Alma mater Studiorum – University of Bologna

Bitcoin Time Series Price Forecasting

Blockchain and Cryptocurrency

Professor: Stefano Ferretti

Yellam Naidu Kottavalasa - yellam.kottavalasa@studio.unibo.it
Matricola: 0000985387
Usha Padma Rachakonda - ushapadma.rachakonda@studio.unibo.it
Matricola: 0000985891

GitHub Repository: https://github.com/yellamnaidukottavalasa555/Bitcoin-Time-Series-Price-Forecasting

Abstract

Bitcoin is a decentralized cryptocurrency, which is a type of digital asset that provides the basis for peer-to-peer financial transactions based on blockchain technology. One of the main problems with decentralized cryptocurrencies is price volatility, which indicates the need for studying the underlying price model. Moreover, Bitcoin prices exhibit non-stationary behaviour, where the statistical distribution of data changes over time. This paper demonstrates high-performance deep learning models for predicting Bitcoin close price forecasting. Particularly, this study compares the accuracy of the prediction of bitcoin close price using different deep learning models: Convolution Neural Networks (CNN), Long-Short Term Memory (LSTM), Bidirectional Long-Short Term Memory (BiLSTM), Gated Recurrent Unit (GRU) and Bidirectional Gated Recurrent Unit (BiGRU). In this dissertation, the results do confirm that GRU predictions have better precision in terms of forecasting errors compared to the other models.

1. Introduction

Digital transformation of economies is the most serious disruption that is taking place now in all economies and financial systems. The economies and financial systems of the world are becoming digital at an unprecedentedly fast pace. The most recent technology for establishing and spending digital assets is distributed ledger technology (DLT), and its most well-known application is the cryptocurrency named Bitcoin. Following these developments, blockchain technology has found its place in the intersections of Fintech and next-generation networks.

Bitcoin has sparked a gigantic interest in cryptocurrency and blockchain technology. Since the inception of Bitcoin, cryptocurrency has gained the trust of the general population. Bitcoin has achieved the highest market capitalization among all the cryptocurrencies. As of this writing Bitcoin market capitalization is more than 134 billion US dollars. Bitcoin gains this market value as there is a huge demand for this cryptocurrency. The demand for cryptocurrency directly translates into people's trust in Bitcoin and the underneath technology. Since people's trust is involved in the rise of the cryptocurrency market, the sentiment of the general population does make a huge impact on the future of cryptocurrency market capitalization.

2. Literature Review

2.1 Bitcoin

2.1.1. Pre-Bitcoin

According to Nathan Reiff's article (Reiff, 2019), one of the first attempts to create a cryptocurrency comes from the Netherlands in the late 1980s. Petrol stations were suffering from thefts and a group of people tried to link money to new smartcards instead of using cash.

At the same time, American cryptographer David Chaum introduced different electronic cash, digiCash (Chaum, 1983). He developed a "blinding formula" to encrypt information to be passed among people. This tool could allow money to be sent safely between individuals only by checking the signature authenticity. This innovation played an important role in future cryptocurrencies. Some companies applied these fundamentals in the 1990s. The company with the greatest lasting impact was PayPal. This world-known enterprise modernized person-to-person online payments. Individuals could quickly and securely transfer money via the internet. One of the most successful applications was the egold, which offered to users the chance to use money in exchange for physical gold or other metals.

In 1998, Wei Dai developed an "anonymous, distributed electronic cash system", called b-Money (Dai, 1998). This system was based on a digital pseudonym used to transfer currency through a decentralized network. The structure also included a contract in the network as

well rather than a third party. Even though it was not as successful as other cryptocurrencies were, Nakamoto (2008) referenced some points of B-money in his bitcoin whitepaper.

Developed in the mid-1990s, Hash cash (Back, 1997) was one of the most effective pre-bitcoin digital currencies. It used proof-of-work algorithms to create and distribute new coins. The Bit Gold proposed by Szabo (2008) introduced a proof-of-work system, which is used in some ways in bitcoin's mining network.

2.1.2. Bitcoin and Blockchain

A Blockchain is a distributed database of records that have been performed and shared among the involved parties. Most of the participants in the network revise each transaction. As soon as the information is entered into the system, it can never be removed. Bitcoin is the most popular example of blockchain technology.

"Bitcoin is a peer-to-peer electronic cash system" was introduced in the well-known paper Nakamoto (2008). The peer-to-peer (P2P) mechanism allows an ownership transfer from one party to another without a third-party intervention (financial institution). Payments can be

made over the internet without any control or cost of a central authority for the first time. Individuals who want to own bitcoins can either run a program on their own computer that implements bitcoin protocol or create an account on a website that runs bitcoin for its users. The bitcoins are saved in a file called a wallet, which the user may secure and backup. These programs connect to each other over the internet forming P2P networks, making the system resistant to a central attack (Zohar, 2015) and (Crosby et al., 2016).

For now, Bitcoins are generated through a process of mining. Any member operates as a miner using their computer knowledge to maintain the network. Mining is a computational process that requires miners to find a solution to a mathematical problem to create a new block in the blockchain. Miners resolve this issue by using the proof-of-work concept. This algorithm involves recurrently difficult mathematical problems until getting to a solution. The first miner to find a solution broadcast it to the network to verify it. Once verified the block is added to the blockchain. Every ten minutes on average is found a new answer and a bitcoin is created. The bitcoin protocol is designed to generate 12 new bitcoins gradually. The difficulty of solving problems is adjusted every two weeks at the rate of six blocks per hour. The size of the reward was initially 50 (genesis block) and it is halved every four years, this implies that the number of bitcoins in circulation will never exceed 21 million. Once the last bitcoin is generated, miners will instead be rewarded with transaction fees (Zohar, 2015) and (Crosby et al., 2016).

The core principle of bitcoin technology is public-key cryptography. This method determines that each transaction is protected through a digital signature and verified by every node in the network. It uses an algorithm that generates two separate keys: the public key and the private key. In this system, the sender sends his private key digitally signed to the public key of the receiver. In order to complete the transaction, the sender needs to prove his ownership and the receiver needs to verify that ownership by using the information received in the

public key. The verifying nodes must ensure that the spender owns the cryptocurrency through the digital signature and confirm that the sender has enough cryptocurrencies in his account to complete the transaction by checking his account or validating in the public key. New transactions are checked against the blockchain to make sure that the bitcoins were already spent, thus solving the double-spending problem. Each transaction has a digital signature and a hash containing all the chronological history of movements and peers that allows for easy detection in case of double-spending attacks. (Zohar, 2015) and (Crosby et al., 2016). According to Velde (2013), "bitcoin solves two challenges of digital money controlling its creation and avoiding its duplication- at once." Nakamoto (2008) showed that if attackers do not have more than 50% of the computer power of the network, the probability of a successful attack is almost null.

3. Dataset

The dataset provides the history of the daily prices of Bitcoin. The data starts from 17-Sep-2014 to 19-Feb-2022. The dataset contains Date, Open, High, Low, Close, Adj Close, and Volume columns. I suppose the price of bitcoin will keep growing in value. This is subject to proof after testing the data. So, the dataset contains 2713 rows by 7 columns. It contains information about the out-price dynamics of bitcoin against the USD that is, the opening and closing price of each day, the low and high price of each day as well as the volume traded on that day. This data was obtained from yahoo finance (https://finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD). However, this is real-time data that keeps updating. For example, if someone accesses it after some time, he/she will find that it has added on depending on the time of access. But the previous data will remain constant.

