BITCOIN PRICE PREDICTION USING DEEP LEARNING ALGORITHM LSTM

Koushik Pabbathireddy

ITCS-5156 Fall 2021 – Lee

February 28, 2022

PRIMARY PAPER

A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market

International Conference on Electrical Engineering and Computer Science (ICECOS) 2019

Authors:

1. Ferdiansyah
2. Siti Hajar Othman
3. Raja Zahilah Raja Md Radzi
4. Deris Stiawan
5. Yoppy Sazaki
6. Usman Ependi

Abstract

This report is presented as a survey of a previous work[1]. Any assertions made within are subjective and do not represent those of the original author.

Bitcoin is a sort of cryptocurrency that has become a popular stock market investment. The stock market is influenced by a variety of risk factors. And bitcoin is one type of cryptocurrency that has been steadily rising in recent years, with occasional sharp drops without any apparent impact on the stock market. Because of the volatility, there is a demand for an automated technique to forecast bitcoin on the stock market. LSTM (Long Short Term Memory) is another form of module supplied for RNN that was later developed and popularized by many researchers, and like RNN, the LSTM also consists of modules with
recurrent consistency.

1 Introduction

Cryptocurrencies have gained widespread acceptance as a new electronic alternative exchange currency system, with significant ramifications for emerging economies and the world economy in general [5]. Because they have invaded virtually all financial activities, cryptocurrency trading is often regarded as one of the most popular and promising types of profitable investments. Nonetheless, this ever-expanding financial sector is marked by substantial volatility and sharp price swings over time. Cryptocurrency forecasting is now widely regarded as one of the most difficult time-series prediction issues due to the vast

number of unknown variables involved and the high volatility of cryptocurrency prices, resulting in complex temporal dependencies.

There are about 5,000 cryptocurrencies on the market as of April 2020. Bitcoin is the most well-known and notable of these. On August 18, 2008, Satoshi Nakamoto registered the domain name bitcoin.org and later that year published a whitepaper titled Bitcoin: A Peer-to-Peer Electronic Cash System under the alias Satoshi Nakamoto. Bitcoin is based on a peer-to-peer network that relies on Blockchain technology to validate transactions rather than depending on central authorities. Bitcoin has been the most influential cryptocurrency since its inception[5].

One of the reasons for Bitcoin's popularity is that it can be easily exchanged and spent anywhere in the world with a cheap transaction charge. Bitcoins may be purchased and sold using both online exchanges and bitcoin ATMs.

1.1 Problem Statement

Bitcoin is a type of cryptocurrency that is unregulated and decentralized. Bitcoin's one-of-a-kind feature is its daily price changes, which alter on a regular basis. To address the fluctuations, an automation tool for prediction is required, therefore in this project, we construct a model that forecasts the bitcoin price.

1.2 Motivation

Stocks and cryptocurrency are hot right now, and they provide opportunities for people who wish to save money and invest it carefully to reap considerable benefits. We choose to examine our Trend Analysis to better understand how stocks and cryptocurrency trading function and how to invest effectively for ourselves and the public.
Bitcoin is a type of cryptocurrency that is unregulated and decentralized. Bitcoin's one-of-a-kind feature is its daily price changes, which alter on a regular basis.
To address the fluctuations, an automation tool for prediction is required, therefore in this project, we construct a model that forecasts the bitcoin price.

1.3 Brief overview of the approach to address the challenge

We'll use LSTM to train a model that uses previous Bitcoin price data as input. To test the model, the data is separated into train and test sets. We'll preprocess the data before predicting the bitcoin price.

We'll compare expected and target data and visualize the results with graphs. The model's efficiency in predicting the Price then is evaluated using measures.

2 Related Works and Backgrounds

2.1 Paper 1[2]: Recurrent Neural Network Based Bitcoin Price Prediction by Twitter Sentiment Analysis by Dibakar Raj Pant, Prasanga Neupane, Anuj Poudel, Anup Kumar Pokhrel, Bishnu Kumar Lama I.

This paper states about RNN which is considered LSTM (Long Short-Term Memory) is another type of RNN. The paper also uses Sentimental analysis to predict the Bitcoin price apart from the historical data. Using Sentimental analysis, it takes twitter tweets into account and predict if the price goes up or down. The main improvement from RNN is that it cannot hold memory over time which is used to for learning long- term temporal dependencies. This is changed in LSTM by including special memory cells.

2.2 **Paper 2[3]:** Bitcoin Price Prediction Based on Deep Learning Methods by Xiangxi Jiang

The paper provides comparison between various methods comparing the bitcoin price prediction. Below I have provided clear explanation for two methods and their relationship with LSTM my primary method.

