Analyzing the Effectiveness of Machine Learning Model in Bitcoin Price Prediction

Akarsh Kallumadiyil Thomas[1], Shivani[2], Manjot Singh[3], Srikanta Mallik[4], Deepti Sinha[5]
Pradeepta Kumar Sarangi[6]

[1, 2, 3, 6]*Chitkara University Institute of Engineering and Technology,*
Chitkara University, Punjab, India
[4]*Cognizant Technology Solutions, UAE,*
[5]*Christ University, Ghaziabad, UP, India*
akarsh0143.cse19@gmail.com, shivani1164.cse19@chitkara.edu.in, srikanta.mallik@gmail.com,
deepti.sinha@christuniversity.in, pradeeptasarangi@gmail.com

***Abstract* – Blockchain is the new era of technology which has started to shape and define new dimensions in computing and information technology. Bitcoin cryptocurrency uses blockchain and it is one of the most well renowned and significant cryptocurrencies. From the last decade, investment in bitcoin or any other cryptocurrency has been in trend but due to its highly volatile nature, investors view it as a risky investment. Therefore, there is a need for good predictions before making any investment decisions. In this research, we seek to accurately prediction of the value of bitcoin with a low error rate. The machine learning technique that we have suggested is LSTM (Long Short-Term Memory) to outlook the value of bitcoin for the oncoming 5 days. Bitcoin price prediction will act as a guidance and solution, which can provide a useful insight for the investors before investing their money into it. This research implements a LSTM model which is a class of machine learning. The experimental results show that the LSTM models produce a very low error rate and are capable of predicting the bitcoin price with a higher accuracy.**

Key words: Bitcoin prediction, Machine Learning, Time series analysis.

1. INTRODUCTION

Bitcoin refers to the digital currency which was created by Satoshi Nakamoto [1] in 2009 and did not gain much popularity till 2012. However, it is now utilized extensively for both transactional and investment purposes. Bitcoin is considered as a decentralized currency because there is on such governing body to regulate it. Basically, bitcoins are generated through the software mining where high performing programming is used. This requires high end servers and computing efficiencies.Apart from bitcoin, there are plenty of other coins that have been created and those are known as “Altcoins”.

Virtual currencies or cryptocurrencies are the recently evolved worldwide phenomenon therefore they maintain a consistent identity, structure, and function. As a matter of fact, they are increasingly recognized as a superior financial medium with significant potential as time progresses. The creation of Bitcoin was done in order to decrease the use of third party like banks, and reduce the time and costs of the transactions. The major factor distinguishing Bitcoin and fiat currencies is its decentralization.

Similar to other digital currencies, Bitcoin prices also varies with time and demand. Nevertheless, the metrics impacting bitcoin and the stock market differ. There are numerous algorithms which are used on stock exchange data for price prediction [2]. Therefore, this research aims to implement one of the most effective ML models namely LSTM to forcast the bitcoin price. To forecast its values, the machine learning and neural network utilizes numerical historical data [3].

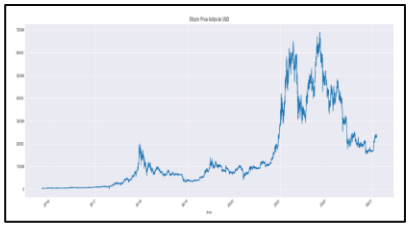

Fig. 1. Bitcoin historical price

The graph in Figure 1 represents the historical price of the Bitcoin from October 2015 to February 2023.

2. RELATED LITERATURE

For Bitcoin price prediction the precision of accuracy and the low error rate of the model is what matters most. A number of researches have been done and is still ongoing to produce the best models for the price prediction. Some of the major techniques used are LSTM, CNN, GRU, Random Forest, and several other machine learning techniques. Mangla et al [4] have implemented various machine learning and statistical models for bitcoin prediction.

Jiang [5] have considered deep learning methods such as Multi-Layer Perceptron (MLP), LSTM, and Gated Recurrent Network (GRU) to predict future bitcoin prices by rearranging bitcoin price data per minute to hours. The trail results demonstrated that LSTM model with two tiers, and two tiers of GRU obtained the finest outcome, with the minimum RMSE of 19.02. Livieris et al. [6] offered a multi-entry cryptocurrency deep neural network model for predicting bitcoin prices. The model used each cryptocurrency data as input which handles them independently so that cryptocurrency information can be processed separately. The CNN model offered with a LSTM layer attains 55.03% precision on the Bitcoin dataset.