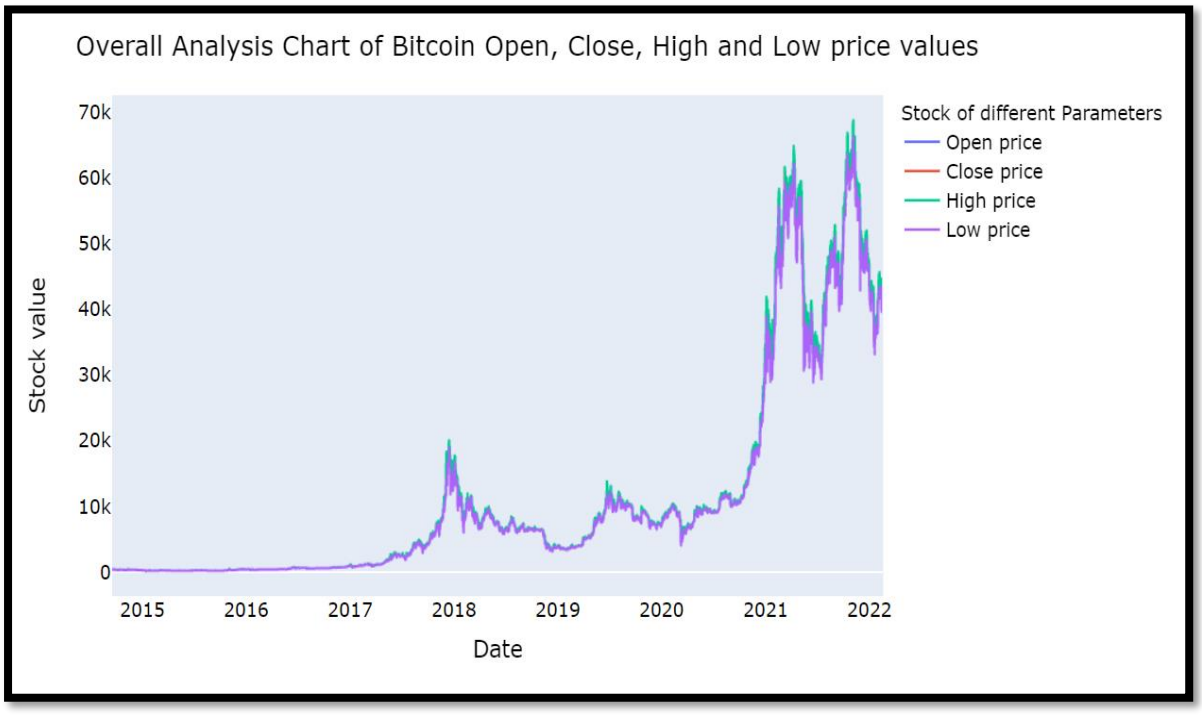


Figure: Overall comparison graph of bitcoin Open, Close, High, and Low price values

4. Forecasting Models Background

4.1. Time Series Forecasting

A time series is a sequence of observations measured sequentially through time. The observations at different moments in the series correlate with each other in different ways. Since the successive observations are dependent, future values may be predicted from the past ones. “Forecasting is about predicting the future as accurately as possible, given all of the information available, including historical data and knowledge of any future events that might impact the forecasts”, as defined by Hyndman and Athanasopoulos (2018).

Time-series forecasting enables us to predict likely future values for a dataset based on historical time-series data. Time-series data collectively represents how a system, process, or behaviour changes over time. When you accumulate millions of data points over a time period, you can build models to predict the next set of values likely to occur.

There are several models used for time series forecasting that can be classified as linear and non-linear, univariate, and multivariate. Linear models are restricted by their assumption of linear behaviour in parameters and/or variables, whereas non-linear models may fit more complex dynamics and functions accurately. Here we are using MLP, CNN, LSTM, Bidirectional LSTM, GRU, and Bidirectional GRU to forecast the bitcoin values.

When choosing forecasting models, it is important to separate the available data into two parts: training and test data, where training is used to estimate the parameters of a forecasting method, and test is used to evaluate its accuracy.

The accuracy of forecasts can only be determined by considering how well a model performs on new data that were not used when fitting the model. It is possible to measure the forecasting accuracy by estimating the forecasting errors, which is the difference between the observed value and its forecast, calculated in the testing set. When comparing data from the same scale, as happens with bitcoin prices, scale-dependent errors are commonly used such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

5. Forecasting Models

5.1 Multilayer Perceptron

The Multilayer Perceptron was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function.

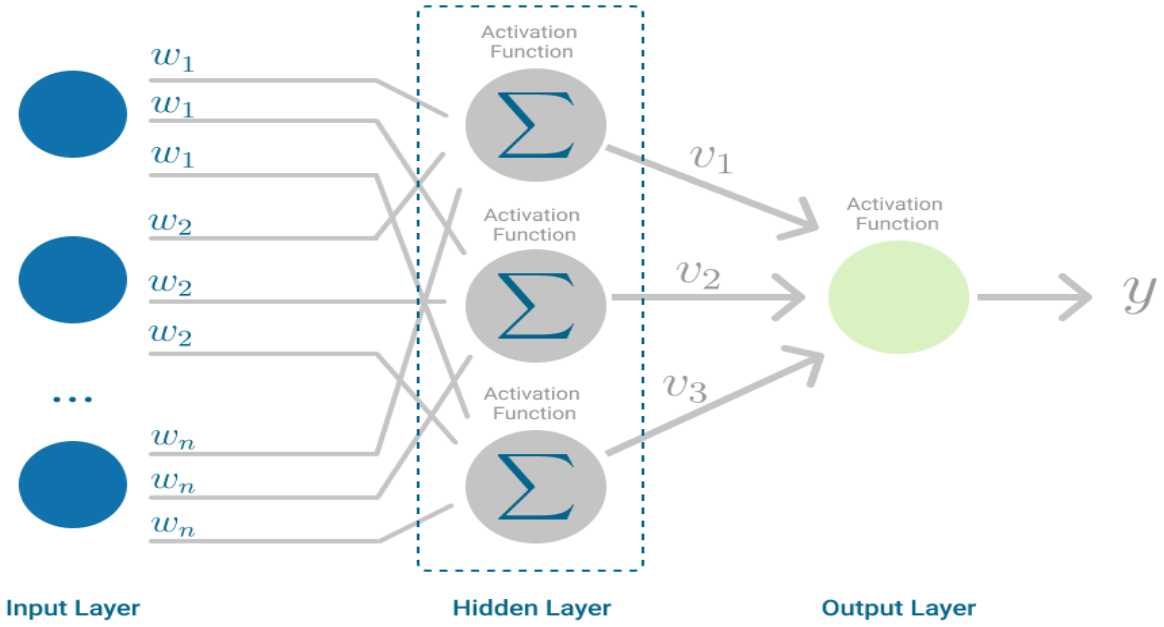


Figure-1. Multilayer Perceptron

Multilayer Perceptron falls under the category of feedforward Algorithm because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

5.2 Convolutional Neural Network

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

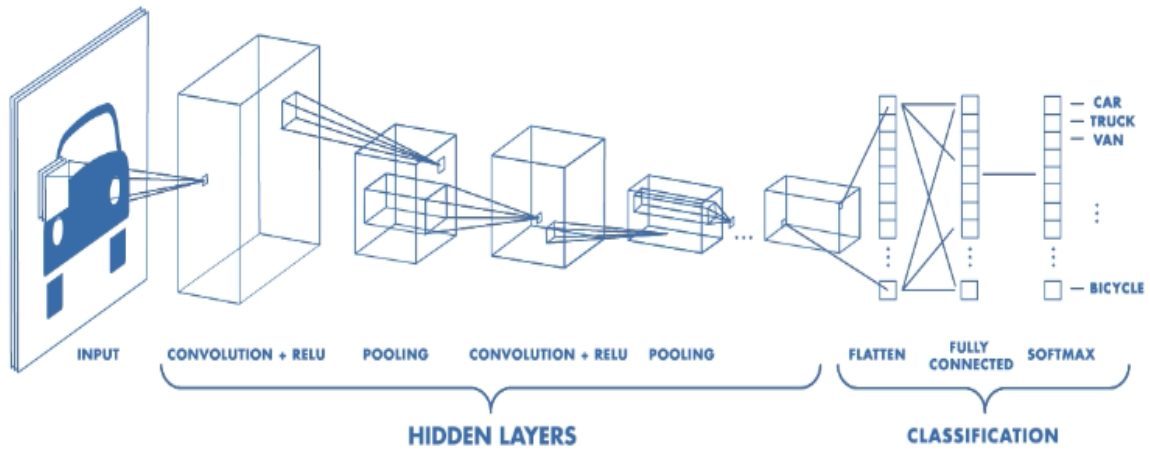


Figure-2. Convolution Neural networks

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load. The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