Conventional Feed-Forward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from its descendant: recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised.

A feedforward neural network is a classification algorithm that is biologically inspired. It is made up of a (potentially large) number of basic neuron-like processing units that are layered together[3]. Every unit in a layer is linked to all of the units in the layer before it. It's for this reason that they're known as feedforward neural networks. The data or input provided in this ANN only flows in one direction. It enters the ANN via the input layer and exits via the output layer, with hidden layers present or absent.

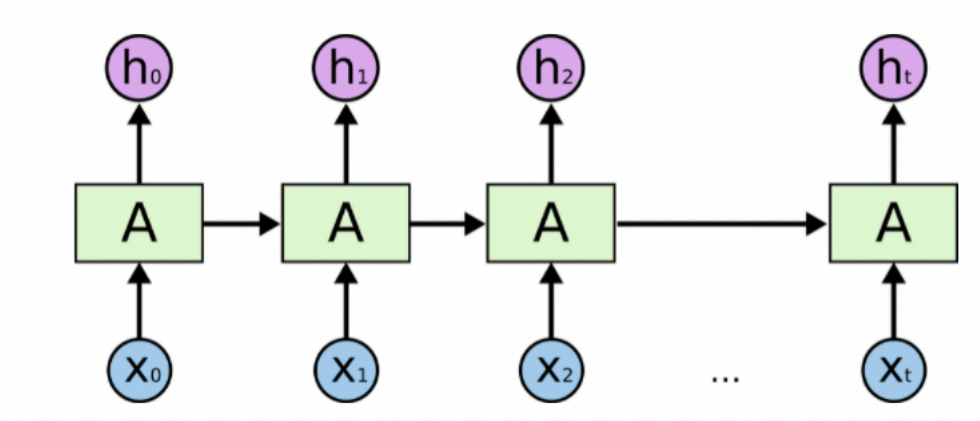

Figure 1: Conventional Feed forward Neural Network[3]

Cons:

1. Feed-forward the information in a neural network can only flow in one direction: from the input layers to the hidden levels to the output layers.
2. Because these Neural Networks have no memory of prior input, it is impossible to forecast what will be received next.

Relation with LSTM's: The feedforward neural network was the first and simplest type of artificial neural network devised.

2.3 Recurrent neural networks

The recurrent neural network (RNN) is a type of artificial neural network or advanced artificial neural network that uses direct memory cycles. Recurrent neural networks can create networks with pre-determined input and output sizes. The information in an RNN loops back on itself. As a result, RNN examines not only the current input but also past inputs[3].

They are networks with loops in them, allowing information to persist.

Recurrent neural networks appear enigmatic because of these loops. However, if you think about it, they're not all that different from a traditional neural network. A recurrent neural network is made up of several copies of the same network, each sending a message to the next.

Recurrent neural networks are intricately tied to sequences and lists, as seen by their chain-like character. They're the most natural neural network architecture to employ for such data.

And they're surely put to good use! RNNs have had remarkable success in the previous several years when applied to a range of tasks, including speech recognition, language modeling, translation, image captioning, and so on. The list could go on and on. I'll defer to Andrej Karpathy's outstanding blog piece, The Unreasonable Effectiveness of Recurrent Neural Networks, for a description of the incredible feats that RNNs can do. They are, nevertheless, very remarkable[3].

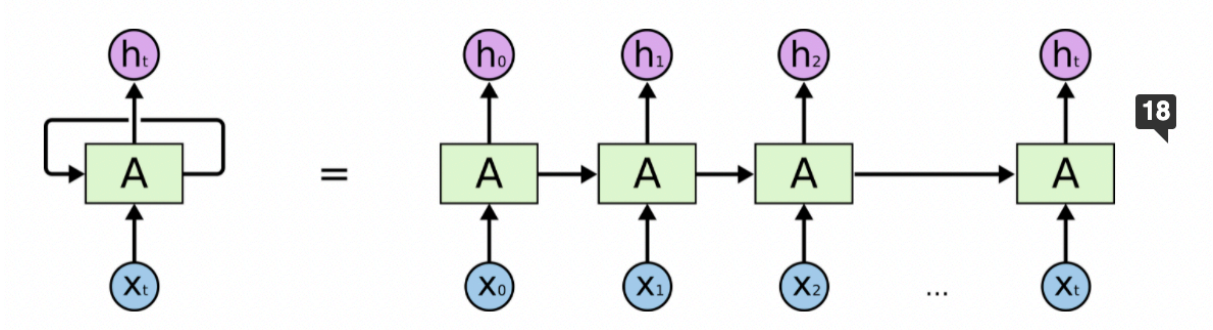

Figure 2: Recurrent Neural Network [3]

Pros:

1. They have memory cells to memorize previous inputs which makes it to predict the Data
2. They have the Back propagation which allows the model to change the weights through reiterating and increasing the efficiency of Model.

