iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** XII **Month of publication:** December 2022

DOI: https://doi.org/10.22214/ijraset.2022.48411

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XII Dec 2022- Available at www.ijraset.com

Univariate Time Series Analysis of Cryptocurrency Data using Prophet, LSTM and XGBoost

Pranathi Kodicherla[1], Nanditha Velagandula[2]
[1]*Student, Dept. of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Telangana, India*
[2]*Student, Dept. of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Telangana, India*

Abstract: ***Cryptocurrency, also known as crypto, is any digital or virtual currency that uses cryptography to safeguard transactions and circulates without the authority of a central bank. Bitcoin, the first and most widely used decentralized cryptocurrency, was introduced in 2009. After a few years of unrivalled dominance, it lost its monopoly in 2011, when the first competitive alternative currencies arose. As of November 2022, there are almost 21,000 cryptocurrencies in circulation. Because there is no government credit backup, cryptocurrency prices are typically volatile. The cost of one Bitcoin rose from zero at its debut in 2009 to $13 in 2013 and then to $68789 in 2021, with numerous shifts and fluctuations along the way. The accurate forecasting of the Bitcoin price is critical for investors to make decisions and for governments to create regulatory laws.***
This paper examines the ability of the models - Prophet, Long Short-Term Memory (LSTM), and eXtreme Gradient Boosting (XGBoost) to predict the price of Bitcoin reliably. Using the performance metrics like RMSE, each model was thoroughly trained and tested to discover which one operates more efficiently. After examining the price of Bitcoin from 2012 to 2021, we concluded that the Long Short-Term Memory (LSTM) model proves to be the most efficient when dealing with variable and difficult-to-predict data such as Bitcoin values since it portrays promising results in comparison.
Keywords: Prophet, Long Short-Term Memory (LSTM), XGBoost, Cryptocurrency, Price Prediction

I. INTRODUCTION

Bitcoin (BTC) is a cryptocurrency or virtual currency intended to act as money and a payment method independent of any single individual, group, or institution, hence eliminating the need for third-party intervention in monetary transactions. At most, there will always be 21 million bitcoins in circulation. These are digital currency units that cannot be altered or inflated. Furthermore, it is not required to purchase one complete bitcoin; you can buy a fraction of one if that is all you need or desire. One bitcoin can be divided into eight decimal places (100 millionths of a bitcoin), and the smallest unit is known as a Satoshi.

Satoshi Nakamoto, a pseudonymous individual or group, developed Bitcoin in 2008, as the banking crisis was in full swing, and outlined the technology in a white paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System," which became the Magna Carta for how Bitcoin operates today. The paper suggested an exquisite solution to the problem of value transfer between two entities without using a trusted intermediary such as a bank, when the entities may be concealed behind pseudonyms (as bitcoin's creator is) or physically located on the other side of the world. Nakamoto created two inextricably linked notions: the bitcoin private key and the blockchain record. You administer bitcoin with a private key, a string of randomized letters and numbers that secures a virtual vault storing your purchase. Each private key is tracked on the virtual, encrypted, distributed, and shared ledger called the blockchain. When a transaction occurs on the blockchain, data from the preceding block is replicated to a new block with the most recent data and is then encrypted. Finally, the transaction is validated by network validators known as miners. When a transaction is confirmed, a new block is opened, and a Bitcoin is created and distributed as a reward to the miner(s) who verified the data within the block—the miner(s) are then free to use, hold, or sell the Bitcoin. The first Bitcoin block, Block 0, was mined on January 3, 2009. This block is frequently referred to as the "genesis block." For every 210,000 blocks, the reward for mining Bitcoin is halved. Initially, the block reward was 50 new bitcoins. The third halving occurred on May 11, 2020, reducing the payout for each block discovery to 6.25 bitcoins.

Another approach to acquiring bitcoins is to use a cryptocurrency exchange to purchase them in fiat currency such as US dollars. Depending on your objectives, mining or buying bitcoin can transfer value over the world, an investment vehicle, a means of exploring developing technology, or a store of wealth comparable to gold.

An accurate Bitcoin price forecast is vital for investors to make decisions and for governments to implement regulatory laws. In this paper, we have employed Time series forecasting.

A time series is a set of observations xt, each being recorded at a specific time t. A discrete-time time series is one in which the set T0 of times at which observations are made is a discrete set. Continuous time series is obtained when values are recorded continuously over some time interval. [9]

Time series forecasting predicts future values by fitting a model to historic, timestamped data. Moving average, exponential smoothing, and ARIMA are some traditional techniques. However, other models such as RNNs, Transformers, and XGBoost can also be used. The Mean Square Error (MSE) or Root Mean Square Error (RMSE) is commonly used to evaluate models.

