CRYPTOCURRENCY PRICE PREDICTION

Ragul TN[1], Karthikeyan I[1], Vaidyanathan K[1],and Dr.GRHemalakshmi[2]
[1]Department of Computer Science and Engineering, National Engineering College, Kovilpatti,Tamil Nadu.
[2]Assistant Professor (Senior Grade), Department of Computer Science, National Engineering College, Kovilpatti, Tamil Nadu, India.

Abstract: Cryptocurrencies are a form of digital currency where all transactions are conducted electronically. Unlike fiat currency, which is centralized and requires third-party intervention, virtual currency users can access services without intermediaries. This means it exists only in digital form and does not have physical notes or coins. Our study aims to develop efficient prediction models using deep learning techniques, precisely long short-term memory (LSTM) and gated recurrent unit (GRU) to handle Bitcoin's price volatility and achieve high accuracy. We will compare the effectiveness of these two-time series deep learning models and demonstrate their ability to forecast Bitcoin prices.

Keywords: *LSTM-Long Short Term Memory, GRU-Gated Recurrent Unit*.

1. Introduction

Bitcoin has multiplied since 2017, and its price has frequently increased, attracting many investors. Its popularity has reached such a level that, in addition to many private companies, a country has recently announced its acceptance as a payment method.
Not to mention that central bank researchers have been analyzing and discussing it since at least 2014.

The most important factors that differentiate Bitcoin from other types are. Currency is characterized by its decentralization. That said, unlike other currencies, bitcoin transactions are not subject to any government processing and control. Its money supply grows over time, albeit non-linear, through a rewarding " mining " process. Computers participate in solving mathematical equations by brute force and are rewarded with bitcoins. The classic law of supply and demand determines the exchange rate of Bitcoin's price against other currencies.

Similar to any other financial process, bitcoin prices can be predicted by artificial neural network methods. While implementing artificial neural network methods to predict other financial processes such as (e.g., stock prices) is long, and due to their novel nature, there is not much literature on the cryptocurrency price prediction. Nevertheless, in recent years, many researchers have attempted to create hybrid artificial neural network models to predict prices and price fluctuations in cryptocurrency prices, primarily focusing on Bitcoin.

A severe limitation of RNNs is the inability to capture long-term dependencies in sequences. One way to handle this situation is to use a long-short-term memory (LSTM) variant of RNN.
The cryptocurrency market differs from the traditional stock market because it has several new features.
It is necessary to apply new technologies suitable for the cryptocurrency market for forecasting. There is less research on cryptocurrency price predictions than on the stock market.

This algorithm works great for time series data such as time series, voice, text, financial data, audio, video, weather, etc. RNNs can understand sequences and their context better than other algorithms. In RNNs, information follows a cycle.
When making a decision, it considers the current input and what it has learned from previously received inputs.

2. Related Work

[1] Sina E. Charandabi, Kamyar Kamyar(2021) -The purpose of the study is to examine whether the application of deep learning-based dual-stage Partial Least Square-Structural Equation Modeling (PLS-SEM) & Artificial Neural Network (ANN) analysis enables better in-depth research results as compared to single-step PLS-SEM approach and to excavate factors which can predict behavioural intention to adopt cryptocurrency.
[2]Ahmed M. Khedr, Ifra Arif, Pravija Raj P V(2021) – Traditional statistical methods require a lot of statistical assumptions that could be unrealistic, leaving machine learning as the best technology in this field. This article comprehensively summarises the previous studies in cryptocurrency price prediction from 2010 to 2020. The discussion presented in this article will help researchers fill the gap in existing studies and gain more insight.
[3]Edwin, Lipo Wang (2017) - Described the features of Bitcoin and the following day's change in the price of Bitcoin using an Artificial Neural Network ensemble. The features of Bitcoin and the following day change in the price of Bitcoin using an Artificial Neural Network ensemble approach called Genetic Algorithm-based Selective Neural Network Ensemble, constructed using Multi-Layered Perceptron as the base model for each of the neural network in the ensemble.