Relation with LSTM's: Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies

3 The Method

LSTM stands for Long Short-Term Memory is an artificial recurrent neural network.

In the field of deep learning, the (RNN) architecture is used. A cell, an input gate, an output gate, and a forget gate make up a typical LSTM unit. The three gates control the flow of information in and out of the cell, and the cell remembers value across arbitrary time interval.

3.1 Method Explanation and Mathematical Depiction

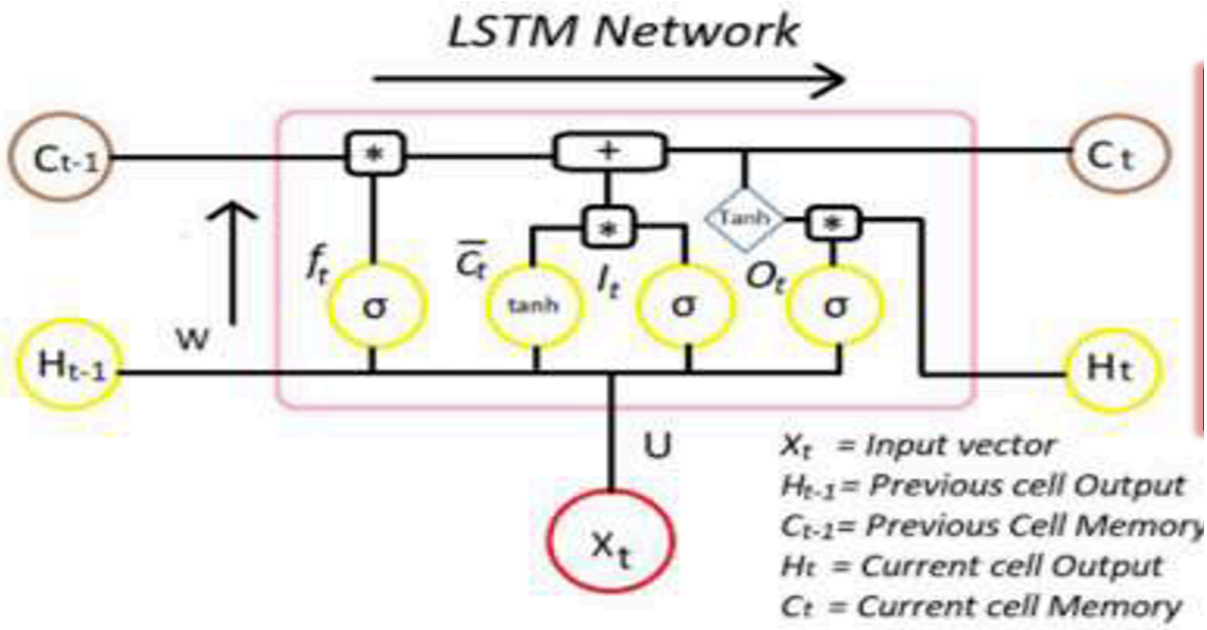

Figure 3: LSTM Neural Network [1]

First Step:

The first stage in our LSTM is to decide which information from the cell state will be discarded. The "forget gate layer," a sigmoid layer, makes this judgment. It examines the values in ht-1 and xt and returns a number between 0 and 1 for each number in the cell state Ct-1. 1 indicates totally keep this, while a 0 indicates entirely discard this.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

[4]

Second Step:

In this step it decides what new data will we store in the cell state. There are two components to this. The "input gate layer," a sigmoid layer, chooses which values we'll update first. A tanh layer then generates a vector of new candidate values, C t, that can be added to the state. We'll combine these two in the next step to make a state update.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

[4]

Third Step:

It's time to switch from the old cell state Ct1 to the new cell state Ct. We already know what to do because of the previous steps; now we just have to perform it.