Forecasting a time series to forecast future demand and sales has enormous business value. It drives essential business planning, procurement, and management processes in many companies, such as finance, manufacturing, and real estate. Forecasting errors would be catastrophic because they would impact the entire supply chain. It is hence critical to obtain accurate projections.

Time series forecasting is divided into two types - Univariate Time Series Forecasting [10] and Multivariate Time Series Forecasting. [11] Univariate Time Series Forecasting is used when only the initial values of the time series are used to anticipate the following matters. Multivariate Time Series Forecasting occurs when predictors other than the series' beginning value or the series are utilized to create predictions.

We will now look into the models used for time series forecasting in this paper.

A. Prophet

Prophet, an open-source software developed by Facebook's Core Research Team, is a technique for forecasting time series data based on an additive model incorporating yearly, monthly, and daily seasonality and holiday impacts. It works best with time series with substantial seasonal effects and data from multiple seasons. Prophet is resistant to missing data and trend alterations and typically handles outliers well. [5] According to Taylor and Letham's [6] research, Prophet is utilized in numerous Facebook applications to make credible forecasts and outperforms any other solution in most circumstances. At its heart is the combination of three functions plus an error term: growth g(t), seasonality s(t), holidays h(t), and error e(t):

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Fig 1: The Prophet model equation

B. Long Short-Term Memory (LSTM)

One of the most powerful dynamic classifiers is Long Short-Term Memory Recurrent Neural networks (LSTM-RNN). [7] Long Short-Term Memory Networks (LSTMNs) are a type of sequential network, that permits information to be stored. It can solve the vanishing gradient problem that RNNs confront. In a larger perspective, LSTM functions similarly to an RNN cell and comprises three sections known as gates. The Forget Gate, the first section, regulates whether the data from the preceding timestamp should be stored or discarded. A sigmoid function determines it. It examines the preceding state (ht-1) and the content input (Xt) for each number in the cell state Ct-1 and returns a value between 0 (overlook this) and 1. (keep this).

$$f_t = \sigma(W_f . [h_t - 1, x_t] + b_f)$$

Fig 2: The Forget Gate Sigmoid function

The cell, known as the Input Gate, strives to learn new data from the input to this cell and selects which of the input values should be used to modify the memory in the second half. The sigmoid function determines whether 0 or 1 values are permitted. And the tanh function gives the data weight, defining its value on a scale of -1 to 1.

$$i_t = \sigma(W_j . [h_t - 1, x_t] + bi)$$
$$C_t = \tanh(W_C . [h_t - 1, x_t] + b_C)$$

Fig 3: The Input Gate Sigmoid and tanh functions

Finally, in the third section, or Output Gate, the cell conveys the updated information from the current timestamp to the next timestamp. The sigmoid function governs whether 0 or 1 values are permitted through. And the tanh function specifies which values can pass through 0 and 1. And the tanh function applies weight to the provided values, calculating their importance on a scale of -1 to 1 and multiplies it with the sigmoid output.

$$O_t = \sigma(W_o[h_t - 1, x_t] + b_o$$
$$h_t = o_t * \tanh(C_t)$$

Fig 4: The Output Gate Sigmoid and tanh functions

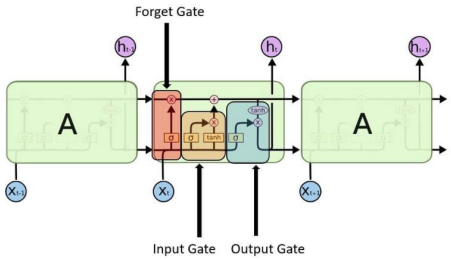


Fig 5: The LSTM Architecture

C. eXtreme Gradient Boosting (XGBoost)

XGBoost is a rapid and scalable implementation of the Gradient Boosting Machine (GBM), a popular tool among artificial intelligence approaches due to its advantages, such as excellent prediction accuracy and easy parallelism. [8]