Awoke et al. [7] have implemented GRU and LSTM. The findings indicated that the LSTM model was only better for forecasting prices in advance for 7 days and having a 12-day window size, but the GRU algorithm was much more effective with the remaining window sizes and the number of days to forecast for. Hence, GRU is better than LSTM for prediction of cryptocurrency time series. Marne et al. [8] have applied the LSTM algorithms for bitcoin price forecasting. The results were computed by plotting graphs and the RMSE of that algorithm was found to be 3.38 which was pretty minimal as we compare to other reference papers they had taken in account for. Jang et al. [9] proposed a Bayesian Neural Network (BNN) model by taking the information from blockchain and macro-economic factors to forecast bitcoin price time series. According to their result, the BNN model forecasts the volatility orientation fairly effectively.

Mittal et al. [10] suggested a multivariable linear regression to forcast the prices for bitcoin and altcoins in accordance with their open, high, and low prices. The result was that the model was highly accurate with a 99% accuracy rate. Hua [11] proposed two different models: ARIMA and LSTM for forecasting the bitcoin price and compared the precision of the two models. His major finding was that LSTM could make predictions more efficiently, and the precision rate is also higher as compared to ARIMA.

3. OBJECTIVE AND METHODOLOGY

This work aims to implement a LSTM model for for predicting the bitcoin prices for further five days. The model will be trained with the train dataset and after productive training, the trained model will be used for price forecasting.

4. THE DATASET

Primely, we collect the dataset from the Coinbase Pro API using a library called Historic-Crypto. The dataset signifies the Bitcoin value in USD. The data consists of all the data like opening price, closing price etc. about bitcoin from 27th October 2015 to 5th February 2023 as shown in Figure 2. It consists of approximately 37.8 million samples of Bitcoin price.

	time	low	high	open	close	volume
0	2023-02-06 00:00:00	22923.32	22942.44	22936.34	22938.03	8.021512
1	2023-02-05 23:59:00	22932.16	22941.29	22935.69	22936.34	3.480214
2	2023-02-05 23:58:00	22931.65	22941.92	22939.28	22936.43	1.724614
3	2023-02-05 23:57:00	22932.83	22943.93	22942.68	22937.51	1.397812
4	2023-02-05 23:56:00	22941.43	22954.03	22951.78	22941.43	2.604386
...	...	...	...	...	...	...
3781142	2015-10-27 00:04:00	287.07	287.07	287.07	287.07	0.086100
3781143	2015-10-27 00:03:00	287.08	287.09	287.08	287.09	0.554400
3781144	2015-10-27 00:02:00	286.89	287.10	287.10	286.89	72.403200
3781145	2015-10-27 00:01:00	287.09	287.09	287.09	287.09	0.822760
3781146	2015-10-27 00:00:00	287.10	287.10	287.10	287.10	0.536200

3781147 rows × 6 columns

Fig. 2. Bitcoin Dataset

Secondly, we filter and clean the dataset. This includes filtering out unneeded features existing in the data gathered like volume, low, high, open. The cleaning involves extracting all the partial data from rows and check if there is any null value present in the data frame.

```
In [13]: print('Null Values:',btc_input_df_datetype.isnull().values.sum())
         print('If any NA values:', btc_input_df_datetype.isnull().values.any())

         Null Values: 0
         If any NA values: False
```

Fig. 3. Checking if any Null value is present

5. IMPLEMENTATION DESIGN

(i) Lag Plots

Log plots are essential for searching patterns like trends, randomness, and seasonality. The plot can be drawn by the representation of time series data in x-axis, and the lag of the time series data points in the y-axis. The lag plots for different time frames are represented in Figure 4.

We can see that the correlation is very high for the minute lag curve, and it decreases as we go to a higher time period such as the hourly, and the daily lag curves. The correlation decreases noticeably with the weekly lag curve and with nearly no correlation for the monthly lag curve. As a result, it is necessary to re-sample the data from the minute level to a maximum of the daily level, thus preserving the auto-correlation.
Therefore, we have converted our 37.8 million samples of minute price into 2660 samples of daily price.