We magnify the previous state by foot, forgetting the items we had previously agreed to forget. Then we add it * C t to the equation. This is the new set of candidate values, scaled by how much each state value was updated.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

[4]

Last Step:

This output will be based on the state of our cells, but it will be filtered. First, we run a sigmoid layer to determine which aspects of the cell state will be output. The cell state is then passed through tanh (to force the values to be between -1 and 1) and multiplied by the output of the sigmoid gate, resulting in only the parts we choose to output.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

[4]

3.2 Framework Figure

Project Flow:

1. First, we collect the historical data which will be used for predicting the Bitcoin price
2. 2.Then we pre-process it by cleaning the data
3. 3.We divide the Data to Train and Test data for model development.
4. 4.Using Training data, we develop LSTM model we predict the Bitcoin Price
5. 5.Then we evaluate the Model using Root Mean Square Error, Mean Absolute Error R Squared Metrics

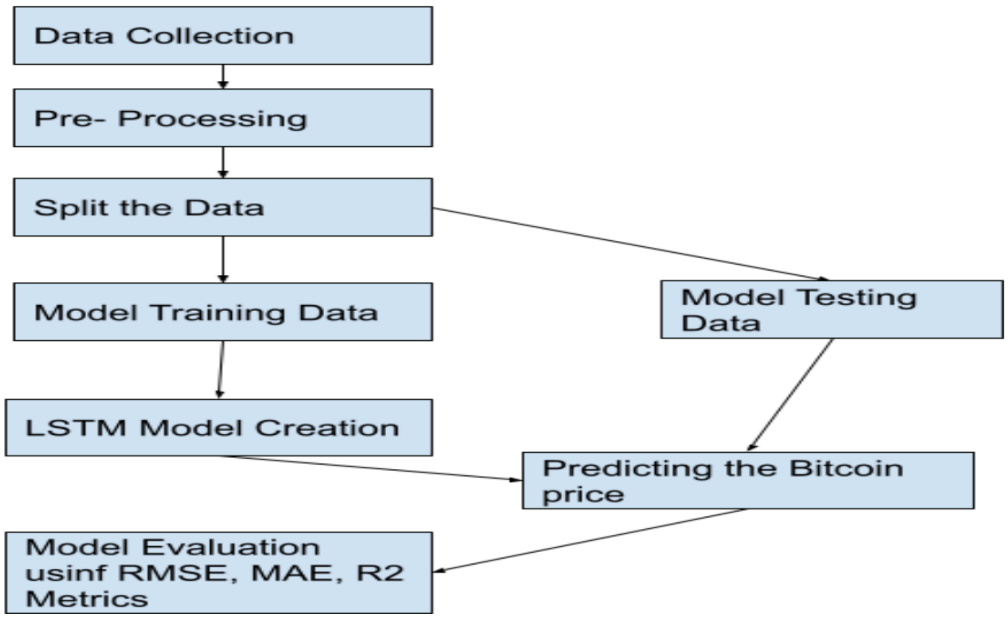

Figure 4: Framework of project Implementation

3.3 Data Preprocessing

The Data set has many null values, so I have used Linear Interpolation.

Interpolation is an imputation technique that assumes a linear relationship between data points and uses non-missing values from surrounding data points to get a value for a missing data point.

	Total Missing Values	Missing %
Timestamp	0	0.000000
Open	1241716	27.157616
High	1241716	27.157616
Low	1241716	27.157616
Close	1241716	27.157616
Volume_(BTC)	1241716	27.157616
Volume_(Currency)	1241716	27.157616
Weighted_Price	1241716	27.157616

Figure 5: Data Set Features and Missing Value

After Interpolation we can see from the below figure that null values are zero in all columns.

```
Timestamp           0
Open                0
High                0
Low                 0
Close               0
Volume_(BTC)        0
Volume_(Currency)   0
Weighted_Price      0
dtype: int64
```

Figure 6: Data Set Features and Missing Value after Interpolation

From the above I have used Feature Selection to select only two columns Timestamp and Weighted Price to predict the Bitcoin Price using the LSTM model.

3.4 Data Visualization

```
bitcoinpricdata.set_index("Timestamp").Weighted_Price.plot(figsize=(14,7), title="Bitcoin Weighted Price")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76728171d0>
```