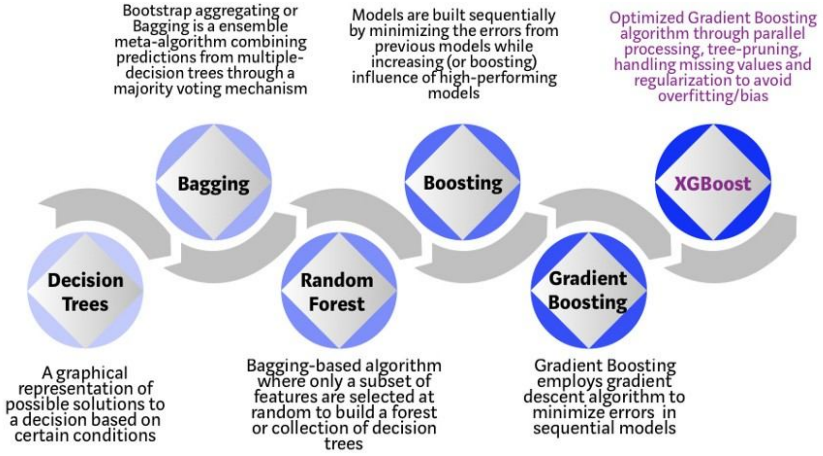


Fig 6: Evolution of the XGBoost model

Boosting is a machine learning ensemble model in which trees are produced consecutively with the goal of reducing the faults of the prior tree. Any arbitrary differentiable loss function and the gradient descent optimization algorithm are used to fit models. The technique is called "gradient boosting" because the loss gradient is minimised when the model is fitted.

Gradient boosting consists of three components:

- A loss function that must be optimised.
- A slow learner who can't make forecasts.
- An additive model in which weak learners are added to minimise the loss function.

The type of loss function utilized is determined by the problem being solved. We use squared error for regression. To make predictions, decision trees are poor learners. Existing trees in the model are not modified, and new trees are inserted one at a time. When adding trees, a gradient descent approach is utilized to reduce loss.

Extreme Gradient Boosting, abbreviated as XGBoost, is a fast open-source adaptation of the gradient boosting technique. This approach is used for both classification and regression. XGBoost enhances the standard GBM framework with system enhancements such as parallelization, tree pruning, cache awareness, and algorithmic advances such as regularization, sparsity awareness, and cross-validation.

We will now look into the performance metrics used for evaluating the models.

D. Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE/RMSD) is the standard deviation of the residuals (prediction errors). Residuals measure how far from the regression line data points are; RMSE measures how to spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. The Root Mean Square Error (RMSE) has been employed as a standard statistical tool to quantify model performance in various domains like meteorology, air quality determination, and climate research investigations. To calculate RMSE, we must first compute the residual, which is the difference between the true and predicted values for each data point. Then we need to compute the residual norm for each data point and then the mean of these residuals followed by the square root of the resulting mean. RMSE requires precise measurements at each projected data point. As a result, it is frequently employed in supervised learning problems.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

RMSD = root-mean-square deviation
i = variable i
N = number of non-missing data points
x_i = actual observations time series
$\hat{x}_i$ = estimated time series

Fig 7: Formula for calculation of RMSE Score

II. LITERATURE REVIEW

Several researchers have studied about the various models for predicting the price of cryptocurrencies. A few of the studies are reviewed below:

In his study [1], Xiangxi Jiang incorporated various novel characteristics, including hour-based prediction, the use of data from the previous 24 hours, window normalisation, and the comparison of different types of models with varying numbers of layers. In this study, six models are compared. According to the findings, Multi-Layer Perceptron (MLP) was an unsatisfactory example for predicting price based on current trends, however Long Short-Term Memory (LSTM) provided the greatest prediction when past memory and Gated Recurrent Network (GRU) were incorporated. Because all six models work similarly, alternative models might be preferred in different contexts.

In this study [2], Mohammed Mudassir, Shada Bennbaia, Devrim Unal, and Mohammad Hammoudeh used ML models to anticipate BTC prices for the short to medium term. Using machine learning, it delivered highly accurate end-of-day, short-term (7 days), and mid-term (30 and 90 days) BTC price estimates. The following ML models were used: ANN, SANN, SVM, and LSTM. Overall, the LSTM performed the best. All of the created models are satisfactory and work well, with the categorization models scoring close to 65% accuracy for the following day's forecast and 62% to 64% success for the seventh-nintieth-day forecast.

Patrick Jaquart, David Dann, and Christof Weinhardt used four distinct models of machine learning on four different prediction horizons to assess the bitcoin industry's short-term predictability [3]. They noticed that every model they evaluated produced statistically reasonable predictions. Their systems were able to estimate binary market activity with an accuracy range from 50.9% to 56% percent. Recurrent neural networks have also been demonstrated to be appropriate for prediction applications.

In this paper, [4] G. Vidyulatha, M. Mounika, and N. Arpith used a time series analysis model ARIMA to estimate the market price of Bitcoin over five and a half years of data from 2015 to 2020. They projected bitcoin prices for the next four months using the ARIMA model. They also contrasted it with the linear regression model. Extensive prediction results reveal that the suggested ARIMA model outperformed the LR model in determining variability in weighted bitcoin costs in the brief run.