[4] Yeray Mezquita, Ana Belén Gil-González, Javier Prieto, Juan Manuel Corchado (2021)-This paper proposes a platform based on blockchain technology and the multi-agent system paradigm to allow for the creation of an automated peer-to-peer electricity market in micro-grids. Using a permissioned blockchain network has multiple benefits, reducing transaction costs and enabling micro-transactions.
[5] Lloyd Kasal, Mihir Shetty, Tanmay Nayak, Ramanath Pai, and Shilpa B(2022) proposed that numerous neural networks may be utilized to analyze cryptocurrency values. The most successful of them all has been determined to be LSTM. The key factors used are available price, close price, high price, low price, volume, and market cap with the interdependencies amid some cryptocurrencies, thus centres on evaluating vital features that influence the trade's unpredictability by applying the model to increase the effectiveness of this process.
[6]Ahmed M. Khedr Ifra Arif Pravija Raj P V Magdi El-Bannany (2021)- proposed that traditional statistical methods, although simple to implement and interpret, require a lot of statistical assumptions. This article comprehensively summarises the previous studies in cryptocurrency price prediction from 2010 to 2020. The discussion presented in this article will help researchers fill the gap in existing studies and gain more insight.
[7]Caporale, Guglielmo Maria Plastun, Alex (2018)-A trading robot approach is then used to establish whether these statistical anomalies can be exploited to generate profits. The results suggest that a strategy based on counter-movements after overreactions is not profitable, whilst one based on inertia appears profitable but produces outcomes not statistically different from the random ones.
[8]Al-Yahyaee KH, Mensi W(2020) -Multifractality, long-memory process, and efficiency hypothesis of six significant cryptocurrencies using the time-rolling MF-DFA approach. The causality between cryptocurrencies and economic factors is undirected. Interestingly, our findings show that cryptocurrencies are insignificant correlations with economic factors. The result implies that cryptocurrencies can not be assumed as financial assets to hedge systematic risks from economic factors.
[9]Magdi El-Bannany, Saadat M. Alhashmi, Meenu Sreedharan(2016) - Cryptocurrency price prediction using traditional statistical and machine-learning techniques, this type of model approach called Genetic Algorithm based Selective Neural Network Ensemble.
[10]Ujan Mukhopadhyay; Anthony Skjellum; Oluwakemi; Hambolu; Jon Oakley; Richard Brooks(2022) proposed that Cryptocurrencies require robust and secure mining algorithms. In this paper, we survey and compare and contrast current mining techniques used by major Cryptocurrencies. Mining adds records of past transactions to the distributed ledger known as Blockchain, allowing users to reach secure, robust consensus for each transaction. Mining also introduces wealth in the form of new units of currency. Cryptocurrencies lack a central authority to mediate transactions because they were designed as peer-to-peer systems
[11] Carmine Ventre, Michail Basios, Leslie Kanthan, David Martinez-Rego(2022), This survey analyzes the research distribution that characterizes cryptocurrency Research and distribution among properties. This also analyzes datasets, research trends and distribution among research objects (contents/properties) and technologies, concluding with promising cryptocurrency trading opportunities.
[12] Sina E. Charandabi; Kamyar Kamyar (2021), This survey paper aims to present and compare multiple research papers that employed multiple neural networks. Researchers would be allowed to decide the more suitable direction of work to provide more accurate alternatives to the field.
[13] Saeed Alzahrani; Tugrul U. Daim(2019) -Paper suggests that the main factors driving the adoption decision are the acceptance by businesses as a payment method and the fast transfer of funds. This aims at providing an in-depth analysis of the factors influencing the adoption of cryptocurrency as well as the ranking of these influencing factors based on the quantification of the users' judgments.
[14]Pravija Raj, Magdi El-Bannany, Saadat M.Alhashmi, Meenu Sreedharan(2021) –Research in this field uses traditional statistical and machine-learning techniques, making it hard to predict using a statistical approach, such as Bayesian regression, logistic regression, linear regression, support vector machine, artificial neural network, deep learning, and reinforcement learning. No seasonal effects exist in cryptocurrency, making it hard to predict using a statistical approach.
[15] George S.Atsalakis(2021) –Fuzzy modelling demonstrating that the closed-loop or feedback control technique can cope with uncertainties associated with the dynamic behaviour of the price of Bitcoin and achieve positive returns. This study proposes a computational intelligence technique that uses a hybrid Neuro-Fuzzy controller, namely PATSOS, to forecast the direction of the change of Bitcoin's daily price.

3. Methodology

LSTM (Long Short-Term Memory) is another module provided for RNN. LSTM was developed and promoted by Hochreiter & Schmidhuber (1997) [3]and later by many researchers. Like RNNs, LSTM networks (LSTM networks) are also composed of cyclically coherent modules.