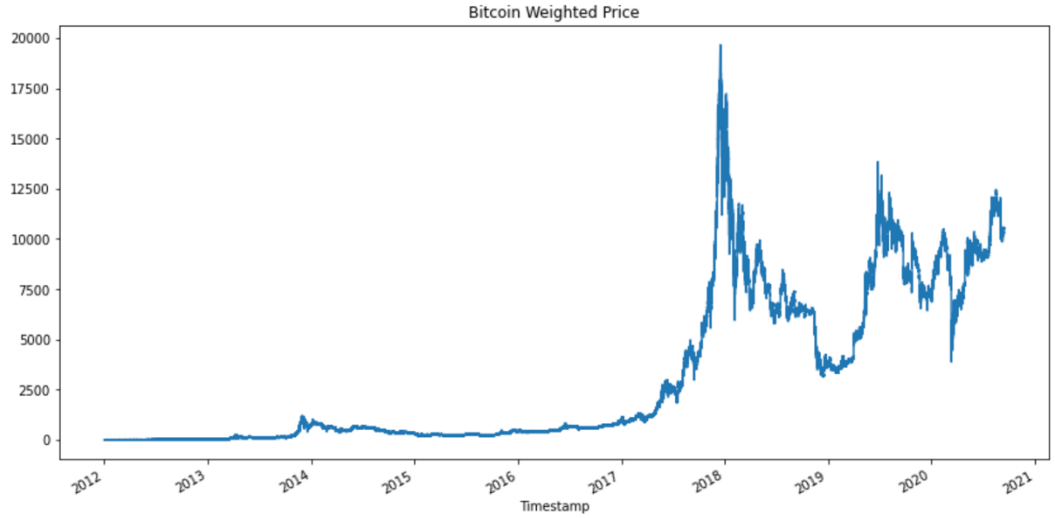

Figure 7: Bitcoin Price for the Data Set in the 2012-2020

Graph above shows the price of Bitcoin in period where we can observe the Bitcoin price have peaked in 2018 year.

4 Model Implementation and Experiments

4.1 Model Implementation

The Model Summary displays the total number of Layers of Neurons and the number of weights associated with each layer, as well as other information about the model. It also gives us the network's output layer (Dense) and total Weights.

The model has 10,400 has Parameters which is nothing but weights of Neurons. I have incorporated three LSTM layers and dense output layers.

```
[35] regressor.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 50)           20200

 dropout_1 (Dropout)         (None, 100, 50)           0

 lstm_2 (LSTM)               (None, 100, 50)           20200

 dropout_2 (Dropout)         (None, 100, 50)           0

 lstm_3 (LSTM)               (None, 50)                20200

 dropout_3 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: Developed LSTM Model Summary

4.2 Experiments Conducted

1. The Dataset consists of Data from 2012 to 2020 hourly data.
2. First I have split the Data to test (2020) and train (2012 -2019)
3. Using the model developed I have conducted experiments by varying the number of EPOCHS from 100 to 200.
4. I have then predicted Bitcoin price for two conditions.
5. I have plotted the Training and Validation Loss Graph
   I have evaluated the model using Root Mean Square Error, Mean Absolute Error R Squared Metrics for the two EPOCHS and got better result with a greater number of EPOCHS.

5 Results and Observations

Yes, I was able to use the papers LSTM method. I have used a different Dataset so got similar results but not Identical. The model has predicted the Bitcoin price with very good efficiency and almost accurately. As the Datapoints are different we just have similar result though I have duplicated the LSTM framework from the paper.

1) Results for EPOCH:100

```
[53] plt.figure(figsize=(16,7))
     plt.plot(history.history["loss"], label= "train loss")
     plt.plot(history.history["val_loss"], label= "validation loss")
     plt.ylabel("Average Loss")
     plt.title("Comparision between Predicted train and Target Train data ")
     plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9ab329bb10>
```

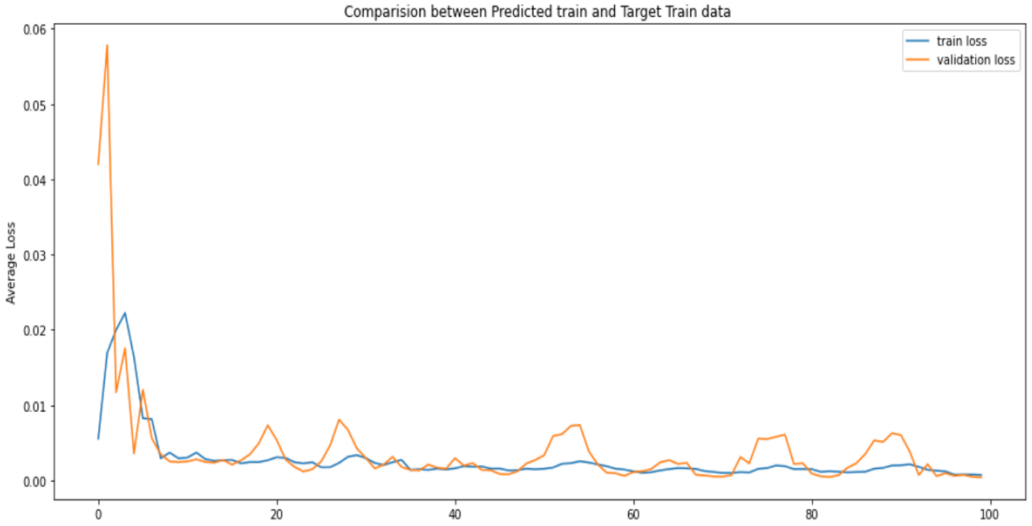

Figure 9: Comparison between Training and Validation Loss for EPOCHS 100

Observations: The training loss is used to determine how well the model fits the training data, whereas the validation loss is used to determine how well it fits new data.
We can see from the graph that both losses are nearly overlapping with a difference less than 0.01 indicating that the model is not overfitting.

```
plt.figure(figsize=(16,7))
plt.plot(y_test_inv.flatten(), marker='.', label="Actual")
plt.plot(X_test_predict_inv.flatten(), 'r', marker='.', label="Predicted")
plt.ylabel("BitCoin Price")
plt.title("Comparision between Predicted Test and Target Test data ")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f9ab674b190>
```

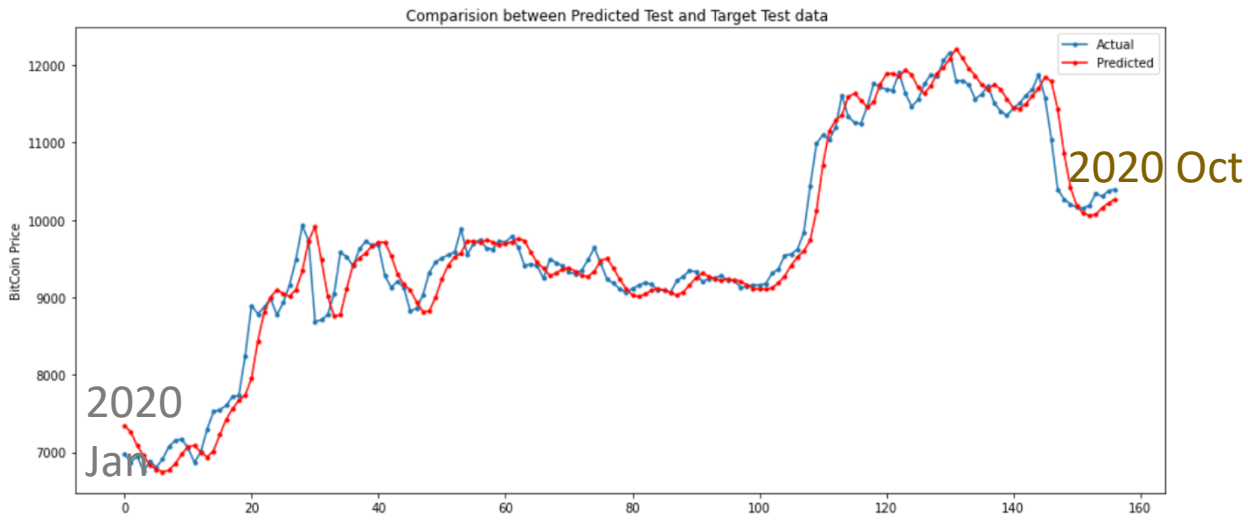

Figure 10: Comparison between Predicted and Target values EPOCHS 100

Observations: As seen in the graph below, we trained the LSTM model using 8 years of data (2012-2019) and projected the next 9 months of test data and compared it to the target variable.
The model performed admirably, as evidenced by the fact that the bitcoin price projected on the Y axis is very near to the target price.

Model Evaluation:

```
[49] from sklearn.metrics import mean_absolute_error, mean_squared_error

     train_RMSE = np.sqrt(mean_squared_error(y_train, X_train_predict))
     test_RMSE = np.sqrt(mean_squared_error(y_test, X_test_predict))
     train_MAE = np.sqrt(mean_absolute_error(y_train, X_train_predict))
     test_MAE = np.sqrt(mean_absolute_error(y_test, X_test_predict))


     print(f"Train RMSE: {train_RMSE}")
     print(f"Train MAE: {train_MAE}")

     print(f"Test RMSE: {test_RMSE}")
     print(f"Test MAE: {test_MAE}")
```

```
Train RMSE: 0.03786759968746118
Train MAE: 0.18440016332992654
Test RMSE: 0.015289931358133045
Test MAE: 0.1036809309837744
```

```
[50] from sklearn.metrics import r2_score
```

```
[51] print('R2 Score : ','{:.2%}'.format(r2_score(y_test_inv,X_test_predict_inv )))
```

```
R2 Score :  95.29%
```

Figure 11: Model Evaluation using RMSE, MAE, R2 EPOCHS 100

Observations: The Root Mean Square Error (RMSE) calculates the difference between all predicted and actual target variables squares it and provides the sum. Absolute Mean score also does the same but applies absolute function after difference. As we see our model performed great with values closer to 0 showing the accuracy predicting the Bitcoin price. I achieved 95% R2 score.

2) Results for EPOCH:200