III.PROPOSED WORK

The cryptocurrency Bitcoin continues to rise in popularity as many organizations have begun adopting it. Many factors influence Bitcoin pricing, including supply and demand for BTC, competition from alternative cryptocurrencies, media coverage, production costs, and local and worldwide legislation. As a result, interpreting the fundamentals of success and making reliable estimates is a difficult job.

We propose a forecasting automation technique to help investors decide whether or not to participate in the bitcoin or other crypto markets. In this system, we tested the Bitcoin forecasting abilities of 4 different models in Python: Prophet, XGBoost, and LSTM, to determine the most suitable model for the abovementioned task.

We have followed the below steps to examine the ability of the models to predict the price of Bitcoin reliably:

1) Data collection
2) Data cleaning
3) Exploratory data analysis
4) Model Building

A. Data Collection

In this research, we forecast BTC prices using various time series models. We gathered the relevant data from Kaggle's "Bitcoin Historical Data" dataset. It provides historical bitcoin market statistics for various bitcoin exchanges where trading occurs at 1-minute intervals. It includes Bitcoin data from January 2012 to March 2021, with minute-by-minute updates of OHLC (Open, High, Low, Close), Volume in BTC, indicated currency, and weighted bitcoin price. The dataset contains the eight columns shown below:

1) Timestamp: Start time of time window (60s window), in Unix time
2) Open: Open price at the start time window
3) High: High price within the time window
4) Low: Low price within the time window
5) Close: Close price at end of the time window
6) Volume (BTC): Volume of BTC transacted in this window
7) Volume (Currency): Volume of corresponding currency transacted in this window
8) Weighted Price: Volume Weighted Average Price

B. Data Cleaning

The dataset that we considered in this research is a large dataset of 317MB. It has 8 columns with minute-by-minute timestamps. It has approximately 4.7 million rows Since it is time series data, we converted the timestamp column format from Device timestamp to datetime64. Then we made the DateTime as the index.

```
# store csv into a pandas dataframe
df = pd.read_csv('bitstampUSD_1-min_data_2012-01-01_to_2020-12-31.csv')
print(df.head())
```

	Timestamp	Open	High	...	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1325317920	4.39	4.39	...	0.455581	2.0	4.39
1	1325317980	NaN	NaN	...	NaN	NaN	NaN
2	1325318040	NaN	NaN	...	NaN	NaN	NaN
3	1325318100	NaN	NaN	...	NaN	NaN	NaN
4	1325318160	NaN	NaN	...	NaN	NaN	NaN

Fig 8: Original timestamps

```
# Start of cleaning the data. Converting Timestamp to datetime64
df.Timestamp = pd.to_datetime(df.Timestamp, unit='s')
# sets the index as the date
df.index = df.Timestamp
df.head()
```

Timestamp	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
2011-12-31 07:52:00	2011-12-31 07:52:00	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
2011-12-31 07:53:00	2011-12-31 07:53:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-31 07:54:00	2011-12-31 07:54:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-31 07:55:00	2011-12-31 07:55:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-31 07:56:00	2011-12-31 07:56:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig 9: Timestamps after modification

1) *Time Resampling:* Next, using the resample() function, we resampled the data to the average daily value of each column to remove nearly identical data corresponding to consecutive minutes of the day.
2) *Missing Values:* Finally, we drop any missing values that are present using dropna() function.

```
# Resample the data to the average daily value of each column. Removes excessive frequency
df = df.resample('D').mean()
# drops any missing values that are present
df = df.dropna()
df.head()
```

Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
2011-12-31	4.465000	4.482500	4.465000	4.482500	23.829470	106.330084	4.471603
2012-01-01	4.806667	4.806667	4.806667	4.806667	7.200667	35.259720	4.806667
2012-01-02	5.000000	5.000000	5.000000	5.000000	19.048000	95.240000	5.000000
2012-01-03	5.252500	5.252500	5.252500	5.252500	11.004660	58.100651	5.252500
2012-01-04	5.200000	5.223333	5.200000	5.223333	11.914807	63.119577	5.208159

Fig 10: Data frame after time resampling and removal of missing values

This left us with 3,376 rows instead of 4.7 million with no missing values and accurate datetime information.

C. Exploratory Data Analysis

For the training of various models, instead of using the complete data from January 2012 to March 2021, we decided to limit the data to only the most recent four years since only in 2017 did Bitcoin prices begin to rise owing to the cryptocurrency bubble and bitcoin scarcity after an extended period of consistently low prices and volume of bitcoins.