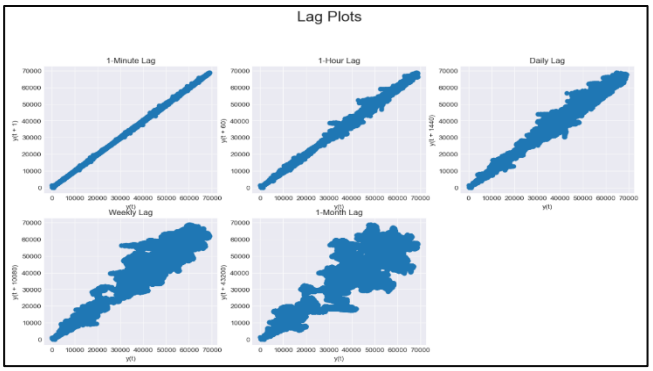

Fig. 4. Lag Plots

(ii) Train-Test Split

Now, we will do the training and testing. We will take 60 numbers of sample dataset for implementing and testing ,and the remaining dataset to train sample in our algorithm. We will follow this by plotting a graph of the train-test split. Figure 5 represents a simple train-test plot of the closing prices. The train data price action is from Oct 27th, 2015, to Dec 6th, 2022, and the test data price action is from Dec 7th, 2022, to Feb 5th, 2023.

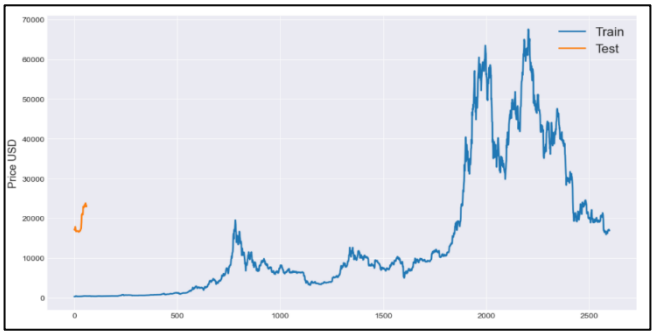

Fig. 5. Graph of Train-Test Split

(iii) Scaling

Following this, the data will be scaled because we need the train and test dataset which will be scaled. Here, we are using the MinMaxScaler function from scikit-learn library to scale our data. Scaling should be completed once then the training and testing sets will be separated and will be scaled separately; if not, the training process will be affected by the test set which will lead to a poor prediction of our model. If the data will be normalized before splitting it then the test set will be used for normalization as a reference point. This will have an impact on training data values and ultimately lead to the leakage of test set data.

(iv) Data Generator for LSTM

After scaling the dataset, our next step will be to generate the data to feed our LSTM model. A function will be defined called dataset_generator_lstm() which will split the input sequences into the windows of dataset suitable for our LSTM algorithm. Now we will define our lookback period, which is the window-size of the number of prior timesteps used in order to forecast the next timestep. For us, we took it to be five, meaning that our model will take the final 5 days of dataset to forecast the data of the current day.

```
In [35]: def dataset_generator_lstm(dataset, look_back=5):
             dataX, dataY = [], []

             for i in range(len(dataset) - look_back):
                 window_size_x = dataset[i:(i + look_back), 0]
                 dataX.append(window_size_x)
                 dataY.append(dataset[i + look_back, 0])
             return np.array(dataX), np.array(dataY)

         trainX, trainY = dataset_generator_lstm(scaled_train)

         testX, testY = dataset_generator_lstm(scaled_test)

         print("trainX: ", trainX.shape)
         print("trainY: ", trainY.shape)
         print("testX: ", testX.shape)
         print("testY", testY.shape)

         trainX:  (2595, 5)
         trainY:  (2595,)
         testX:  (55, 5)
         testY (55,)
```

Fig. 6. Code for Data generator for LSTM

(v) Restructuring the input into a shape of 3D Tensor

And now we will restructure the input (trainX and testX) into a shape of 3D structure form of samples, timesteps and features for LSTM. It implies that input tier awaits a 3D array of dataset when mounting the algorithm and when we make forecasting, the input data needs to be transformed in shape of 3D structure of features, samples and timesteps. The spacimens represent the volume of data points we have and timesteps are equal to the number of time steps we execute our RNN and lastly features include the number of features in each time step.