```
plt.figure(figsize=(16,7))
plt.plot(history.history["loss"], label= "train loss")
plt.plot(history.history["val_loss"], label= "validation loss")
plt.ylabel("Average Loss")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f76399a6c10>
```

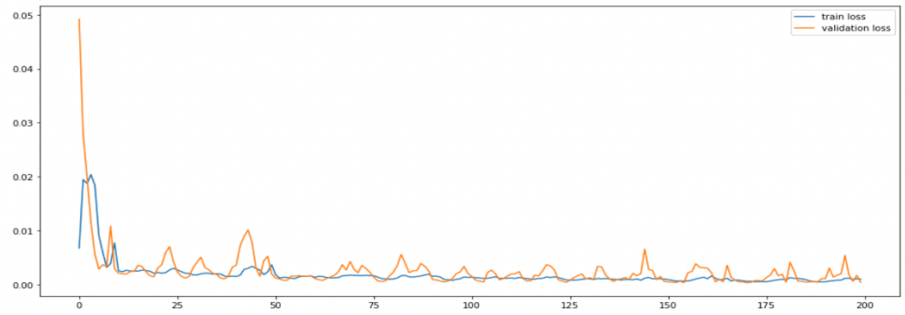

Figure 12: Comparison between Training and Validation Loss for EPOCHS 200
200

Observations:
We can clearly see that the model performed well with more EPOCHS even decreasing the loss.

```
plt.figure(figsize=(16,7))
plt.plot(y_train_inv.flatten(), marker='.', label="Actual")
plt.plot(X_train_predict_inv.flatten(), 'r', marker='.', label="Predicted")
plt.ylabel("BitCoin Price")
plt.title("Comparision between Predicted train and Target Train data ")

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f76409ca110>
```

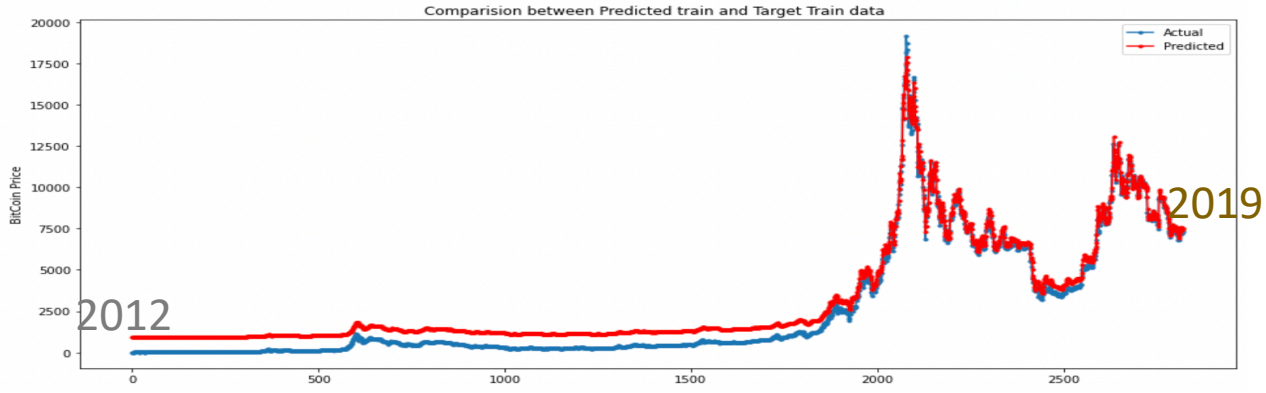

Figure 13: Comparison between trained predicted and Target values EPOCHS
200

Observations:
Above graph shows comparison between X_train predicted values and Target values. We can clearly see that the model performed well with more EPOCHS giving better predicted Bitcoin Price when compared with train Target values.