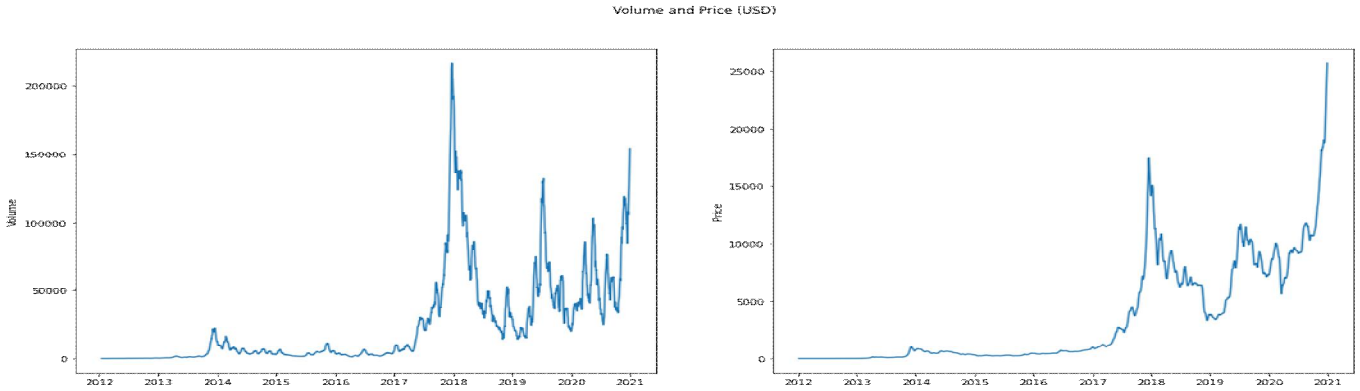


Fig 11: Yearly Fluctuations in Volume and Price of Bitcoin in USD

D. Model Building

To evaluate the efficacy of our forecasting model, we usually divide the time series into two parts: training and validation or testing. This method of splitting the time series is known as fixed partitioning.

We will train our model during the training period and then evaluate it during the validation period. In this work, we split the dataset 70:30, meaning that 70% of the data will be used for training and 30% for validation.

1) Prophet

Prophet framework includes its own specific data frame that allows it to easily handle time series and seasonality. Two fundamental columns are required for the data frame. Column names cannot be changed in the model. One of these columns is "ds," which holds the datetime series. It is Timestamp in our data. The other column is "y," and it contains the values of the time series in the data frame. It is 'Weighted-Price', or volume weighted average price, according to our data. The framework works well with seasonal time series and gives several choices for dealing with the dataset's seasonality. Seasonality can be set on a yearly, weekly, or daily basis. In our example, we assumed that the seasonality was daily. First, we created a data frame with future dates for prediction. Then, instantiating the Prophet model object (Prophet()). The data frame with future dates requires periods (the number of forecasted periods, which is the length of the test data), freq (frequency), and include history (default value is true). The predict() method is then called, with the future data frame as an argument. Columns in the output include ds, yhat, yhat_upper, and yhat_lower. The RMSE value was then determined using the predicted values. Finally, as indicated, we visualized the plot of the forecasted (yhat) values along with the upper (yhat_upper) and lower (yhat_lower) bounds of the predicted values.

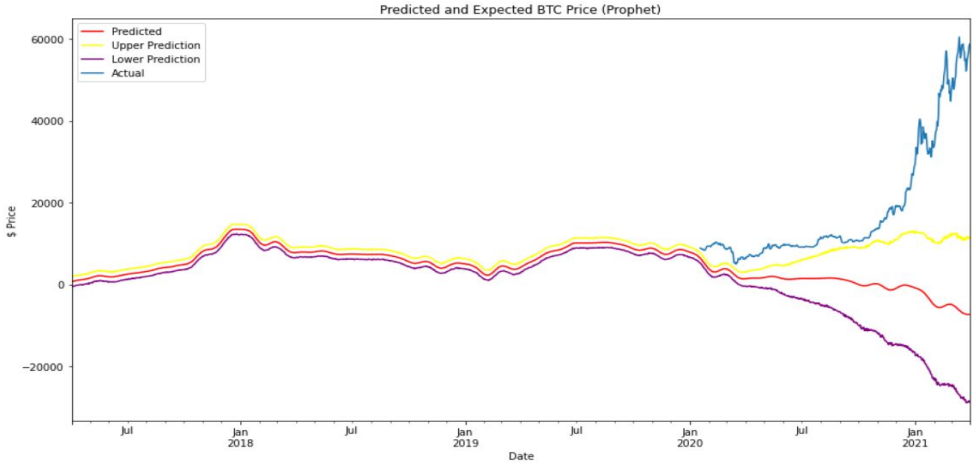


Fig 12: Predicted and Expected Bitcoin prices (Prophet)