Since convolution is a linear operation and images are far from linear, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map. Here we are using ReLU activation function. The Rectified Linear Unit (ReLU) has become very popular in the last few years. It computes the function $f(\kappa)=\max(0, \kappa)$. In other words, the activation is simply threshold at zero. In comparison to sigmoid and tanh, ReLU is more reliable and accelerates the convergence by six times. Unfortunately, a con is that ReLU can be fragile during training. A large gradient flowing through it can update it in such a way that the neuron will never get further updated. However, we can work with this by setting a proper learning rate.

5.3 Long Short-Term Memory

An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.

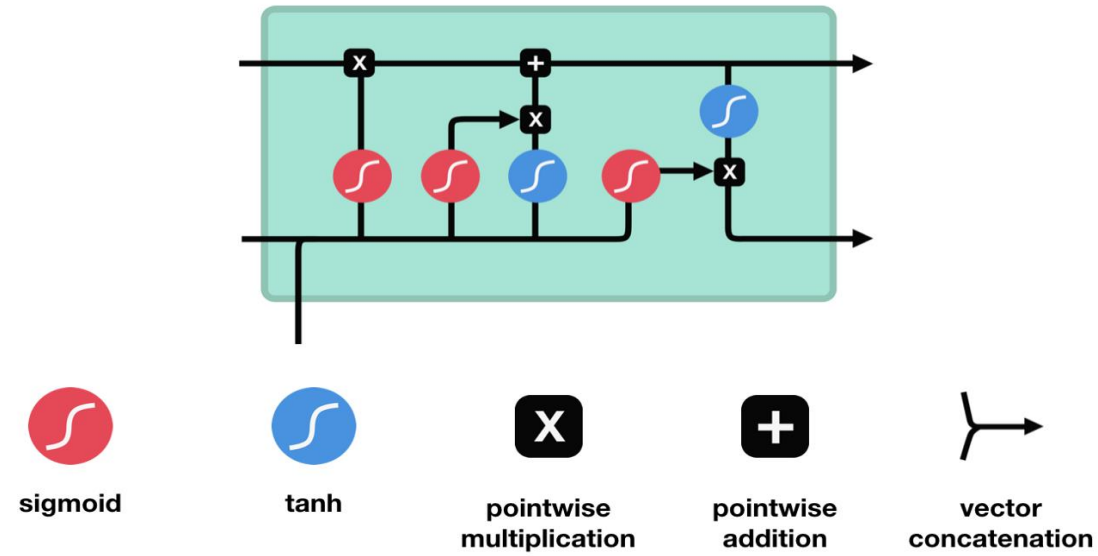


Figure-3. Long Short-Term memory

These operations are used to allow the LSTM to keep or forget information. Now looking at these operations can get a little overwhelming, so we'll go over this step by step.

The core concept of LSTMs is the cell state, and its various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the "memory" of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make its way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information gets added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

5.4 Gated Recurrent Units

The GRU is the newer generation of Recurrent Neural networks and is pretty much like an LSTM. GRUs got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate, and an update gate.

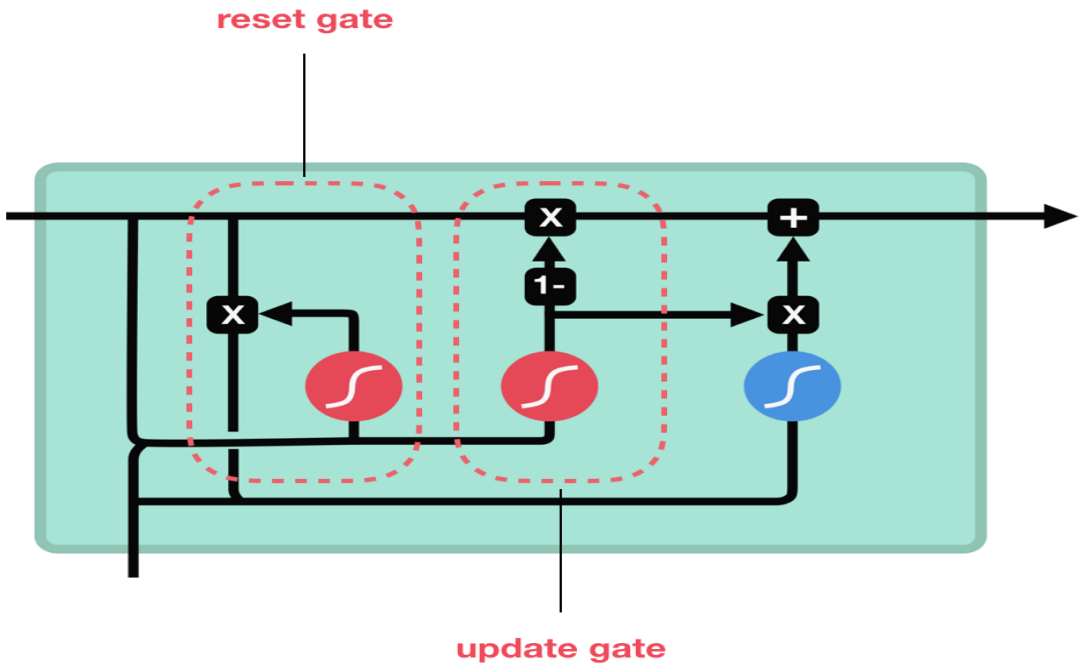


Figure-4. Gated Recurrent Units

The update gate acts like the forget and input gate of an LSTM. It decides what information to throw away and what new information to add. The reset gate is another gate that is used to decide how much past information to forget.

5.5 Bidirectional Long Short-Term Memory

Bidirectional long-short term memory(bi-lstm) is the process of making any neural network o have the sequence information in both directions backward (future to past) or forward (past to future).

In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backward or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.

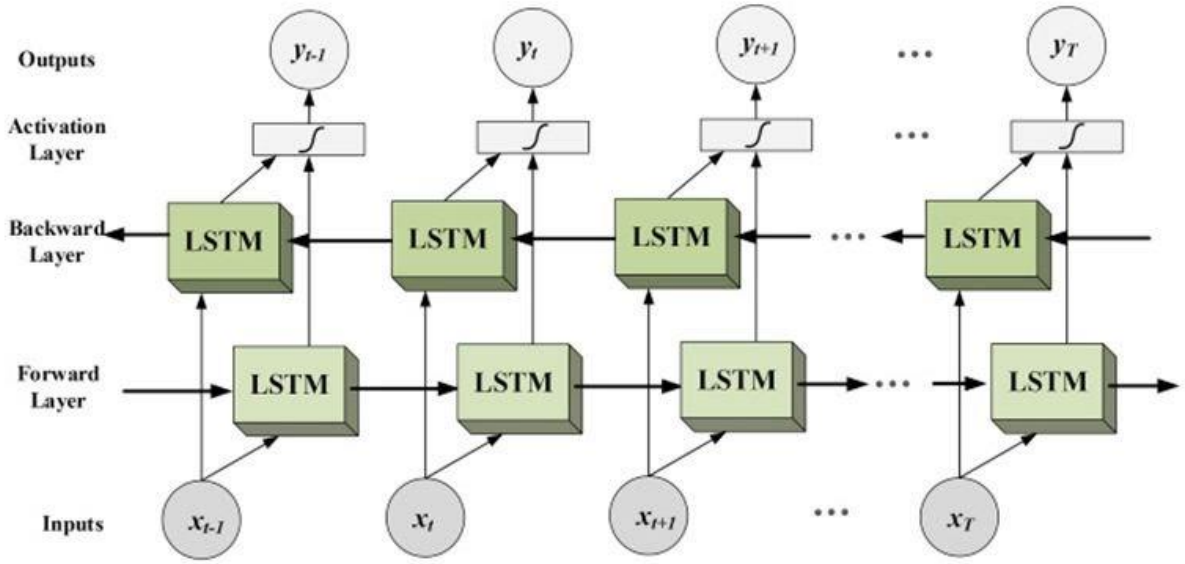


Figure-5. Bidirectional LSTM

In the diagram, we can see the flow of information from backward and forward layers. Bi-LSTM is usually employed where the sequence-to-sequence tasks are needed. This kind of network can be used in text classification, speech recognition, and forecasting models. Next in the article, we are going to make a bi-directional LSTM model using python.

5.6 Bidirectional Gated Recurrent Units

Models with bi-directional structures can learn information from previous and subsequent data when dealing with the current data. The structure of the bi-GRU model diagram is shown in Fig. 6. The bi-GRU model is determined based on the state of two GRUs, which are

unidirectional in opposite directions. One GRU moves forward, beginning from the start of the data sequence, and the other GRU moves backward, beginning from the end of the data sequence. This allows the information from both the future and past to impact the current states

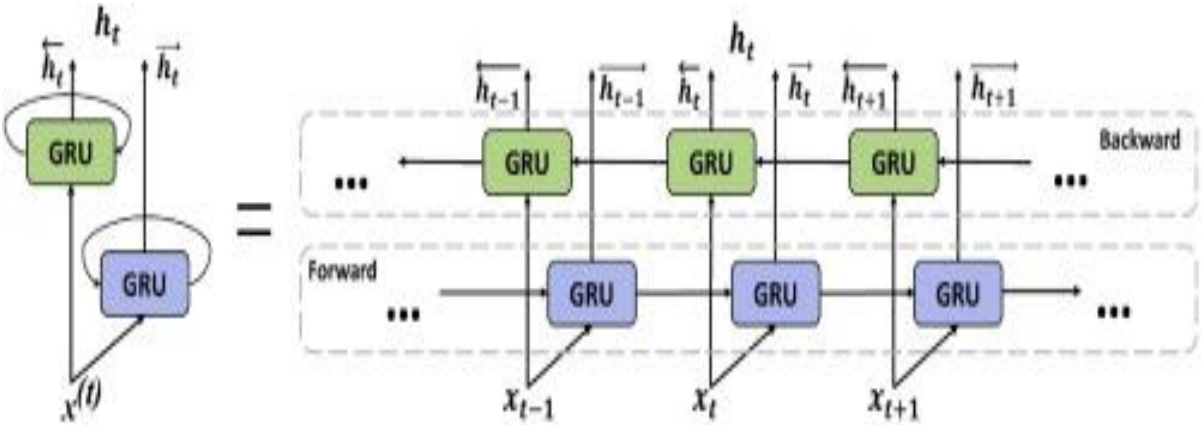


Figure-6. Bidirectional GRU

6. Model Implementation

6.1. MLP

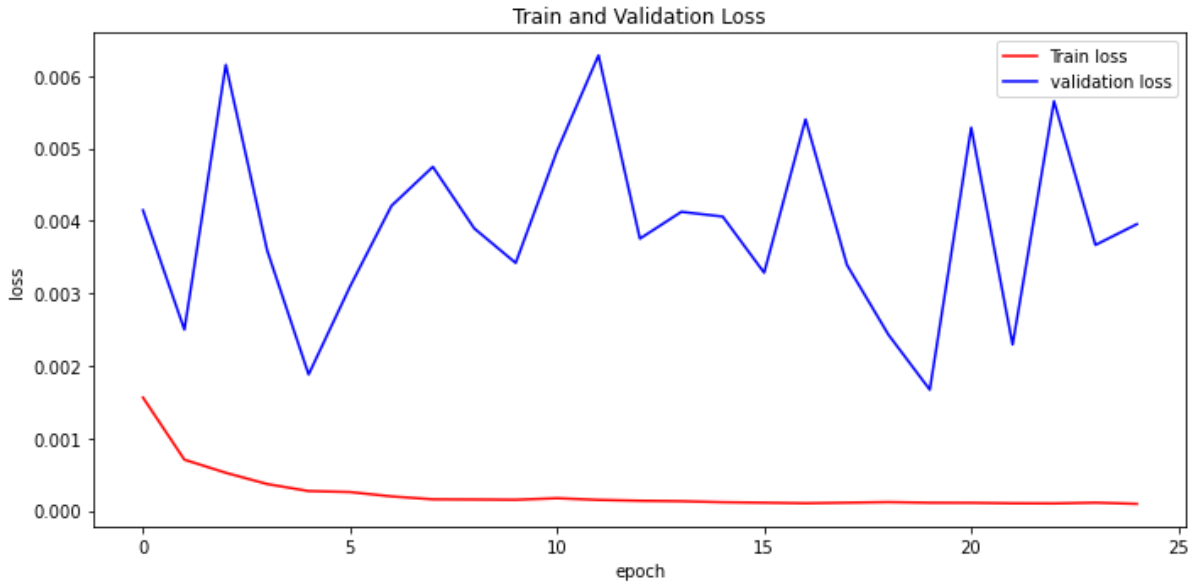
First, we will use a Multilayer Perceptron model, here the model will have input features. The thing with MLP models is that the model doesn't take the input as sequenced data, so for the model, it is just receiving inputs and don't treat them as sequenced data, that may be a problem since the model won't see the data with the sequence patter that it has. Here the model takes the input shape is [samples, timestamps].

I considered a total number of observations of 731 daily bitcoin close price values. Where I split the dataset into 70% into training and 30% testing, respectively.

The MLP model has 128 neurons or nodes with a dropout layer (0.2) that randomly drops 20 of the input before passing to the output layer. The details about the MLP layers shape can be shown in below figure.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               2048

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 2,177
Trainable params: 2,177
Non-trainable params: 0
_________________________________________________________________
```

Now is time to build the training model, we defined the few steps with X_train, Y_train as variables with 25 epochs. In this case, we define to reduce the mean squared error loss function and to optimize with the adam optimizer. The “val_loss” is the value of cost function for the cross-validation data and “loss” is the value of cost function for training data. The training and validation loss is shown in the figure below.

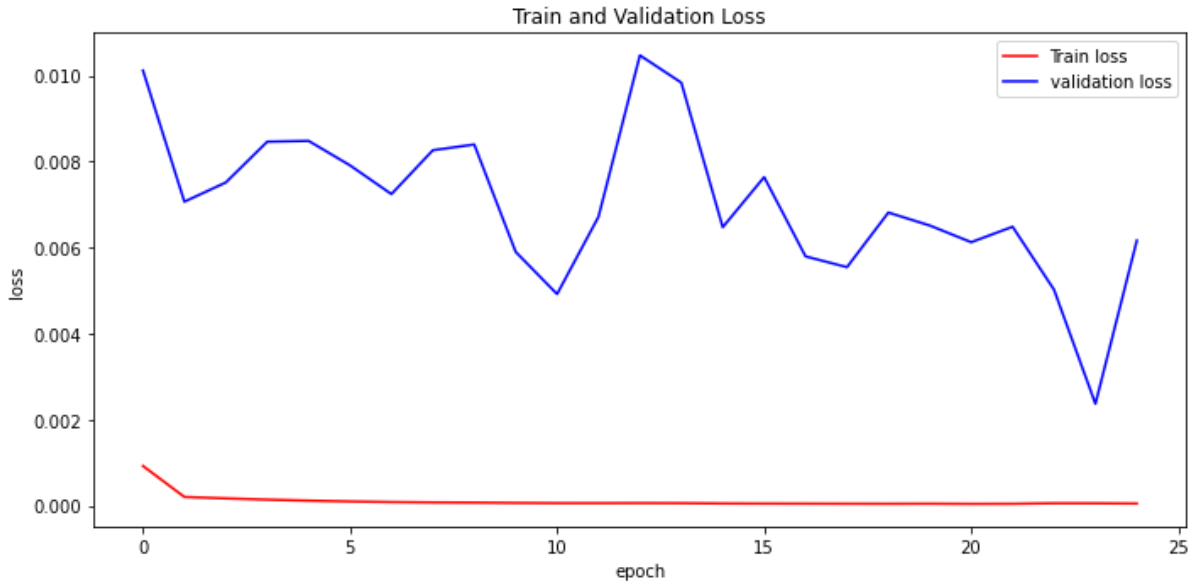


6.2. CNN

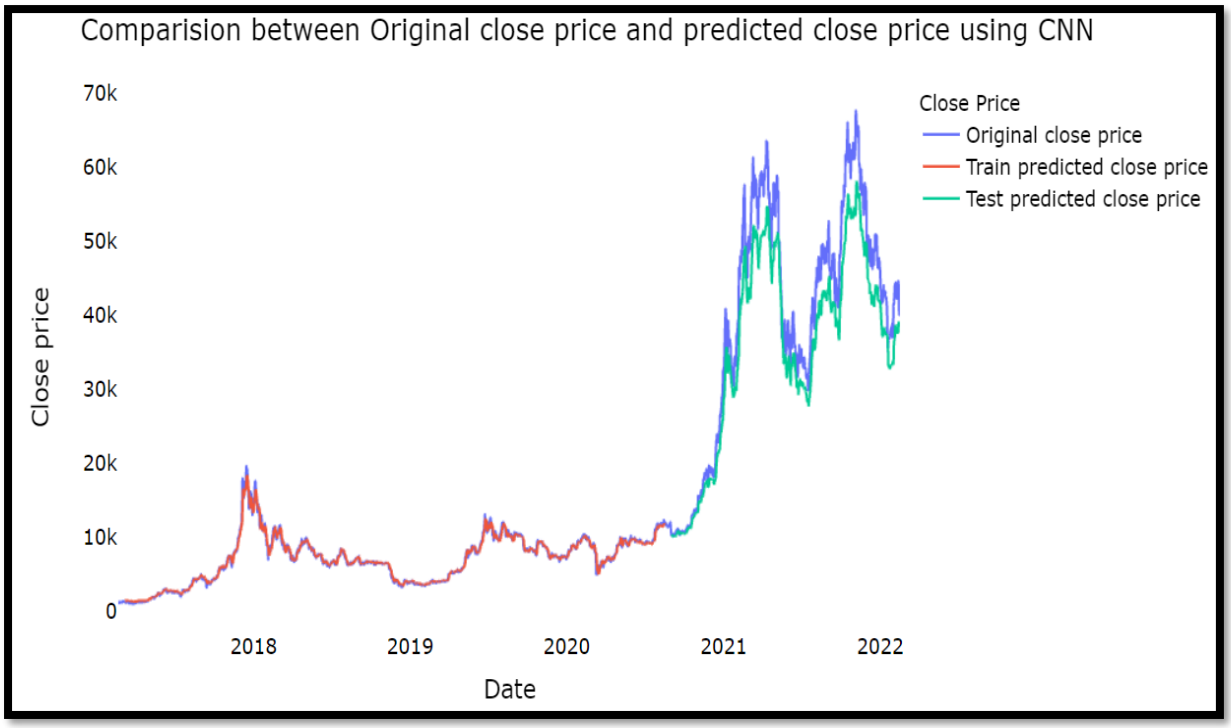
We are applying the Convolution Neural Network layer with 64 filters, convolution window is 3, relu activation function, then the layer with Max pooling1D, then flatten layer, finally dense layer with 50 neurons. The details about the CNN layer’s shape can be shown in the below figure.

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 module_wrapper_8 (ModuleWra  (None, 14, 64)           192
 pper)

 module_wrapper_9 (ModuleWra  (None, 7, 64)            0
 pper)

 flatten_4 (Flatten)          (None, 448)              0

 dropout_12 (Dropout)         (None, 448)              0

 dense_14 (Dense)             (None, 50)               22450

 dense_15 (Dense)             (None, 1)                51

=================================================================
Total params: 22,693
Trainable params: 22,693
Non-trainable params: 0
_________________________________________________________________
```

After defining the model, the training and validation loss is sometimes increasing and decreasing. The training and validation loss of the CNN model is shown in the below figure.



After setting the CNN model, we predict the bitcoin close price values by applying all the parameters. The below figure shows the comparison between the actual and predicted close price values after applying this model.



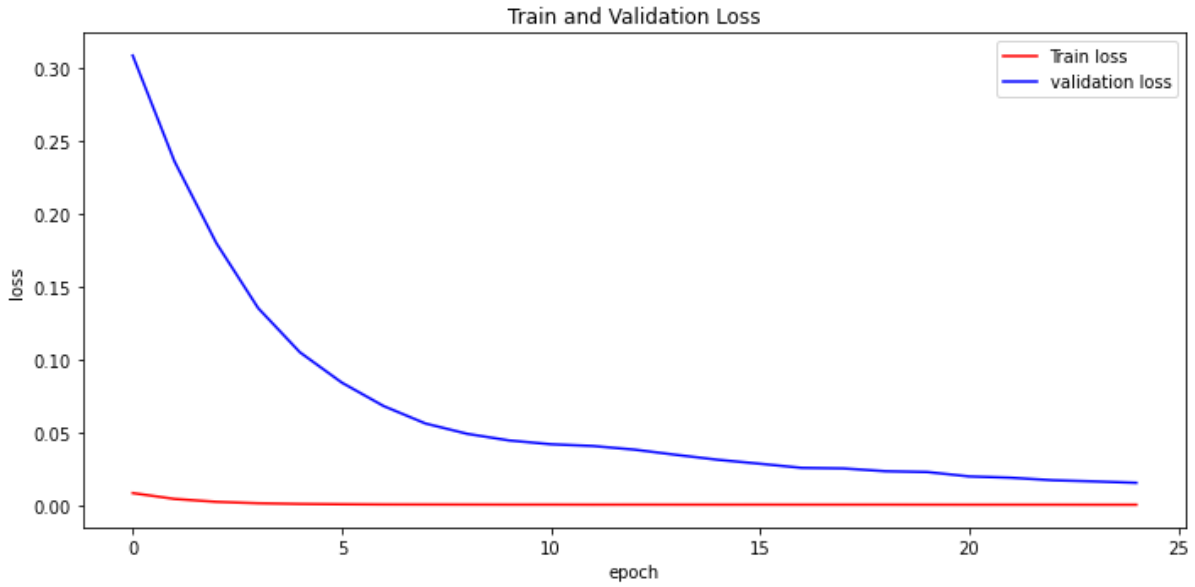
6.3. LSTM

As a rule of thumb, whenever we use a neural network, you should normalize or scale our data. We will use the Min-MaxScaler from Sklearn pre-processing library to scale data to a specific given range. In order to train LSTM, it is necessary to convert our data into the shape accepted by the LSTM.