```
plt.figure(figsize=(16,7))
plt.plot(y_test_inv.flatten(), marker='.', label="Actual")
plt.plot(X_test_predict_inv.flatten(), 'r', marker='.', label="Predicted")
plt.ylabel("BitCoin Price")
plt.title("Comparision between Predicted Test and Target Test data ")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f764070f2d0>
```

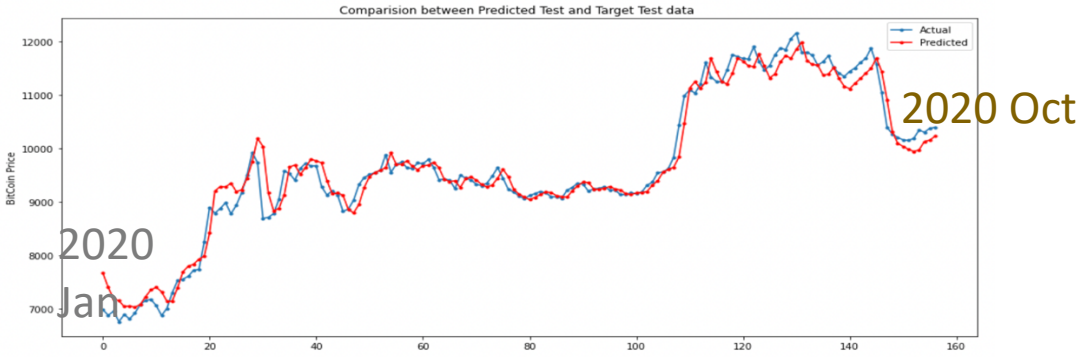

Figure 14: Comparison between Predicted and Target values EPOCHS 200

Observations:

Above graph shows comparison between X_test predicted values and Target values. We can clearly see that the model performed well with more EPOCHS giving better predicted Bitcoin Price when compared with train Target values

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

train_RMSE = np.sqrt(mean_squared_error(y_train, X_train_predict))
test_RMSE = np.sqrt(mean_squared_error(y_test, X_test_predict))
train_MAE = np.sqrt(mean_absolute_error(y_train, X_train_predict))
test_MAE = np.sqrt(mean_absolute_error(y_test, X_test_predict))


print(f"Train RMSE: {train_RMSE}")
print(f"Train MAE: {train_MAE}")

print(f"Test RMSE: {test_RMSE}")
print(f"Test MAE: {test_MAE}")
```

```
Train RMSE: 0.03858855395608676
Train MAE: 0.18863711947120684
Test RMSE: 0.013227889068732877
Test MAE: 0.09763174286017254
```

```
from sklearn.metrics import r2_score
```

```
print('R2 Score : ','{:.2%}'.format(r2_score(y_test_inv,X_test_predict_inv )))
```

```
R2 Score :  96.47%
```

Figure 15: Model Evaluation using RMSE, MAE, R2 EPOCHS 200

Observations:

Here we even got better results with more EPOCHS increasing our R Squared to 96.47% . It calculates the correlation between our predicted and target values. If the value is 100% then they are completely correlated. I Achieved 96% which is a great result.

6 Conclusion and Future Scope

To Conclude, using LSTM Neural Network trained a model that take historical Bitcoin price data as input which covered the years 2012 to 2019 data. I preprocessed the data by removing the null values and cleaned it. Using the trained model forecasted the data for the next 9 months in 2020. I have learnt LSTM model in depth and learnt new Machine Learning approaches towards real time problems.

I compared it to the target data and got fantastic results, as evidenced by the findings.
I was able to attain R Squared and Mean Square error values of 96 percent and close to 0, respectively, indicating that the model can accurately forecast the Bitcoin price.

This model can be used to predict the prices of other cryptocurrencies like Litecoin, ether etc.,
The model can be optimized further by varying the EPOCH values for better performance.
Further we can get the present-day Bitcoin price by gathering recent data and we can then decide to invest in Bitcoin and earn from it wisely.

7 Contributions

From Primary Paper[1]
I have duplicated the LSTM method from the primary paper [1]. I have taken the Dataset from Kaggle website link. I have plotted the Resultant graphs like the paper after predicting the Bitcoin price comparing it with target values. The Project repository link has all the code.

I have built most of the code with knowledge I have gained from the course lab assignments.
A. I have written the code for
1. Loading the Data,
2. Pre-processing data using imputations and cleaned the Data
3. Developed the model with layers according for better efficiency.
4. Plotted the results comparing the Predicted and target values.
5. Calculated metrics to check the efficiency of the model.

B. I have used some bits of code from Keras website to make the code work like
1. Scaling the data using Min max scaler.
2. Used Plotly from python website.

References

[1] A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market 1st Ferdiansyah Department of Informatics, Universitas Bina Darma Palembang, Indonesia ferdi@binadarma.ac.id

[2] Recurrent Neural Network Based Bitcoin Price Prediction by Twitter Sentiment Analysis Dibakar Raj Pant, Prasanga Neupane, Anuj Poudel, Anup Kumar Pokhrel, Bishnu Kumar Lama *Department of Electronics and Computer Engineering, Central Campus Pulchowk I.O.E, Tribhuvan University Lalitpur, Nepal*

[3] Bitcoin Price Prediction Based on Deep Learning Methods by Xiangxi Jiang Barstow School of Ningbo, Ningbo, China

[4] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[5] Nasir, M.A.; Huynh, T.L.D.; Nguyen, S.P.; Duong, D. Forecasting cryptocurrency returns and volume using search engines. Financ. Innov. 2019, 5, 2.