2) LSTM

Long Short-Term Memory (LSTM) models are a type of recurrent neural network capable of learning sequences of observations and hence well suited for time series forecasting. Before building the model and training it, we performed Data Scaling using the fit_tranform() function of the MinMaxScaler() class where the minimum of feature is made equal to zero and the maximum of feature equal to one on the training set without changing the shape of the original distribution.
The transformation is given by:

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max-min) + min
where min, max = feature_range.
The fit_tranform() function takes the training set as a parameter and performs two operations in a single step - fit and transform. The fit function computes the formulation to transform the column but does not apply the actual transformation. The computation is stored as a fit object. The transform method takes advantage of the fit object in the fit() method and applies the actual transformation to the column.
We then constructed a Sequential() model and added the LSTM layer. The LSTM function takes units and activation parameters. The value '12' for units signifies the number of output dimensions after hot encoding the integer values. The activation value 'relu' refers to the rectified linear activation function. It is used in place of tanh to overcome the vanishing gradient problem in RNNs.
In the fit() function, we used 50 epochs and the "adam" optimizer after referring to existing research[13] and testing other alternatives.
The values were predicted for the next year after performing suitable transformations using the predict() function. Finally, the predicted values are plotted alongside the actual values using the plot() function.

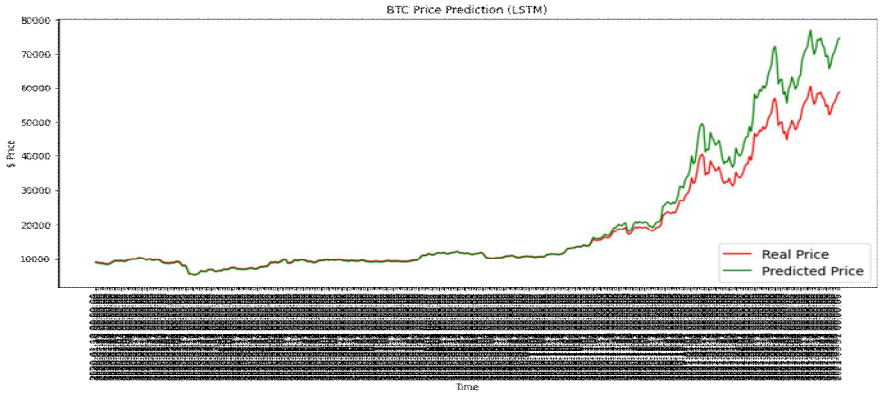


Fig 13: Predicted and Expected Bitcoin prices (LSTM)

3) XGBoost

This research used the XGBoost model on the time series dataset. Therefore, the model required creating time series features from the DateTime index to be used alongside its target price labels for predicting prices. For this purpose, we wrote a specific method, create_features, which takes training or testing data frame and the label weighted price as arguments. This method is used to create a variety of features like an hour, day of the week, quarter, month, year, day of the year, day of the month, and week of the year.
First, we split the data into 70 and 30 percent for training and testing, respectively. Then we assigned training and testing features and labels using the above-mentioned specific method create_features. Next, we created a model using the xgboost package. The XGBoost library has its custom API, but for this research, we used the method via the scikit-learn wrapper class, which is XGBRegressor. It takes several arguments like the min_child_weight, which represents the minimum sum of instance weight needed in a child and is 10 in our case; an objective which specifies the learning task and the corresponding learning objective i-e., reg: linear; booster which specifies which booster to use taken as gbtree; colsample_bytree indicates subsample ratio of columns when constructing each tree which is taken as 0.3; learning rate as 0.1; max_depth which specifies the maximum depth of the tree for base learners taken as 5; n_estimators which represents the number of gradient boosted trees, taken as 100 in our case.
Then we fit the model into training data using a built-in function fit() . After the training, we predict the prices using test data. We compare the predicted prices with actual prices by visualization using the plot() built-in function.