LSTM is an updated version of RNN; the difference is in the relationship between the hidden layers of RNN. The interpretation structure of RNN is shown in Figure 1. The LSTM recurrent neural network also has a similar structure; another difference is the hidden layer memory unit structure. And the unique three-door design effectively solves the gradient problem. LSTM memory structure for hidden layers.

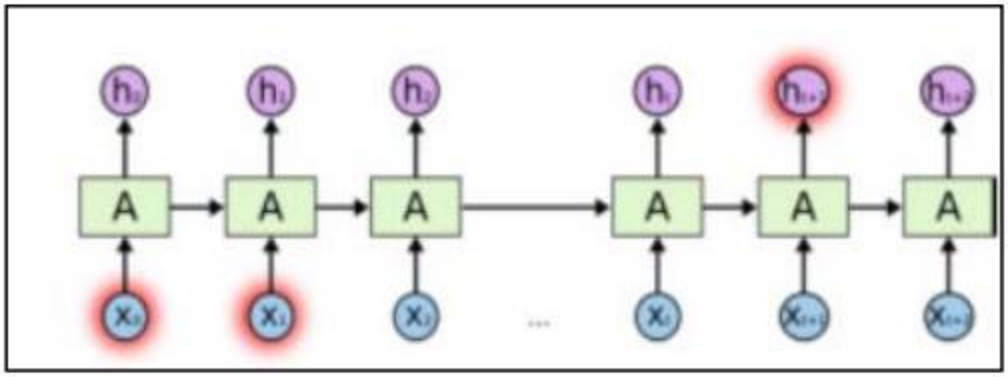


Fig 3.1: LSTM architecture

Some things in the figure show that RNN has flaws. Faults can be seen in the input X0, X1 has a lot of miscellaneous information Xt, Xt+1 so that when 1+1 information needs those related X0, X1 to RNN cannot learn Because the association of information stored in old memory becomes increasingly useless over time as it is overwritten or replaced by new memory, Bengio et al. (1994) [22]. Unlike RNN, LSTM has no drawback in that LSTM can manage memory for each input using memory cells and gate cells. The prediction is the easy part. It involves taking prepared input data (X) and calling one of the Keras prediction methods on the loaded model. The input to make predictions (X) contains only the input sequence data needed to make predictions, not all previous training data. In the case of predicting the next value in a sequence, the input sequence will be 1 sample with a fixed number of time steps and features used. The predictive model calculates future predictions using a new autoregressive scheme called the autoregressive moving pointer model. Initially, AMPM is used to generate input-output pairs, given these input-output pairs and generate future predictions. The next step is to form an optimal portfolio based on these predictions, assuming a normal distribution of stock prices.