First check the current shape of trainX and testX

```
In [40]: print(trainX.shape)
         print(testX.shape)

         (2595, 5)
         (55, 5)
```

And now reshape trainX and testX

```
In [41]: trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))

         testX = np.reshape(testX, (testX.shape[0], testX.shape[1], 1 ))

         print("Shape of trainX: ", trainX.shape)
         print("Shape of testX: ", testX.shape)

         Shape of trainX:  (2595, 5, 1)
         Shape of testX:  (55, 5, 1)
```

Fig. 7. Reshaping the input layer for the LSTM network

(vi) Setting Up LSTM Network Architecture

After reshaping the input layer, we will use the Tensorflow library to create our LSTM network architecture. We will use the sequential model imported from Keras as well as two LSTM layers for our model. The drop-out was implemented to regularize. The amount of units allocated to the LSTM parameter is 128 with a 20% dropout rate.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5, 128)            66560

 dropout (Dropout)           (None, 5, 128)            0

 lstm_1 (LSTM)               (None, 64)                49408

 dropout_1 (Dropout)         (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 116,033
Trainable params: 116,033
Non-trainable params: 0
```

Fig. 8. Overview of the LSTM network architecture

(vii) Training LSTM Model

At this stage, we are nearly ready to shape our LSTM model with the training set. But beforehand, we need to explain an optimizer and a loss function for our model. The optimizer that we have used to optimize the problem is “adam” and for the loss function we have Mean squared error. Starting with the keras library, we import the callback: EarlyStopping, and ModelCheckpoint. These callback helps to have control over the training process and are used to save the algorithm at various checkpoints or after each stage. Also, EarlyStopping is used to stop the training when the best loss is reached. ‘val_loss’ is the value of the cost function for the cross-approval data, while ‘loss’ is the value of the cost feature for the training data.

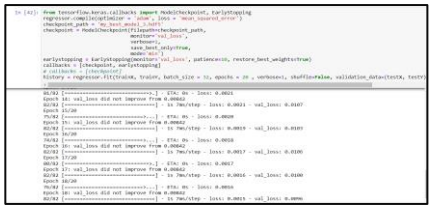

Fig. 9. Code for generating Epochs

(viii) Prediction using LSTM and plotting line graph

The implementation code is shown in Figure 10.

```
In [45]: predicted_btc_price_test_data = model_from_saved_checkpoint.predict(testX)
         predicted_btc_price_test_data = scaler_test.inverse_transform(predicted_btc_price_test_data.reshape
         test_actual = scaler_test.inverse_transform(testY.reshape(-1, 1))
```

Fig. 10. Inverse transformation on test data

After that, we are generating a graph plot of the actual test data (in blue) along with the predicted test data (in red), in Figure 11. Here, we can see that the predicted test data and actual test data are moving in tandem.

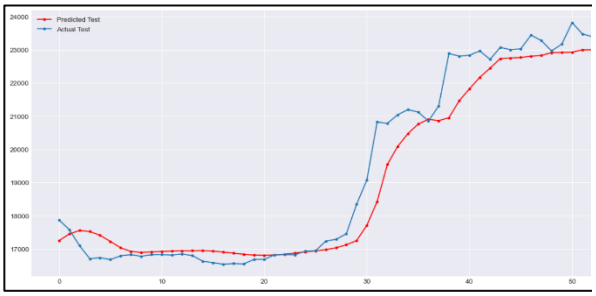

Fig. 11. Plot of predicted test and actual test data

(ix) Prediction of LSTM using trainX and line graph plotting against actual trainY

This process is also similar to the last step, having exception that is carried out by the inverse transformation on our training dataset.

```
In [47]: predicted_btc_price_train_data = model_from_saved_checkpoint.predict(trainX)
         predicted_btc_price_train_data = scaler_train.inverse_transform(predicted_btc_price_train_data.reshape
         train_actual = scaler_train.inverse_transform(trainY.reshape(-1, 1))
```

Fig. 12. Inverse transformation on train data

Thereafter, we will generate the graph for the predicted train data (in red) and the actual train data (in blue). Figure 13 shows that the predicted train and actual train are very synchronized.

Fig. 13. Plot of predicted train and actual train data

(x) RMSE

Lastly, the root mean square error (RMSE) will be generated for testing and training dataset. RMSE is the measure of the degree to which a regression line will match the data points. Figure 14 indicates that the RMSE for the test data is 667.90.