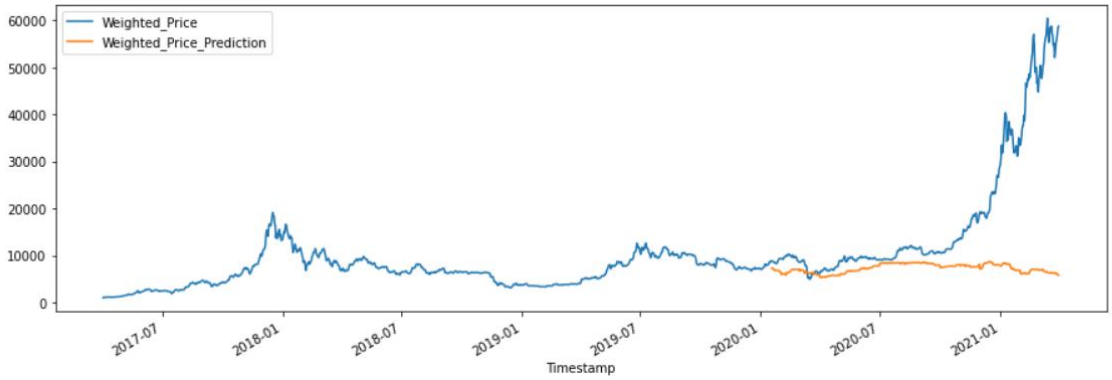


Fig 14: Predicted and Expected Bitcoin prices (XGBoost)

IV. RESULT ANALYSIS

In this section, we present the results of the models used to predict the bitcoin prices.

A. Prophet

The Prophet model failed to accurately predict the bitcoin prices on the test data. The RMSE score obtained in this model is 25141.556. While the upper bounds estimate has an accurate slope direction, this model didn't perform well when compared to the test data.

B. LSTM

The Long Short-Term Memory (LSTM) model did a fairly good job in forecasting the prices of Bitcoin. The RMSE score obtained in this model is 5200.278 which is the best of all the three models compared in this paper. The model accurately predicted the slope direction and identified the trends as well.

C. XGBoost

The XGBoost model failed to forecast the rise in prices and predicted that the prices will remain stagnant with few fluctuations. The calculated RMSE score is 18484.329 indicating that the model is better than Prophet but falls behind LSTM by a huge margin.

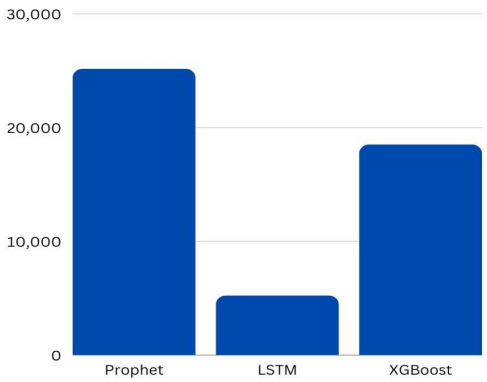


Fig 15: Comparison of RMSE scores of Prophet model, LSTM model and XGBoost model

V. CONCLUSIONS

In conclusion, we successfully laid out the differences in the models' performances – Prophet, LSTM, and XGBoost in predicting the price of cryptocurrencies like Bitcoin using performance metrics like RMSE. We also articulated the probable reasons for the high performance of the LSTM model compared to the others in making reliable predictions.