The time series-forecasting model has an input layer, which feed into the LSTM layer and subsequently feed the output layer. The input layer only specifies the sequence length that ranges between 1 and -1. The LSTM layer has 64 neurons or nodes with a dropout layer (0.2) that randomly drops 20 of the input before passing to the output layer. The details about the LSTM layers shape can be checked in the below figure.

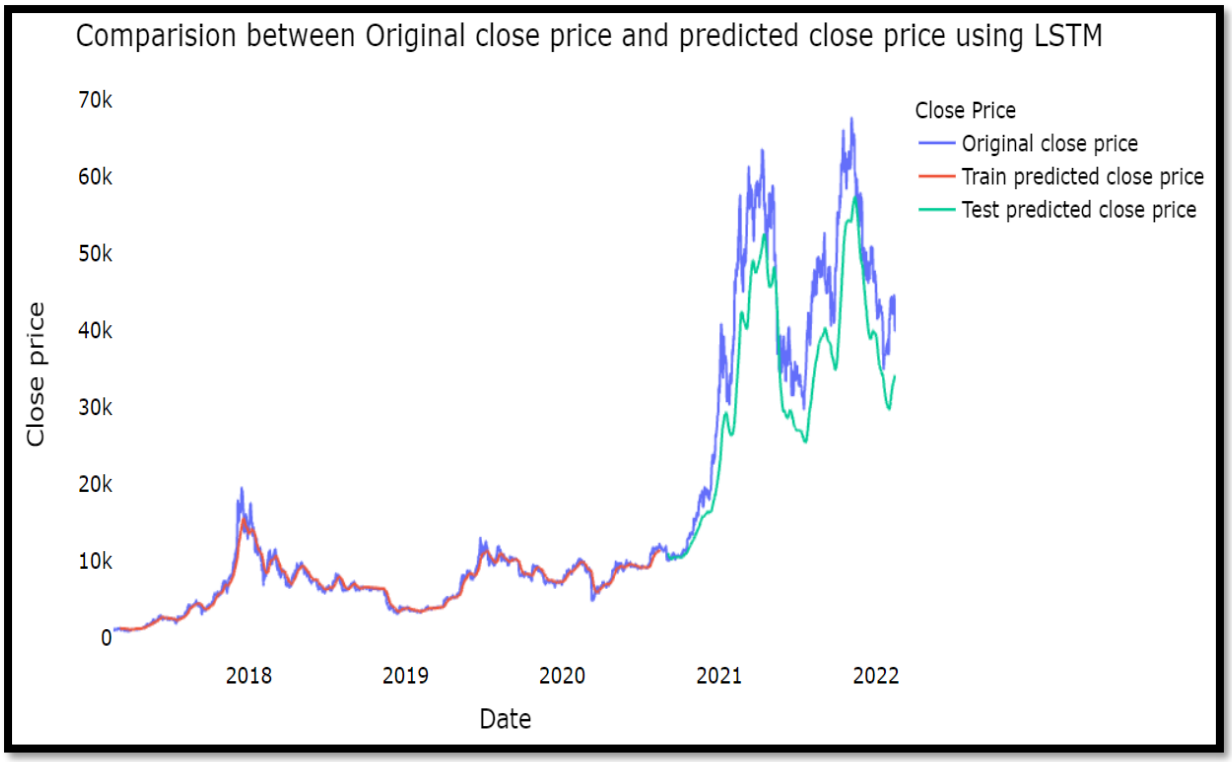
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dropout_1 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 1)                 65

=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

Now is time to build the training model, we defined the few steps with X_train, Y_train as variables with 25 epochs. The cross-validation set is used to select the best performing algorithm. In this case, I define to reduce the mean squared error loss function and to optimize with the adam optimizer. The “val_loss” is the value of cost function for the cross-validation data and “loss” is the value of cost function for training data. The training and validation loss is shown in the figure below.



The learning rate is a hyperparameter used in the training of neural networks which sets how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update. In this case, I defined an automatic condition of optimizing the learning rate at a specific positive number of epochs.

After setting LSTM model, we predict the bitcoin close prices values by applying all the parameters. In the below figure show the comparison between the actual and predicted close price values after applying this model.

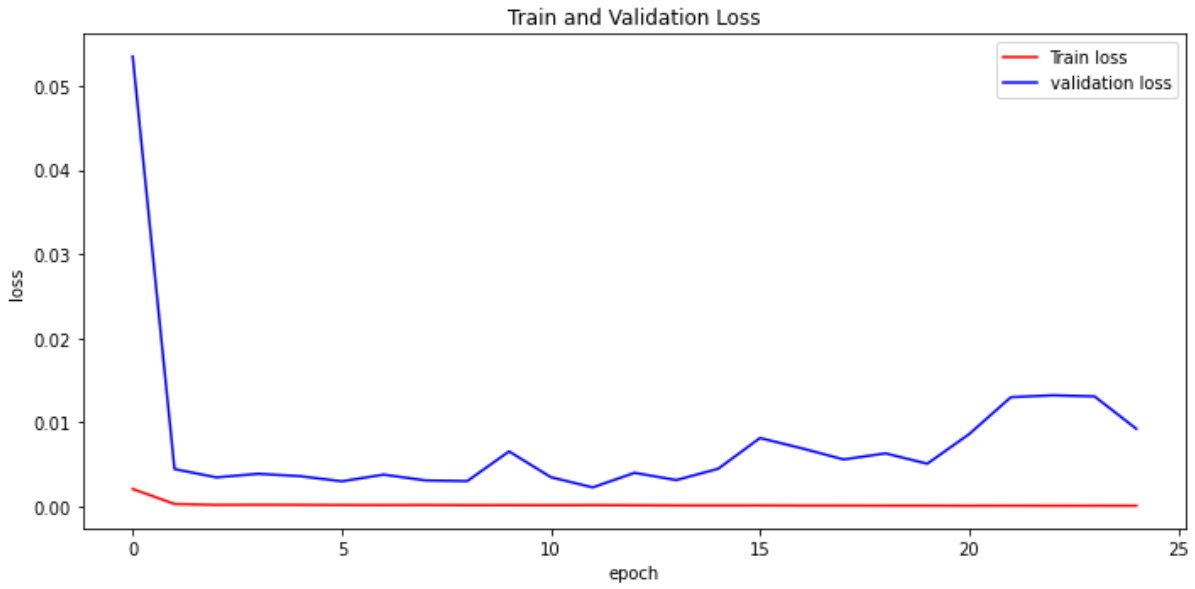


6.4. Bidirectional LSTM

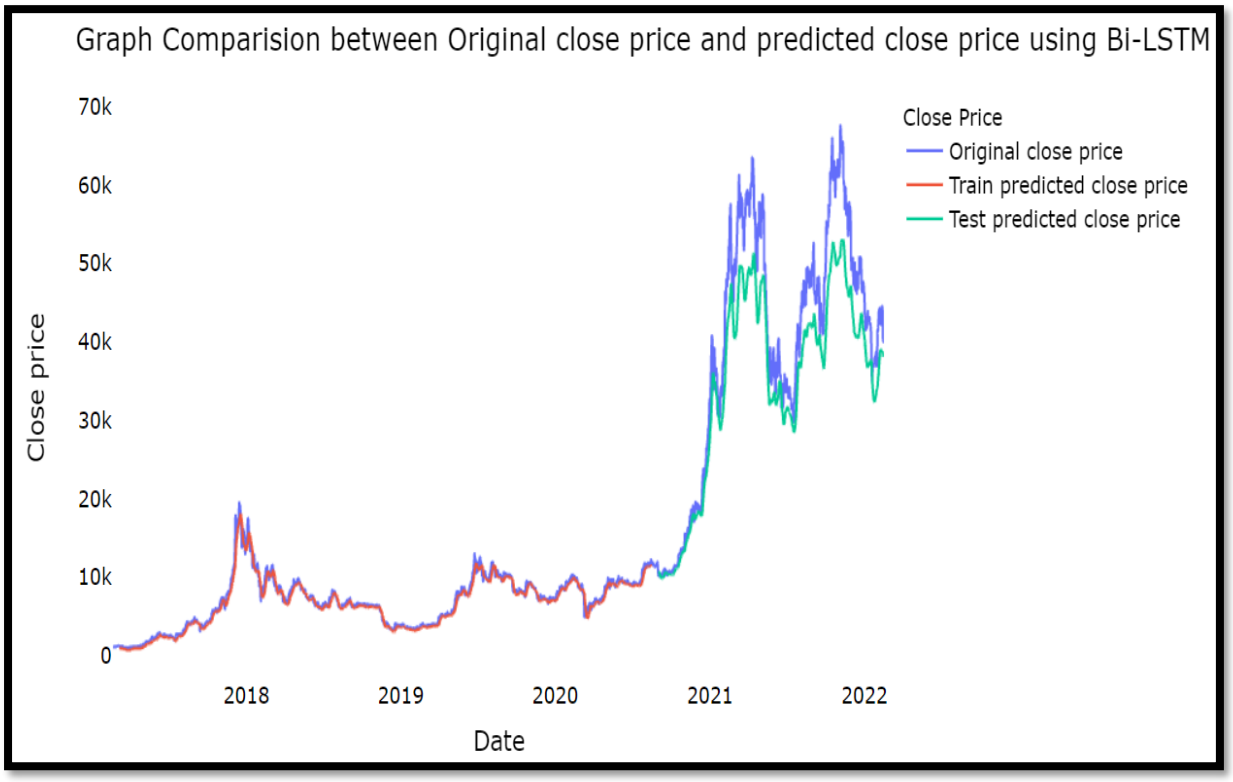
In this Bidirectional LSTM model, we are applying three bidirectional layers with 128 hidden units along with the tanh activation function. The details about the Bidirectional LSTM layer's shape can be checked in the below figure.

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_6 (Bidirectio  (None, 15, 256)          133120
 nal)

 dropout_16 (Dropout)        (None, 15, 256)           0

 bidirectional_7 (Bidirectio  (None, 15, 256)          394240
 nal)

 dropout_17 (Dropout)        (None, 15, 256)           0

 bidirectional_8 (Bidirectio  (None, 256)              394240
 nal)

 dropout_18 (Dropout)        (None, 256)               0

 dense_17 (Dense)            (None, 1)                 257

=================================================================
Total params: 921,857
Trainable params: 921,857
Non-trainable params: 0
_________________________________________________________________
```

After defining the Bi-LSTM model, the training loss is decreasing at one point then the loss is constant, but the validation loss is constant to all epochs. The training and validation loss of Bi-LSTM model is shown in the below figure.



After setting the Bi-LSTM model, we predict the bitcoin close prices values by applying all the parameters. The below figure show the comparison between the actual and predicted close price values after applying this model.

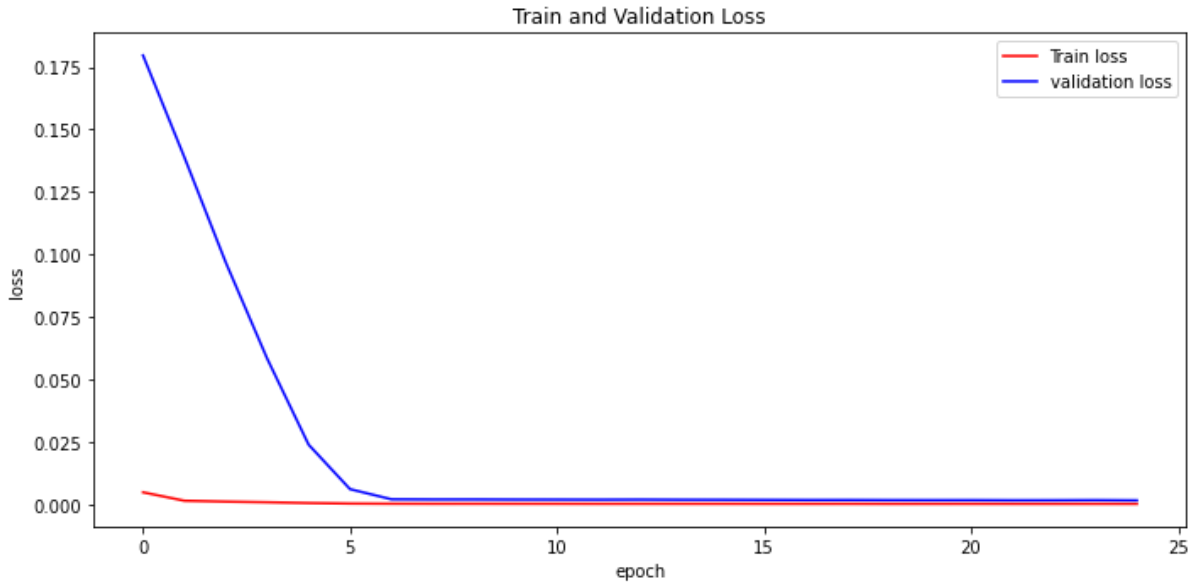


6.5. GRU

In this time series forecasting GRU model, we are applying the GRU layer with 128 hidden units along with the tanh activation function, then with a dropout layer (0.2) that randomly drops 20 of the input before passing to the output layer. The details about the GRU layers' shape can be checked in the below figure.

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_2 (GRU)                 (None, 128)               50304

 dropout_20 (Dropout)        (None, 128)               0

 dense_19 (Dense)            (None, 1)                 129

=================================================================
Total params: 50,433
Trainable params: 50,433
Non-trainable params: 0
_________________________________________________________________
```

After defining the GRU model, the training loss and validation loss is gradually decreasing up to one point, both the losses are maintaining the same range. The training and validation loss of the GRU model is shown in the below figure.



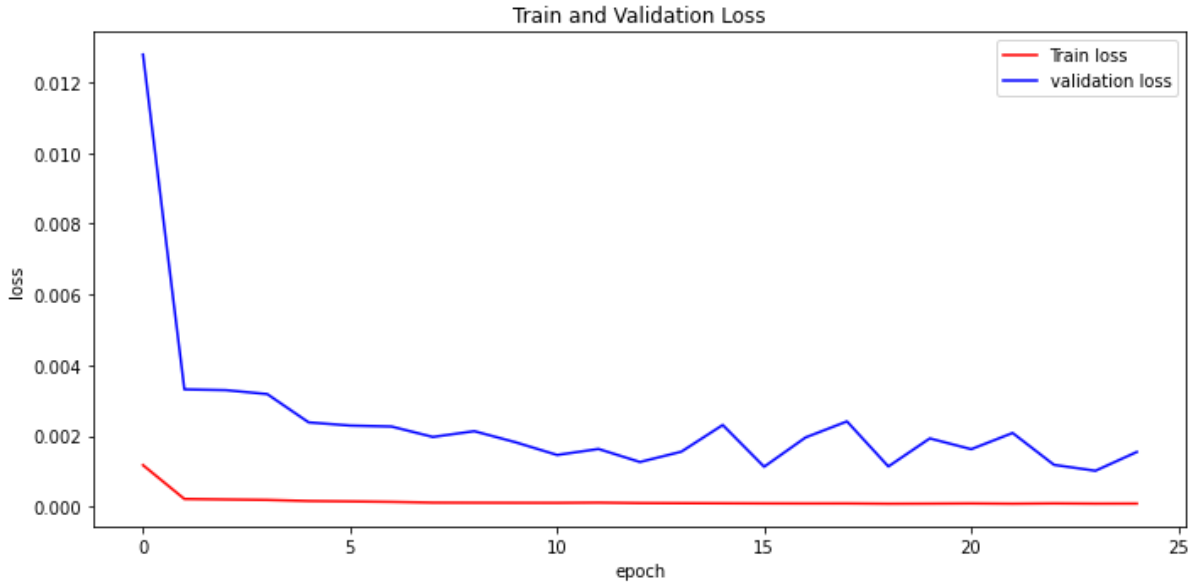
After setting the GRU model, we predict the bitcoin close prices values by applying all the parameters. The below figure shows the comparison between the actual and predicted close price values after applying this model.