```
In [61]: rmse_lstm_test = math.sqrt(mean_squared_error(test_actual, predicted_btc_price_tes
         print('Test RMSE: %.3f' % rmse_lstm_test)
         Test RMSE: 667.899
```

Fig. 14. RMSE of test data

```
In [62]: rmse_lstm_train = math.sqrt(mean_squared_error(train_actual, predicted_btc_price_tra
         print('Train RMSE: %.3f' % rmse_lstm_train)
         Train RMSE: 1679.382
```

Fig. 15. RMSE of train data

Subsequently, we will also generate the RMSE for train data which will be 1679.38 as illustrated in Figure 15. Here, we can see that the RMSE loss obtained for the test data is much less significant than the RMSE loss for the train data.

6. RESULT ANALYSIS

Now the LSTM algorithm has been trained on historical dataset, prices of Bitcoin for the next five days will be forecasted. The dataset used by us for this algorithm, the latest historic price we have with us is the value Bitcoin on 5th February 2023. So, for the upcoming 5 days, what will be the price of Bitcoin, we are going to forecast. As our lookback period is set to be five days, which means we will forecast the Bitcoin price for the next day using the information on the Bitcoin prices of the immediately preceding five days. Finally, we generate a graph of the entire predicted test data including the next five days (in red) relative to the actual test data (in blue), as shown in Figure 16.

From the chart above, we can see that the predicted bitcoin prices are following the actual bitcoin price trend closely. This demonstrates LSTM's effectiveness in working with time series or sequential data such as the cryptocurrency prices.

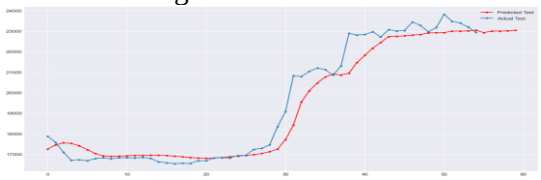

Fig. 16. Plot of the entire test data prediction (including the five future days) and actual testY

7. CONCLUSION AND FUTURE WORK

Ultimately, predicting the Bitcoin price is quite challenging considering the many factors affecting the market. However, the LSTM algorithm, is applied here, is a basic algorithm that takes into concerns only a few features like the historic prices and can be a great tool for Bitcoin price prediction. The model we have developed is very accurate with an accuracy of approximately 97%, with a low RMSE of 667.9 for test data and 1679.38 for train data. Nevertheless, it is worth noting that the price of bitcoin for the upcoming five days forecasted by our model should not serve as a guide for making investment decisions without adequate analysis. The model's forecast is based only on historical price developments, which are not the only factors influencing future price developments. Market sentiment, political and policy systems, inflation, and many other factors also play a key role in bitcoin price movements.

Nonetheless, our model can still be enhanced by further adding sentimental analytics by utilizing a twitter dataset and adding more complex and sophisticated models that support high level neural networks to improve the efficiency and accuracy of our model.

REFERENCES

[1] M. Chauhan, S. Gupta, M. Sandhu, "Short-Term Electric Load Forecasting Using Support Vector Machines", ECS Transactions, Volume 107, Number 1, 2022.
[2] M. Singh, I. Batra, B. B. Gill and A. Kakkar, "Improving Power Load Forecasting using FIS," IEMCON, Vancouver, BC, Canada, 2022.
[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system". Available online: https://bitcoin.org/bitcoin.pdf.
[4] N. Mangla, A. Bhat, G. Avarbratha, N. Bhat; "Bitcoin Price Prediction Using Machine Learning," International Journal of Information and Computer Science, Volume 6, Issue 5, May 2019.
[5] Jiang, X. (2020)" Bitcoin Price Prediction Based on Deep Learning Methods". *Journal of Mathematical Finance*, 10, 132-139.
[6] Livieris, I.E.; Kiriakidou, N.; Stavroyiannis, S.; Pintelas, P. "An Advanced CNN-LSTM Model for Cryptocurrency Forecasting". *Electronics* 2021, *10*, 287.
[7] T. Awoke, M. Rout, L. Mohanty, S. C. Satapathy, "BitcoinPrice Prediction and Analysis Using Deep Learning Models," ResearchGate.
[8] S. Marne, S. Churi, D. Correia, J. Gomes, "Predicting Price of Cryptocurrency - A Deep Learning Approach", International Journal of Engineering Research & Technology, Volume 9, Issue 3, 2021.
[9] H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information," in IEEE Access, vol. 6, pp. 5427-5437, 2017.
[10] R. Mittal; S. Arora; M.P.S Bhatia; "Automated cryptocurrencies price prediction using machine learning" ICTACT Journal on Soft Computing, Volume 8, Issue 4, July 2018.
[11] Y. Hua, "Bitcoin price prediction using ARIMA and LSTM" E3S Web Conf. Volume 218, Issue 01050, 2