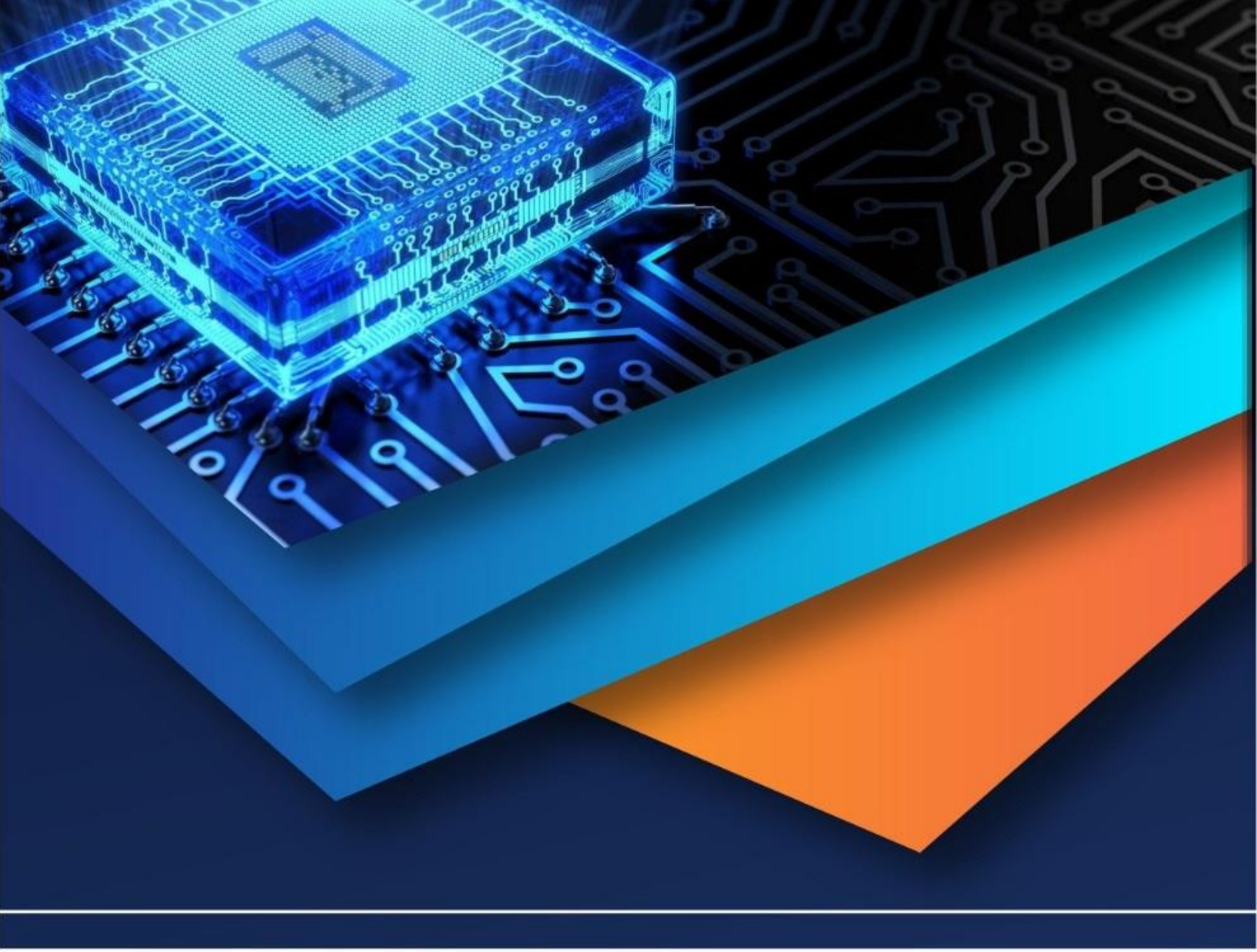
VI. ACKNOWLEDGMENT

We extend our gratitude to the Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology for their extraordinary cooperation and invaluable feedback.

We take this opportunity to extend our sincere obligation towards all the people who have helped us in this endeavor. Without their active guidance, encouragement and cooperation, we would not have made headway in the research paper.

REFERENCES

[1] Xiangxi Jiang "Bitcoin Price Prediction Based on Deep Learning Methods", Journal of Mathematical Finance, February 11, 2020.
[2] Mohammed Mudassir, Shada Bennbaia, Devrim Unal & Mohammad Hammoudeh, "Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach," Neural Computing and Applications, 04 July 2020.
[3] Patrick Jaquart, David Dann and Christof Weinhardt, "Short-term bitcoin market prediction via machine learning," November 2021.
[4] G. Vidyulatha, M. Mounika and N. Arpith, "Crypto Currency Prediction Model using ARIMA", Turkish Journal of Computer and Mathematics Education,2020.
[5] Emir Žunić, Kemal Korjenić, Kerim Hodžić and Dženana Đonko, "Application of Facebook's Prophet Algorithm for Successful Sales Forecasting Based on Real-world Data", International Journal of Computer Science & Information Technology (IJCSIT) Vol 12, No 2, April 2020.
[6] Taylor, S. J., and Letham, B. (2018), "Forecasting at scale. The American Statistician", 72(1), 37-45. https://doi.org/10.1080/00031305.2017.1380080.
[7] Ralf C. Staudemeyer and Eric Rothstein Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks", arXiv:1909.09586v1. [cs.NE] 12 Sep 2019.
[8] T. Chen and C. Guestrin, ''XGBoost: A scalable tree boosting system,'' in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016, pp. 785–794.
[9] P.J.Brockwelland, R.A.Davis, "Introduction to Time Series and Forecasting", 3rd edition. Springer, 2016.
[10] Moritz, S. et al. (2015) "Comparison of different Methods for Univariate Time Series Imputation in R", arXiv.org. Available at: https://arxiv.org/abs/1510.03924.
[11] Chakraborty, K., Mehrotra, K., Mohan, C., & Ranka, S. (1992), "Forecasting the behavior of multivariate time series using neural networks".
[12] Prophet Forecasting at Scale. https://research.fb.com/prophet-forecasting- at-scale/, 2018. [Online; accessed April 2018.
[13] Diederik P. Kingma, and Jimmy Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION", arXiv:1412.6980v9 [cs.LG] 30 Jan 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY