Bitcoin Price Prediction Using LSTM

Rcik Dey[1], Saurabh Shukla[2], Sarthak Jasani[3], Hezal Lopes[4]

[1,2,3]Undergraduate Student, Dept. Of Computer Engineering, Universal College of Engineering, Mumbai, India
[4]Assistant Professor, Dept. Of Computer Engineering, Universal College of Engineering, Mumbai, India

---***---

Abstract - *Bitcoin is one of the most popular and valuable cryptocurrency in the current financial market, attracting traders for investment and thereby opening new research opportunities for researchers. Countless research works have been performed on Bitcoin price prediction with different machine learning prediction algorithms. For the research: relevant features are taken from the dataset having strong correlation with Bitcoin prices and random data chunks are then selected to train and test the model. The random data which has been selected for model training, may cause unfitting outcomes thus reducing the price prediction accuracy. Here, a proper method to train a prediction model is being scrutinised. The proposed methodology is then applied to train a simple Long Short Term Memory (LSTM) model to predict the bitcoin price for the upcoming 5 days. When the LSTM model is trained with a suitable data chunk, thus identified, sustainable results are found for the prediction. In the end of this paper, the work culminates with future improvements.*

***Key Words*: Bitcoin, Cryptocurrency, Machine Learning, Price Prediction, LSTM**

1. INTRODUCTION

Instead of any direct human investments, generating profit with the help of algorithms is a common practice in the stock market. Many case studies have been performed to reach the conclusion that mathematical models warrant better results than humans. Bitcoins are an eye catching initiative in the fields of cryptography, economics, and computer sciences, as such currencies have a special character which is gained when integrating currency units with cryptographic technology. Due to the fact that cryptocurrency has a minute history, when compared to the stock market, new and unexplored territories are thus being scouted. Structurally, both the stock market and the cryptocurrency price data are having characteristics such as time series data, but high volatility is routinely present in the latter, with heavy wavering in the prices. A cryptocurrency market differs from a traditional stock market in the respect that the former has a lot of new features. It is required to apply new techniques for prediction suitable for the cryptocurrency market. Fewer studies have been conducted on cryptocurrency price prediction when compared to the stock market. In this paper, we are predicting the Bitcoin price trend using a Long Short-Term Memory (LSTM) model. Our model is aimed to predict the next five day's price of Bitcoin.

2. REVIEW OF LITERATURE

A literature survey was carried out to find various papers published in international journals related to various Bitcoin price prediction algorithms, and associate the best algorithm for the same.

2.1 Existing Systems

Numnoda et al. [1] have obtained highly accurate results on implementing their prediction Gated Recurrent Unit (GRU) model. However, their prototype has a large time complexity. Thus, complicating the expected results in this ever-changing environment. Additionally, the selected features aren't enough to predict the Bitcoin prices; as various factors like social media, policies, and laws that each country announces to deal with digital currency, can all play a major effect on the fluctuation of the Bitcoin prices.

Mangla et al. [2] have compared four different price prediction models: Recurrent Neural Networks (RNN), Logistic Regression, Support Vector Machine, and Auto Regressive Integrated Moving Average (ARIMA). Their major findings are that- ARIMA performs poorly for predictions extending beyond the next day. Their RNN model can accurately predict price fluctuations for up to six days. And the logistic regression model can give accurate results only if a separable hyperplane exists.

Guo et al. [3] have used a hybrid method consisting of multi-scale residual blocks and an LSTM network to predict Bitcoin price. Although, their work does not include comprehensive metrics which measure the investor's attention to more timely detection of bitcoin market volatility, therefore resulting in a less accurate prediction.

Awoke et al. [4] have considered basic deep learning models like GRU and LSTM. However, their research lacks further investigation to enhance the model accuracy by considering different parameters.

Rana et al. [5], while implementing a highly accurate LSTM model, have conducted their research on a large scale, thus making their methodology a bit complex.

3. PROBLEM STATEMENT AND PROJECT SCOPE

3.1 Problem Statement

To develop a model which can help us to predict the price of the crypto currency used (in this case: Bitcoin), with low error rate and a high precision of accuracy. The model will not tell the future, but it might forecast the general trend and the direction to expect the prices to move.