6.6. Bidirectional GRU

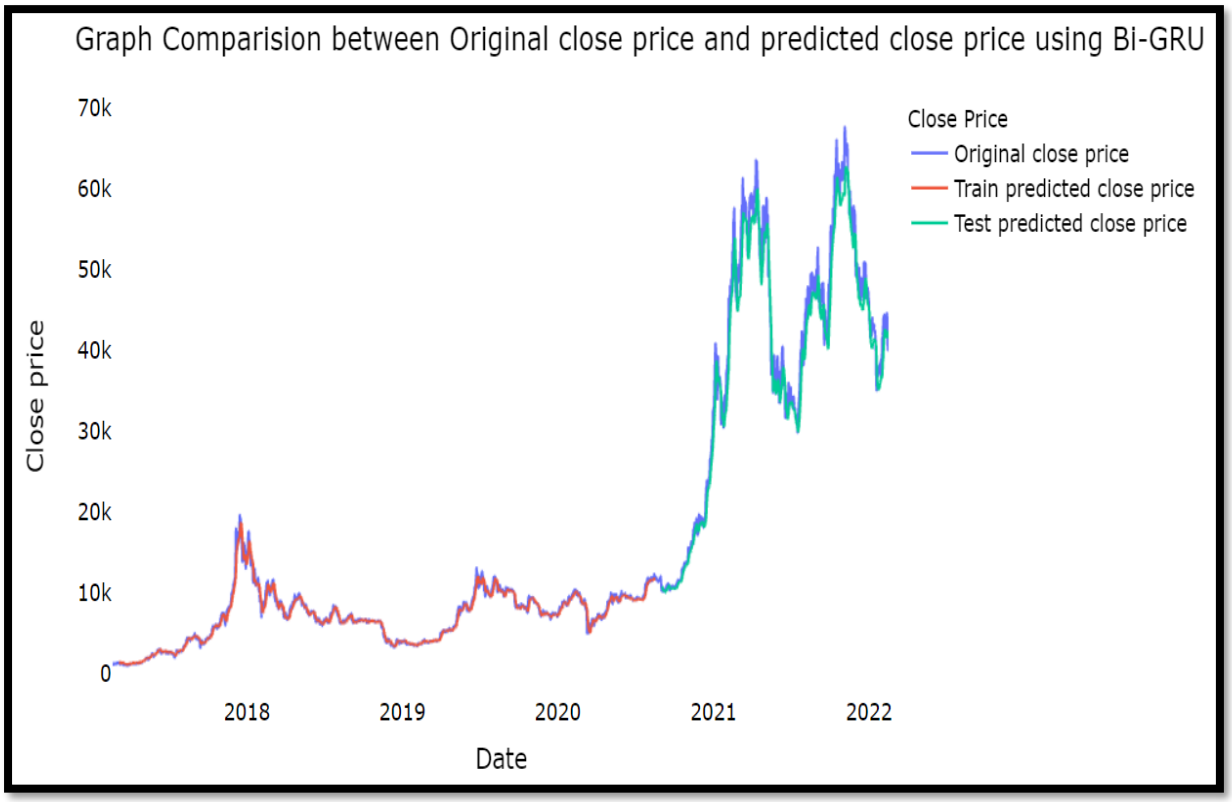
In this time series forecasting Bidirectional GRU model, we are applying three bidirectional GRU layers with 128 hidden units along with the activation function. The details about the Bidirectional GRU layers shape can be checked in the below figure.

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional_9 (Bidirectio  (None, 15, 256)          100608
 nal)

 dropout_21 (Dropout)        (None, 15, 256)           0

 bidirectional_10 (Bidirecti  (None, 15, 256)          296448
 onal)

 dropout_22 (Dropout)        (None, 15, 256)           0

 bidirectional_11 (Bidirecti  (None, 256)              296448
 onal)

 dropout_23 (Dropout)        (None, 256)               0

 dense_20 (Dense)            (None, 1)                 257

=================================================================
Total params: 693,761
Trainable params: 693,761
Non-trainable params: 0
_________________________________________________________________
```

After defining the Bi-GRU model, the training loss and validation loss are gradually decreasing up to one point, and both the losses are maintaining the same range. The training and validation loss of the GRU model is shown in the below figure.



After setting the Bi-GRU model, we predict the bitcoin close prices values by applying all the parameters. The below figure shows the comparison between the actual and predicted close price values after applying this model.



7. Results

The results demonstrate how all these models perform when forecasting bitcoin close price. Evaluating the model accuracy is an essential part of the process of creating machine learning models to describe how well the model is performing in its predictions. Each model predicts the bitcoin close price in terms of its forecasting error value for both RMSE, MSE, MAE, and R2 Score.

MAE (Mean absolute error) represents the difference between the original and predicted values extracted by averaging the absolute difference over the data set.

MSE (Mean Squared Error) represents the difference between the original and predicted values extracted by squaring the average difference over the data set.

RMSE (Root Mean Squared Error) is the error rate by the square root of MSE.

R-squared (Coefficient of determination) represents the coefficient of how well the values fit compared to the original values. The value from 0 to 1 are interpreted as percentages. The higher the value is, the better the model is.

The above metrics can be expressed,

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ – *predicted value of y*

$\bar{y}$ – *mean value of y*

	RMSE	MSE	MAE	R2 Score
CNN	5232.02	27374017.20	4387.37	0.88
LSTM	8255.16	68147705.60	6988.87	0.72
Bi-LSTM	6406.97	41049392.91	5199.89	0.83
GRU	2604.54	6783632.43	1955.65	0.97
Bi-GRU	2625.24	6891926.65	2017.39	0.96

Table-1. RMSE, MSE, MAE, R2 Score forecasting errors of all the models.

The lower value of MAE, MSE, and RMSE implies higher accuracy of a regression model. However, a higher value of R square is considered desirable.

The results show that GRU predictions have better precision in terms of forecasting errors: RMSE, MSE, MAE, and R2 Score, display the lower values when compared to the remaining models. These results confirm the literature insights since the GRU approach by large outperforms the CNN, LSTM, BI-LSTM, and Bi-GRU outputs as previously mentioned.

7. Conclusion

Machine learning techniques have recently gained a lot of popularity among the international community. The main purpose of this dissertation was to know whether these new approaches are more powerful than the traditional methods, or not. For this, I compared the accuracy of all the above-mentioned forecasts for a daily time series of bitcoin close prices.

The GRU predictions have better precision results in terms of forecasting errors are RMSE (2604.54), MSE (6783632.43), MAE (1955.65), and R2 Score (0.97), this model gives better forecasting error values when compared to the other models. We concluded that the GRU approach by large outperforms the other model outputs as previously mentioned.