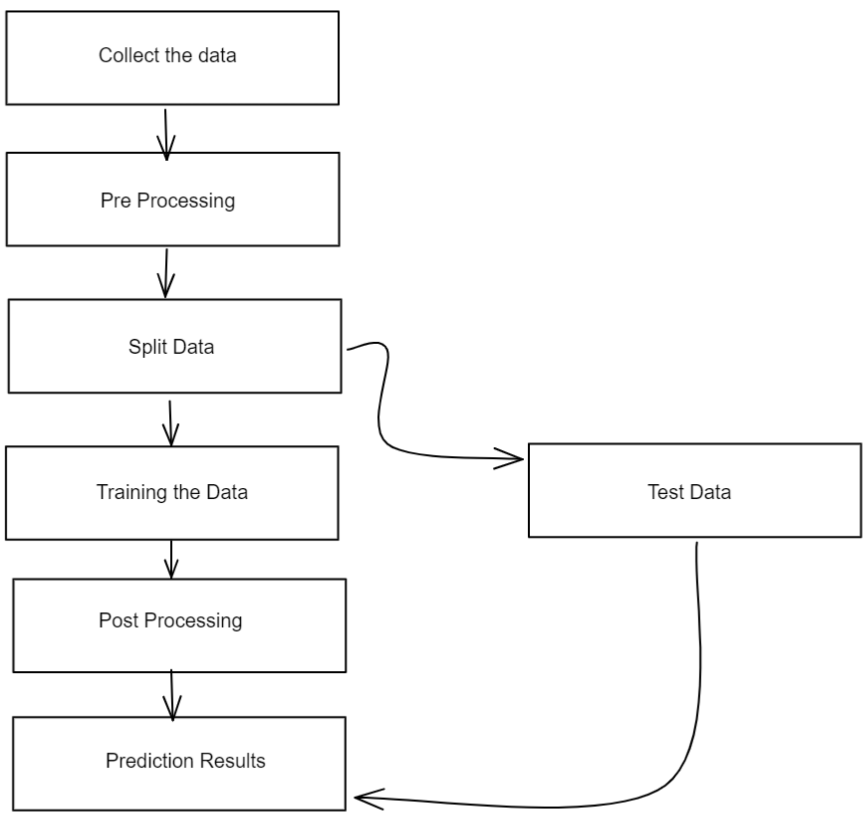


Fig 3.2: Workflow diagram

Statistical methods include the logistic regression model, ARCH model, etc. Artificial intelligence methods include multi-layer perceptrons, convolutional neural networks, naive bayesian networks, backpropagation networks, single-layer LSTMs, support vector machines, recurrent neural networks, etc. Long-short-term memory (LSTM) network.

3.1.Long Short Term Memory (LSTM)

LSTM is a unique network structure with three "gate" structures. Three gates are placed in an LSTM cell: entry gate, forget gate, and exit gate. Information entering the LSTM network can be selected according to rules. Only information that conforms to the algorithm will be retained, and the forgetting gate will forget information that does not conform.

LSTMs use a series of "gates" to control how information from a data series enters, stores, and leaves the network. There are three doors in a typical LSTM; the door of oblivion, the door of entry and the door of exit. These gates can be considered filters, and each is its neural network. This network (within the forgetting gate) is trained to have an output close to 0 when a component of the input is considered irrelevant and close to 1 when it is relevant. It is helpful to think of each element of this vector as a filter/sifter, allowing more information to pass as the values approach 1. These output values are then sent and multiplied point by point by the previous state of the cell.

In summary, the forget gate decides which pieces of long-term memory should now be forgotten (with less weight) given the previous hidden state and the new data points in the queue.

3.2.Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is an artificial neural network that uses temporal or chronological data. These deep learning algorithms are often used for sequential or temporal problems such as language translation, natural language processing (NLP), speech recognition, and image captioning; they're built into popular apps like Siri, voice search, and Google Translate. Like feedforward and convolutional neural networks (CNNs), recurrent neural networks use training data to learn. They are known for their "memory" as they retrieve information from past entries to influence current entries and exits. Traditional deep neural networks assume that the input and output are independent of each other, while recurrent neural networks' output depends on the sequence's previous elements. Although future events can also be used to determine the output of a given sequence, one-way recurrent neural networks cannot factor these events into their predictions.

3.3.Convolutional Neural Networks (CNNs)

CNN is an artificial neural network widely used in image/object recognition and classification. Therefore, deep learning uses CNNs to recognize objects in images. CNNs play an essential role in various tasks/functions such as image processing problems, computer vision tasks such as localization and segmentation, video analysis, identification of obstacles in self-driving cars and speech recognition in natural language processing. Since CNNs play an essential role in these emerging and rapidly developing fields, they are trendy in deep learning.

CNNs are another type of neural network that can reveal critical information in time series and image data. Therefore, it benefits image-related tasks, such as image recognition, object classification, and pattern recognition. To recognize patterns in images, CNNs use the principles of linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data. The architecture of CNN is similar to the connection model of the human brain. Just as the brain is made up of billions of neurons, a CNN has neurons arranged in a specific way. The neural arrangement of the CNN resembles the frontal lobe of the brain, the area responsible for processing visual stimuli. This arrangement ensures that the entire field of view is covered, avoiding traditional neural networks' segmented image processing problems, which must segment input images at lower resolutions. Compared to older networks, CNNs perform better on image and voice or audio signal inputs. A deep-learning CNN consists of the convolutional, pooling, and fully connected (FC) layers. The convolutional layer is the first layer and the FC layer is the last layer. The complexity of CNN increases from convolutional layers to fully connected layers. This increasing complexity allows a CNN to continuously recognize more significant parts of an image and more complex features until it finally recognizes the entire object. Analyze and represent human language.

