they operate without a central authority, using blockchain technology to record and verify transactions across a network of computers. This decentralization provides a level of transparency and security previously unseen in financial systems. As interest and adoption continue to grow, cryptocurrencies are reshaping the landscape of finance, offering individuals around the globe new avenues for transactions, investments, and financial freedom.

By using forecasting models and machine learning algorithms to analyze historical data of Bitcoin, Ethereum, and BNB, the research objective is to provide valuable insights for investors and traders to forecast the cryptocurrency price movement. To evaluate the performance of each predictive model, we use three metrics Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE).

II. RELATED WORKS

Mahir Iqbal and his team [1] applied a machine learning based time series analysis in predicting the price of Bitcoin. They used ARIMA, FBProphet, and XG Boosting for time series analysis and evaluated these models by using three parameters RMSE, MAE, and R Squared (R^2). After conducting the analysis, they concluded that ARIMA is the best of three with RMSE score and MAE score are 322.4 and 227.3 respectively.

Harsha Nanda Gudavalli and Khetan Venkata Ratnam Kancherla [2] identified four machine learning algorithms RF, Gradient Boosting (GB), LSTM, and GRU to make long-term predictions of cryptocurrency prices. Overall, LSTM is the most accurate model based on RMSE score of 0.01083, MSE score of 0.00011, and R Squared score of 0.80618.

Junyi Zhu [3] applied LR for Bitcoin price prediction with the accuracy rate for training RSS and testing RSS are 94.6% and 96.99% respectively.

Namrata Hemraj Gawali [4] developed predictive models to predict the price of Bitcoin using LSTM, Seasonal Autoregressive Integrated Moving Average (SARIMA), SARIMAX, and RNN. In conclusion, LSTM outperformed all other models with R Squared score of 0.9519, MAE score of 2476.68, MSE score of 10958440.29 and RMSE score of 3310.35.

Alikhan Bulatov [5] used N-BEATS trained on Bitcoin daily,

hourly, and up-to-the-minute data in comparison with LSTM and ARIMA. In this analysis, it is concluded that N-BEATS is the best model for forecasting the Bitcoin price with the RMSE score in daily, hourly, and minute data are 308.859, 59.303, and 13.678 respectively.

III. MATERIALS

A. DATASET

This research focuses on analyzing cryptocurrency trends through three data sets of three cryptocurrencies:

- BTC/USD
- ETH/USD
- BNB/USD

Those data sets are extracted from the website investing.com with the selected period from March 1, 2019, to March 1, 2024. Each data set contains 1828 data rows and has 6 attributes:

- Price: The last transacted price in a day.
- Open: The opening price in a day.
- High: The highest price in a day.
- Low: The lowest price in a day.
- Vol: The total amount traded in a day.
- Change %: The percentage change in the closing price compared to the previous day's closing price.

B. DESCRIPTIVE STATISTICS

TABLE 1. BTC, ETH, BNB's Descriptive Statistics

	BTC	ETH	BNB
Count	1828	1828	1828
Mean	25,983.05	1,488.75	212.44
Std	16,801.607	1,160.628	171.7707
Min	3,715	107.58	9.28
25%	10,255.075	256.75	25.3325
50%	23,637.8	1,573.75	242.4
75%	38,686.75	2,152.043	318.675
Max	67,520	4,808.09	676.12

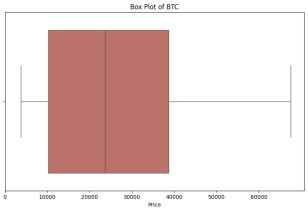

FIGURE 1. Bitcoin price's boxplot

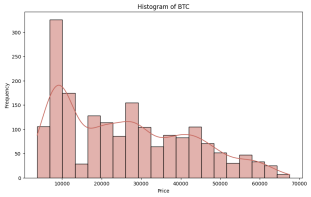

FIGURE 2. Bitcoin price's histogram

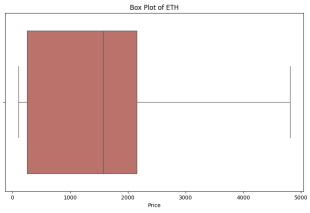

FIGURE 3. Ethereum price's boxplot

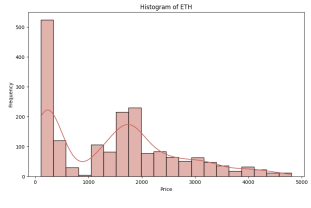

FIGURE 4. Ethereum price's histogram

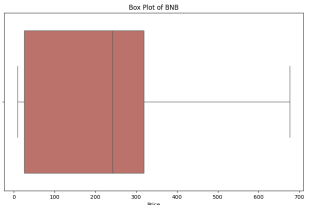

FIGURE 5. BNB price's boxplot

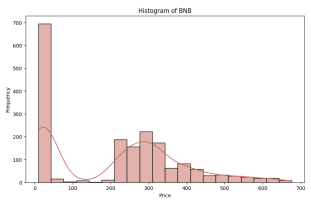

FIGURE 6. BNB price's histogram

IV. METHODOLOGY

A. LINEAR REGRESSION

Linear regression is an algorithm that provides a linear relationship between an independent variable and a dependent variable. A linear regression model with more than one independent variable is called a multiple linear regression model. A multiple linear regression model has the form:

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

Where:

- Y is the dependent variable.
- $X_1, ..., X_k$ 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.

B. AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

ARIMA is an acronym for AutoRegressive Integrated Moving Average. It is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. The full model can be written as:

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Where:

- y'_t is the value of the time series at time t.
- c is the constant term.
- $\phi_1, ..., \phi_p$ are the autoregressive coefficients.
- $\theta_1, ..., \theta_q$ are the moving average coefficients.
- ε_t is the white noise at time *t*.

A non - seasonal ARIMA model is classified as an ARIMA (p, d, q) model, where:

- p is the number of lag observations included in the model.
- d is the number of times that the raw observations are differenced.
- q is the size of the moving average window, also called the order of moving average.

C. AUTOREGRESSIVE INTEGRATED MOVING AVERAGE WITH EXOGENOUS FACTORS (SARIMAX)

SARIMAX is an acronym for Seasonal Autoregressive Integrated Moving Average with eXogenous factors. The SARIMAX model is an improved version of the SARIMA model, with exogenous factors (X) as external feature parameters for enhancing the model's performance, reducing the prediction errors, overcoming the autocorrelation issues, and improving

the prediction results. The SARIMAX model consists of both seasonal effects and exogenous factors that can be used as SARIMAX (p, d, q) * (P, D, Q), while the exogenous factors are optional parameters. The exogenous factors are used to support the prediction model and to provide it with more details. The SARIMAX model can be presented as:

$$d_t = c + \sum_{n=1}^{p} \alpha_n d_{t-n} + \sum_{n=1}^{q} \theta_n \epsilon_{t-n} + \sum_{n=1}^{r} \beta_n x_{n_t} + \sum_{n=1}^{P} \phi_n d_{t-sn} + \sum_{n=1}^{Q} \eta_n \epsilon_{t-sn} + \epsilon_t$$

Where:

- P is seasonal autoregressive order.
- D is seasonal differencing order.
- Q is seasonal moving average order.
- s is the length of the seasonal cycle.

D. RANDOM FOREST

Random forest is an ensemble learning method for classification, regression, and other tasks, that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

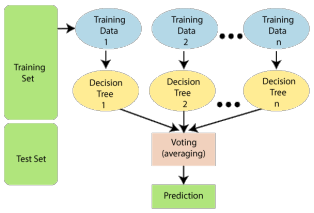

FIGURE 7. The structure of Random Forest model

Each tree in a random forest randomly samples subsets of the training data in a process known as bootstrap aggregating (bagging). The model is fit to these smaller data sets and the predictions are aggregated. Several instances of the same data can be used repeatedly through replacement sampling, and the result is that trees that are not only trained on different sets of data, but also different features used to make decisions.

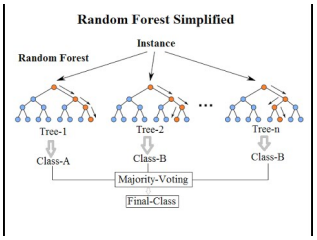

FIGURE 8. Overview of Random Forest model

The following steps explain the working Random Forest algorithm:

1) Select random samples from a given data or training set.
2) This algorithm will construct a decision tree for every training data.
3) Voting will take place by averaging the decision tree.
4) For classification, use majority voting; for regression, use averaging to derive the final output.

E. RECURRENT NEURAL NETWORK (RNN)

Recurrent Neural Networks (RNNs) are a class of artificial neural networks specifically designed for processing sequences of data. Unlike traditional feedforward neural networks, RNNs have cyclical connections, allowing them to maintain a memory of previous inputs.

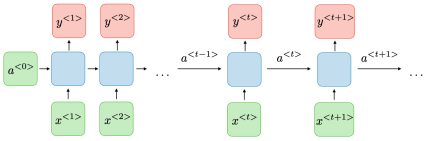

FIGURE 9. Overview of Recurrent Neural Network

For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

and

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Where:

- $W_{ax}, W_{as}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally.
- g_1, g_2 are activation functions.

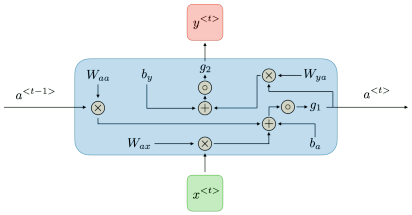

FIGURE 10. Architecture of a traditional RNN

In the case of a recurrent neural network, the loss function $\mathcal{L}$ of all time steps is defined based on the loss at every time step:

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

Backpropagation is done at each point in time. At timestep T, the derivative of the loss $\mathcal{L}$ with respect to weight matrix W is expressed as follows:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\bigg|_{(t)}$$

F. GATED RECURRENT UNIT (GRU)

A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture designed to handle sequence dependencies efficiently. GRUs address some of the limitations of standard RNNs, such as the vanishing gradient problem,

by incorporating gating mechanisms that regulate the flow of information. GRUs simplify the RNN architecture by using two primary gates: the reset gate and the update gate. These gates determine which information should be passed to the output and what should be retained in the hidden state.

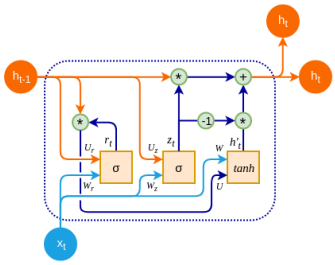

FIGURE 11. Architecture of GRU

In general, a GRU cell consists of 4 components:

- Reset gate:
$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$
Where:
  - x_t represents the current information.
  - h_{t-1} represents the previous information.
  - W_r represents how much current information is weighted.
  - U_r represents how much previous information is weighted.
- Update gate:
$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$
Where:
  - x_t represents the current information.
  - h_{t-1} represents the previous information.
  - W_z represents how much current information is weighted.
  - U_z represents how much previous information is weighted.
- New memory content:
$$h'_t = tanh(W_h x_t + r_t . U_h h_{t-1})$$
Where:
  - x_t represents the current information.
  - h_{t-1} represents the previous information.
  - r_t represents how much previous information should be forgotten.
  - W_h represents how much current information is weighted.
  - U_h represents how much previous information is weighted.
- Final memory content:
$$h_t = z_t . h'_t + (1 - z_t) . h_{t-1}$$
Where:
  - z_t represents how much the unit updates its information with the current information.
  - h_{t-1} represents the previous information.
  - h'_t represents the current information with some degree of dependency on the previous information.

G. LONG SHORT-TERM MEMORY (LSTM)

LSTMs Long Short-Term Memory is a type of RNNs Recurrent Neural Network that can detain long-term dependencies in sequential data. LSTMs are able to process and analyze sequential data, such as time series, text, and speech. They use a memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs.

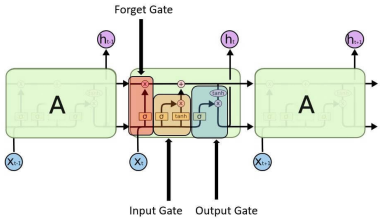

FIGURE 12. Architecture of LSTM

LSTM have 4 different components: cell state (memory cell), forget gate, input gate, output gate.

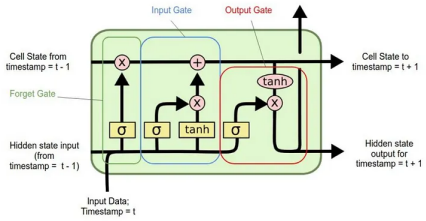

FIGURE 13. LSTM components

- Forget gate:
$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f)$$
- Input gate:
$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C)$$
- Output gate:
$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

H. NEURAL BASIS EXPANSION ANALYSIS TIME SERIES (NBEATS)

Neural Basis Expansion Analysis for Time Series (NBEATS) is a deep learning architecture tailored for forecasting time series data. It diverges from traditional methods that depend on statistical techniques or basic machine learning models by employing deep neural networks to directly learn intricate temporal patterns from the data.

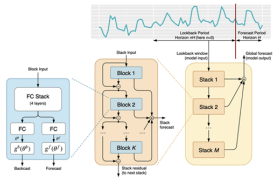

FIGURE 14. Architecture of NBEATS

1) Basic building blocks

These blocks have two fully connected layers in a fork architecture after a stack of fully connected layers. Once the initial stack of layers has processed the input data, the subsequent forecast and backcast layers come into play. These layers use the forward and backward expansion coefficients generated by the initial fully connected network to create the final forecast and backcast outputs.

The forecast layers focus on generating accurate predictions for future data points, while the backcast layers work on estimating the input values themselves, given the constraints on the functional space that the network can use to approximate signals.

2) Stacking blocks

These basic building blocks are stacked together using a technique known as “doubly residual stacking”. This method creates a hierarchical structure with residual connections across different layers, allowing for better interpretability and a more transparent network structure. Each block receives an input and generates a forecast (forward prediction) and a backcast (backward estimation).he backcast output of the previous block is subtracted from its input to create the input for the next block. This residual branch plays a key role in how the network processes the input signal. By handling the input signal step-by-step, it ensures that each block focuses on a specific part of the data, making the overall prediction more accurate and efficient. The partial forecasts, created by individual blocks, each capture different patterns and components of the input data. When move through the hierarchical structure, these partial forecasts are combined to form the final output. This creates a hierarchical decomposition of the forecasting process, where forecasts from the basic building blocks are combined to form the overall prediction.