3.2 Project Scope

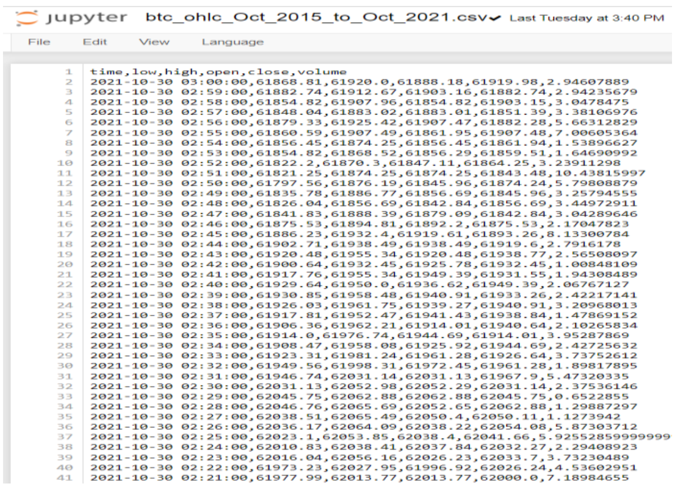
While using this model, first, the dataset of the crypto currency used needs to be uploaded. This, usually, contains the various features that the prediction model has to depend on. For e.g. average block size, total number of Bitcoins mined, day high & day low (highest and lowest values of different days), number of transactions, trade volume, etc. Then, secondly, the dataset will be applied on the regression model to obtain the predicted price.

What the model proposes to do is that, first the data on Bitcoin Price fluctuations is gathered, of the past couple of years, from the internet. Then, after the process of data acquisition, the database should be organised. The database is divided into various spreadsheet files, which are then uploaded to the software mainly used for data processing. The necessary calculations, like classification and regression, are then done. And finally the results are evaluated in terms of accuracy, error rates involved.

4. DESIGN DETAILS

4.1 System Architecture

Firstly, we collect the data set from the online source: Kaggle. The data set represents the Bitcoin price in United States Dollars (USD). The dataset includes all the information about bitcoin prices from 27th October, 2015 to 30th October, 2021.


jupyter btc_ohlc_Oct_2015_to_Oct_2021.csv✓ Last Tuesday at 3:40 PM
File
Edit
View
Language

```
time,low,high,open,close,volume
2021-10-30 03:00:00,61868.81,61920.0,61888.18,61919.98,2.94607889
2021-10-30 02:59:00,61882.74,61912.67,61903.16,61882.74,2.94235679
2021-10-30 02:58:00,61854.82,61907.96,61854.82,61903.15,3.0478475
2021-10-30 02:57:00,61848.04,61883.02,61883.01,61851.39,3.38106976
2021-10-30 02:56:00,61879.33,61925.42,61907.47,61882.28,5.66312829
2021-10-30 02:55:00,61860.59,61907.49,61861.95,61907.48,7.00605364
2021-10-30 02:54:00,61856.45,61874.25,61856.45,61861.94,1.53896627
2021-10-30 02:53:00,61854.82,61868.52,61856.29,61859.51,1.64690992
2021-10-30 02:52:00,61822.2,61870.3,61847.11,61864.25,3.23911298
2021-10-30 02:51:00,61821.25,61874.25,61874.25,61843.48,10.43815997
2021-10-30 02:50:00,61797.56,61876.19,61845.96,61874.24,5.79808879
2021-10-30 02:49:00,61835.78,61886.77,61856.69,61845.96,3.25794555
2021-10-30 02:48:00,61826.04,61856.69,61842.84,61856.69,3.44972911
2021-10-30 02:47:00,61841.83,61888.39,61879.09,61842.84,3.04289646
2021-10-30 02:46:00,61875.53,61894.81,61892.2,61875.53,2.17047823
2021-10-30 02:45:00,61886.23,61932.4,61919.61,61893.26,8.13300784
2021-10-30 02:44:00,61902.71,61938.49,61938.49,61919.6,2.7916178
2021-10-30 02:43:00,61920.48,61955.34,61920.48,61938.77,2.56508097
2021-10-30 02:42:00,61900.64,61932.45,61925.78,61932.45,1.00848109
2021-10-30 02:41:00,61917.76,61955.34,61949.39,61931.55,1.94308489
2021-10-30 02:40:00,61929.64,61950.0,61936.62,61949.39,2.06767127
2021-10-30 02:39:00,61930.85,61958.48,61940.91,61933.26,2.42217141
2021-10-30 02:38:00,61926.03,61961.75,61939.27,61940.91,3.20968013
2021-10-30 02:37:00,61917.81,61952.47,61941.43,61938.84,1.47869152
2021-10-30 02:36:00,61906.36,61962.21,61914.01,61940.64,2.10265834
2021-10-30 02:35:00,61914.0,61976.74,61944.69,61914.01,3.95287869
2021-10-30 02:34:00,61908.47,61958.08,61925.92,61944.69,2.42725632
2021-10-30 02:33:00,61923.31,61981.24,61961.28,61926.64,3.73752612
2021-10-30 02:32:00,61949.56,61998.31,61972.45,61961.28,1.89817895
2021-10-30 02:31:00,61946.74,62031.14,62031.13,61967.9,5.47320335
2021-10-30 02:30:00,62031.13,62052.98,62052.29,62031.14,2.37536146
2021-10-30 02:29:00,62045.75,62062.88,62062.88,62045.75,0.6522855
2021-10-30 02:28:00,62046.76,62065.69,62052.65,62062.88,1.29887297
2021-10-30 02:27:00,62038.51,62065.49,62050.4,62050.11,1.1273942
2021-10-30 02:26:00,62036.17,62064.09,62038.22,62054.08,5.87303712
2021-10-30 02:25:00,62023.1,62053.85,62038.4,62041.66,5.9255285999999
2021-10-30 02:24:00,62010.83,62038.41,62037.84,62032.27,2.29408923
2021-10-30 02:23:00,62016.04,62056.16,62026.23,62033.7,3.73230489
2021-10-30 02:22:00,61973.23,62027.95,61996.92,62026.24,4.53602951
2021-10-30 02:21:00,61977.99,62013.77,62013.77,62000.0,7.18984655
```

Fig -1: Bitcoin Dataset

The second step involves filtering and cleaning the data set. This involves removing all the incomplete data from the rows. It also involves filtering out unnecessary features present in the data collected. For our model, we will only use the columns labelled: Date, Price, Open, High, and Low, as shown in Table 1 below.

Table -1: Features Used In the Dataset

Features Used In The Dataset			
Sr. No.	Variable Name	Variable Description	Data Type
1	Time	Date and time of observation	Date
2	Volume	Sum total of trades taking place.	Numbe r
3	Open	Opening price on the given day.	Numbe r
4.	High	Highest price on the given day.	Numbe r
5.	Low	Lowest price on the given day.	Number
6.	Close	Closing price on a given day.	Number

The next step is training, followed by testing the dataset. We train our model, using the algorithm and the features taken into account to assist our model, to predict the future price of the crypto currency. Moving on to the testing part, we test the data to measure the accuracy of the algorithm that our model is using to predict the price of the Bitcoin.

Finally after the processes of training with the help of the data set features and testing, we evaluate the accuracy of our model. We compare the predicted price of the crypto currency, at a given time period with the real world Bitcoin price at that particular period of time, and evaluate the accuracy and efficiency of our model.

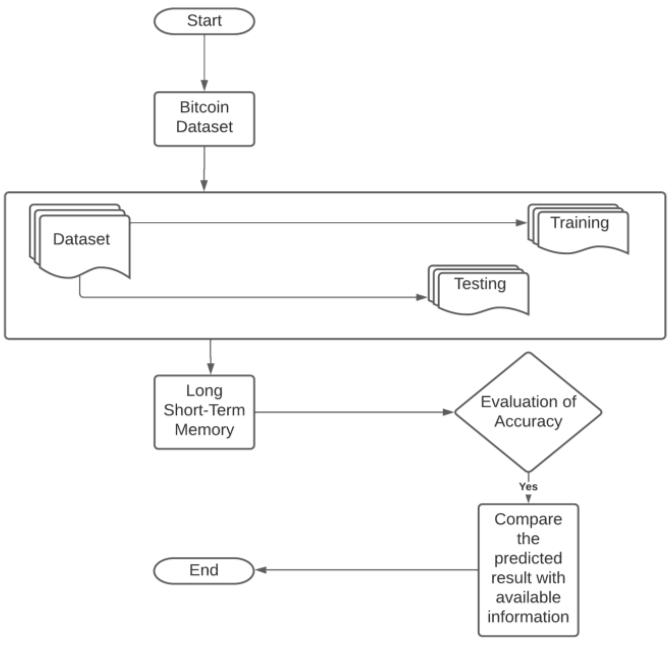


Fig -2: System Architecture

5. MODEL IMPLEMENTATION

5.1 Lag Plots

After the dataset has been filtered and cleaned, we need to generate a lag plot of the time series data. A lag in a time series data defines how much a data point is falling behind in time from another data point. Lag plots are put into use to analyse and find out whether the time series data follows any pattern. They are essential for searching patterns like trends, randomness, and seasonality. The plot can be brought about by the representation of time series data in x-axis, and the lag of the time series data points in the y-axis. We are plotting lag plots for a minute lag, an hourly lag, daily lag, weekly lag, and a monthly lag. The lag plots are represented in Figure 3.

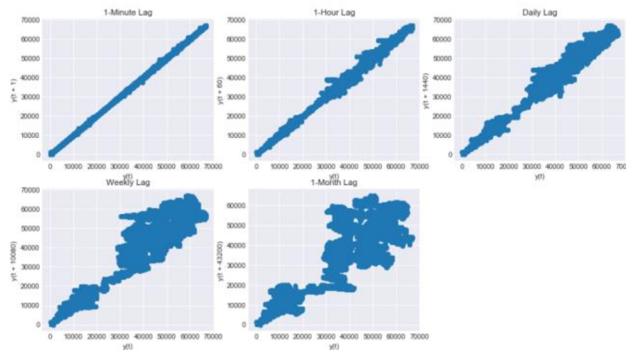


Fig -3: Lag Plots

We can see that there is a positive correlation for minute, hour, and daily lag plots. Correlation decreases greatly with weekly lag, with almost no correlation for monthly lag plots. Thus, it makes sense to re-sample the data at most at daily level, thereby preserving the autocorrelation as well.

5.2 Train-Test Split

Now, the next step that is needed to be performed is train-test split. For our model, we will be considering sixty numbers of data samples for implementing the testing, and the rest of the re-sampled data as the training sample. We will follow this by plotting a graph of the train-test split. Figure 4 represents a simple train-test plot of the closing prices.

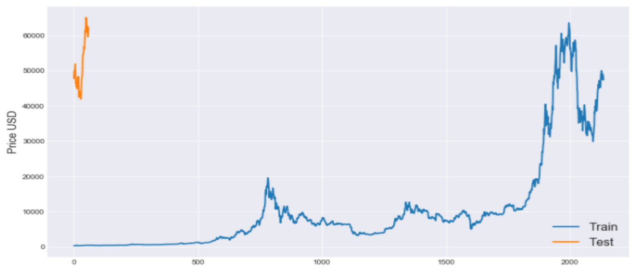


Fig -4: Graph of Train-Test Split

5.3 Scaling

Following this, we are going to scale the data, as we need the training and the test set to be scaled. One important point that needs to be mentioned is that the scaling should be performed after the train-test split has been performed, because scaling before the train-test split would introduce data leakage from the test set to the training set. Scaling before train-test split would result in the training process getting influenced by the test set, thus resulting in a bad prediction from the model.

5.4 Data Generator

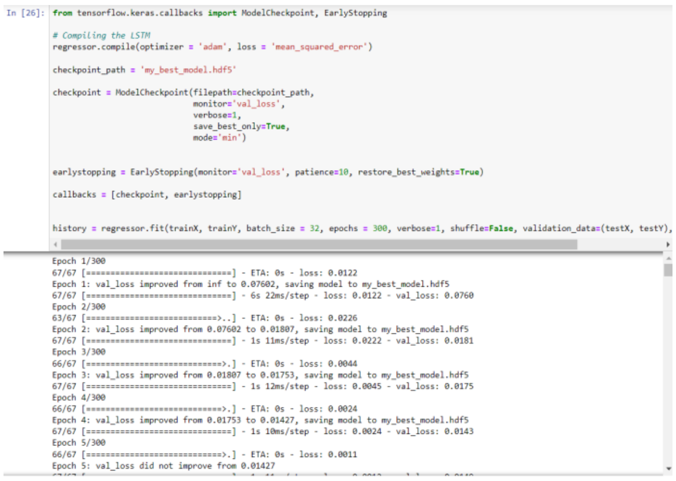
Post scaling, we will proceed with making the data generator function, which will make the data ready for the LSTM model feed. We frame our model, using a "lookback" period to take a window of the last five days of data to predict the data of the current day. A new function is defined, which will split the input sequence into windows of data appropriate for fitting a LSTM model. We need to define a lookback period which tells us how many previous timesteps are used to predict the subsequent timestep.

5.5 Restructuring Input into a shape of 3D Tensor

For LSTM, we have to reshape the input data into the shape of a three dimensional Tensor of samples, timesteps, and features. Samples are the amount of data points that we are having. A sample consists of multiple timesteps, which define the width of the sliding window. It should be noted that timesteps are different from the sliding step of the sliding window. Thus, timesteps is equivalent to the number of time steps we are to be running our RNN. Finally, features include the amount of features in every timestep.

5.6 Generating the epochs

From the callback module of the keras library we are importing the callbacks: ModelCheckpoint, and EarlyStopping. These callbacks are used as a best practice to save the model at various checkpoints or after each epoch. Also, EarlyStopping is used to stop the training when the best loss is reached, that is, when a monitored metric has stopped improving.

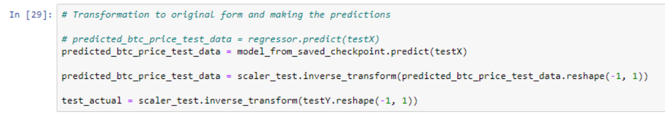


```
In [26]: from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

         # Compiling the LSTM
         regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

         checkpoint_path = 'my_best_model.hdf5'

         checkpoint = ModelCheckpoint(filepath=checkpoint_path,
                                      monitor='val_loss',
                                      verbose=1,
                                      save_best_only=True,
                                      mode='min')


         earlystopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

         callbacks = [checkpoint, earlystopping]


         history = regressor.fit(trainX, trainY, batch_size = 32, epochs = 300, verbose=1, shuffle=False, validation_data=(testX, testY),

         Epoch 1/300
         67/67 [==============================] - ETA: 0s - loss: 0.0122
         Epoch 1: val_loss improved from inf to 0.07602, saving model to my_best_model.hdf5
         67/67 [==============================] - 6s 22ms/step - loss: 0.0122 - val_loss: 0.0760
         Epoch 2/300
         63/67 [===========================>..] - ETA: 0s - loss: 0.0226
         Epoch 2: val_loss improved from 0.07602 to 0.01807, saving model to my_best_model.hdf5
         67/67 [==============================] - 1s 11ms/step - loss: 0.0222 - val_loss: 0.0181
         Epoch 3/300
         66/67 [============================>.] - ETA: 0s - loss: 0.0044
         Epoch 3: val_loss improved from 0.01807 to 0.01753, saving model to my_best_model.hdf5
         67/67 [==============================] - 1s 12ms/step - loss: 0.0045 - val_loss: 0.0175
         Epoch 4/300
         66/67 [============================>.] - ETA: 0s - loss: 0.0024
         Epoch 4: val_loss improved from 0.01753 to 0.01427, saving model to my_best_model.hdf5
         67/67 [==============================] - 1s 10ms/step - loss: 0.0024 - val_loss: 0.0143
         Epoch 5/300
         66/67 [============================>.] - ETA: 0s - loss: 0.0011
         Epoch 5: val_loss did not improve from 0.01427
```

Fig -5: Code for the generation of epoch

5.7 LSTM Prediction using testX and plotting line graph against actual testY

Due to scaling done earlier with the help of MinMaxScaler, the predicted scale will be between zero and one. We have to transfer this scale to the original data scale. Thus, we are going to use inverse transformation to scale back the data to the original presentation, as shown in Figure 6.



```
In [29]: # Transformation to original form and making the predictions

         # predicted_btc_price_test_data = regressor.predict(testX)
         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(-1, 1))

         test_actual = scaler_test.inverse_transform(testY.reshape(-1, 1))
```

Fig -6: 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 7. Here, we can see that the predicted test data and actual test data are moving in tandem.

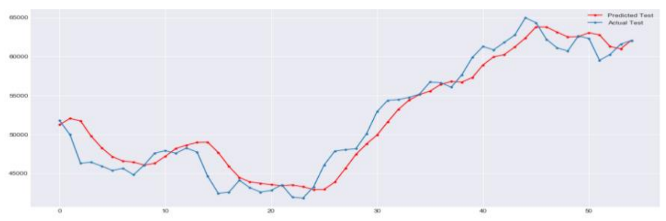
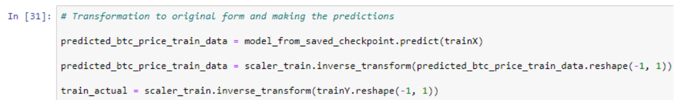


Fig -7: Plot of predicted test and actual test data

5.8 LSTM Prediction using trainX and plotting line graph against actual trainY

This step is similar to the previous step, except the fact that we are performing inverse transformations on the train data.



```
In [31]: # Transformation to original form and making the predictions

         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(-1, 1))

         train_actual = scaler_train.inverse_transform(trainY.reshape(-1, 1))
```

Fig -8: Inverse transformation on train data

Following that, we will generate the plot for predicted train data (in red) and the actual train data (in blue). In Figure 9 we can see that the graph for predicted train and actual train are quite synced as they are trained data points.

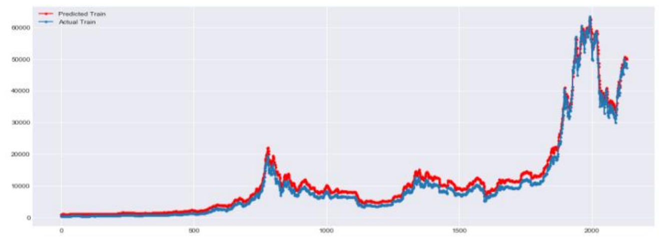
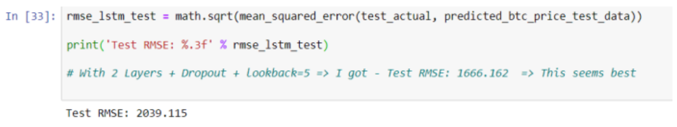


Fig -9: Plot of predicted train and actual train data

5.9 Root Mean Square Error

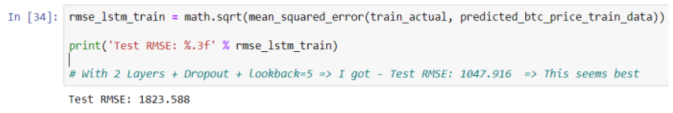
Finally, we will be generating the root mean square error (RMSE) for both the test and the train data. RMSE is the measure of how well a regression line will fit the data points.



```
In [33]: rmse_lstm_test = math.sqrt(mean_squared_error(test_actual, predicted_btc_price_test_data))

         print('Test RMSE: %.3f' % rmse_lstm_test)

         # With 2 Layers + Dropout + lookback=5 => I got - Test RMSE: 1666.162  => This seems best

         Test RMSE: 2039.115
```

Fig -10: RMSE of test data

Following this, we also generate the RMSE of train data. The RMSE loss achieved for train data is much lesser compared to the RMSE loss for test data, because the whole training and fit function was run on the training data set.



```
In [34]: rmse_lstm_train = math.sqrt(mean_squared_error(train_actual, predicted_btc_price_train_data))

         print('Test RMSE: %.3f' % rmse_lstm_train)

         # With 2 Layers + Dropout + lookback=5 => I got - Test RMSE: 1047.916  => This seems best

         Test RMSE: 1823.588
```

Fig -11: RMSE of train data

6. RESULTS

Now that we have a trained LSTM model on historical data, we are generating predictions on Bitcoin prices for the future five days. From the dataset that we use for the model, the Bitcoin price on 30th October, 2021 is the last historical price that we are having. Thus now, we are going beyond that date to predict the Bitcoin prices on the next five days. It should also be noted that we are again using the lookback period to predict the future price of the next day. Here, the lookback period is set to five days, that is, using the information on the Bitcoin prices of the immediately preceding five days, we are predicting the Bitcoin price for the next day.

Our model is implementing sixty numbers of data points for testing. It should be noted that the testX has been reshaped into a three dimensional array in the form of samples, timesteps, and features, since the LSTM needs the input to be fed into its model. We are using the last five elements in the three dimensional tensor.

Thus, we are looping this process five times, with each iteration generating the predicted price for the upcoming five days consecutively.

Finally, we are generating a graph of the entire prediction of the test data (including the future five days) against actual testY, as shown in Figure 12. Up to 30th October, 2021, we book the predicted test data (in red) and the actual test data (in blue) on the ground, because this is the time period for which we have the actual ground truth. Beyond the aforementioned date we are having only the forecasted price of Bitcoin.

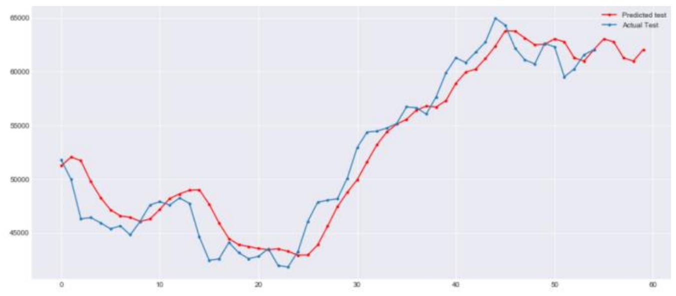


Fig -12: Plot of the entire test data prediction (including the five future days) and actual testY

7. CONCLUSIONS AND FUTURE WORK

The LSTM model, implemented here, is a basic model that takes into consideration only a few features that affect the Bitcoin price. Our model is fairly accurate when predicting the future prices. However, to increase the efficiency of the model, more Bitcoin price features need to be taken into consideration. We recommend using Kaggle as the source of datasets, since information present in this website holds a high degree of authenticity. Our future work would include in-depth scrutinisation on the topic of LSTM, and deep learning at large. Such fact-findings would be beneficial for forecasting the prices of cryptocurrencies with the help of LSTMs, in the future.

ACKNOWLEDGEMENT

We take this opportunity to extend our deep and humble gratitude to our project guide: Prof. Hezal Lopes, and project organiser: Prof. Vishakha Shelke, for their continuous support & guidance to help us navigate through our Major Project work. It is on account of their expertise and tremendous knowledge that we have accomplished the Major Project in the prescribed time. We would also like to extend our thanks to Dr. Jitendra Saturwar, the Head of the Computer Engineering Department, for his steadfast guidance and support. Additionally, we would also like to extend our thanks to our Principal: Dr. J. B. Patil, and the management faculty of Universal College of Engineering, Vasai, for providing us with all the necessary facilities and a work friendly environment. We tend to acknowledge with humble appreciation, the support provided by the division employees, library and the research lab attendants.

REFERENCES

[1] T. Phaladisailoed, and T. Numnoda, "Machine Learning Models Comparison for Bitcoin Price Prediction," 10th International Conference on Information Technology and Electrical Engineering, 2018.

[2] Neha Mangla, Akshay Bhat, Ganesh Avarbratha, and Narayana Bhat, "Bitcoin Price Prediction Using Machine Learning," International Journal of Information and Computer Science, Volume 6, Issue 5, May 2019.

[3] Q. Guo, S. Lei, Q. Ye, Z. Fang "MRC-LSTM: A Hybrid Approach of Multi-scale Residual CNN and LSTM to Predict Bitcoin Price," MDPI, May 2021.

[4] T. Awoke, M. Rout, L. Mohanty, S. C. Satapathy, "Bitcoin Price Prediction and Analysis Using Deep Learning Models," ResearchGate.

[5] A. Rana, R. Kachchhi, J. Baradia, V. Shelke "Stock Market Prediction Using Deep Learning" International Research Journal of Engineering and Technology, Volume 8, Issue 4, April 2021.