3.4.Natural Language Processing (NLP)

NLP-based systems have enabled a wide range of applications, such as Google's powerful search engine and, more recently, Amazon's voice assistant, Alexa. NLP also helps teach machines the ability to perform complex natural language tasks, such as machine translation and dialogue generation. NLP can recognize and predict disease based on electronic health records and patients' speech. This ability is being studied in health conditions ranging from cardiovascular disease to depression and even schizophrenia. For example, Amazon Comprehend Medical is a service that uses NLP to extract disease states, medications, and treatment outcomes from patient medical records, clinical trial reports, and other electronic health records. Organizations can determine what customers are saying about a service or product by identifying and extracting information from sources such as social media. This sentiment analysis can provide insight into customer choices and the drivers behind their decisions. NLP is also used in the search and selection phase of talent acquisition, identifying the skills of potential recruits and scouting them before they become active in the job market.

4. Result and Discussion

LSTM is a complex area of deep learning. LSTMs are often referred to as sophisticated RNNs. Vanilla RNN has no cellular state. They only have hidden states, which serve as the memory of the RNN. At the same time, LSTM has both a cell state and a hidden state, and our proposed model successfully provided Yahoo Finance stock market results to predict Bitcoin. A long-short-term memory (LSTM) network is a recurrent neural network capable of learning sequence dependencies in prediction problems. This is the desired behavior in complex problem areas like machine translation, speech recognition, etc. Our models using time series techniques can provide results that predict prices for the next few days by splitting the data to train and test what we mentioned in the article above.

2. Importing Library

```
# First we will import the necessary Library

import os
import pandas as pd
import numpy as np
import math
import datetime as dt

# For Evalution we will use these library

from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score, r2_score
from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accuracy_score
from sklearn.preprocessing import MinMaxScaler

# For model building we will use these library

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import LSTM


# For PLotting we will use these library

import matplotlib.pyplot as plt
from itertools import cycle
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
```

Fig 4.1: Importing Library

3. Loading Dataset

We can use this link to download bitcoin dataset from yahoo finance

```
# Load our dataset
# Note it should be in same dir


maindf=pd.read_csv('BTC-USD.csv')
```

```
print('Total number of days present in the dataset: ',maindf.shape[0])
print('Total number of fields present in the dataset: ',maindf.shape[1])
```

```
Total number of days present in the dataset:  2713
Total number of fields present in the dataset:  7
```

```
maindf.shape
```

```
(2713, 7)
```

```
maindf.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800
1	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200
2	2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	37919700
3	2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	36863600
4	2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	26580100

Fig 4.2: Loading Dataset

```
maindf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2713 entries, 0 to 2712
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2713 non-null   object
 1   Open       2713 non-null   float64
 2   High       2713 non-null   float64
 3   Low        2713 non-null   float64
 4   Close      2713 non-null   float64
 5   Adj Close  2713 non-null   float64
 6   Volume     2713 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 148.5+ KB
```

```
maindf.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2713.000000	2713.000000	2713.000000	2713.000000	2713.000000	2.713000e+03
mean	11311.041069	11614.292482	10975.555057	11323.914637	11323.914637	1.470462e+10
std	16106.428891	16537.390649	15608.572560	16110.365010	16110.365010	2.001627e+10
min	176.897003	211.731003	171.509995	178.102997	178.102997	5.914570e+06
25%	606.396973	609.260986	604.109985	606.718994	606.718994	7.991080e+07
50%	6301.569824	6434.617676	6214.220215	6317.609863	6317.609863	5.098183e+09
75%	10452.399414	10762.644531	10202.387695	10462.259766	10462.259766	2.456992e+10
max	67549.734375	68789.625000	66382.062500	67566.828125	67566.828125	3.509679e+11

Fig 4.3: Fixing Range for data

Checking for Null Values

```
print('Null Values:',maindf.isnull().values.sum())
```

```
Null Values: 0
```

```
print('NA values:',maindf.isnull().values.any())
```

```
NA values: False
```

```
# If dataset had null values we can use this code to drop all the null values present in the dataset

# maindf=maindf.dropna()
# print('Null Values:',maindf.isnull().values.sum())
# print('NA values:',maindf.isnull().values.any())
```

```
# Final shape of the dataset after dealing with null values

maindf.shape
```

```
(2713, 7)
```

Fig 4.4: Checking for Null values

4. EDA(Exploratory Data Analysis)

```
# Printing the start date and End date of the dataset

sd=maindf.iloc[0][0]
ed=maindf.iloc[-1][0]


print('Starting Date',sd)
print('Ending Date',ed)
```

```
Starting Date 2014-09-17
Ending Date 2022-02-19
```

StockPrice Analysis from Start

Fig 4.5: Exploratory Data Analysis

Analysis of year 2014

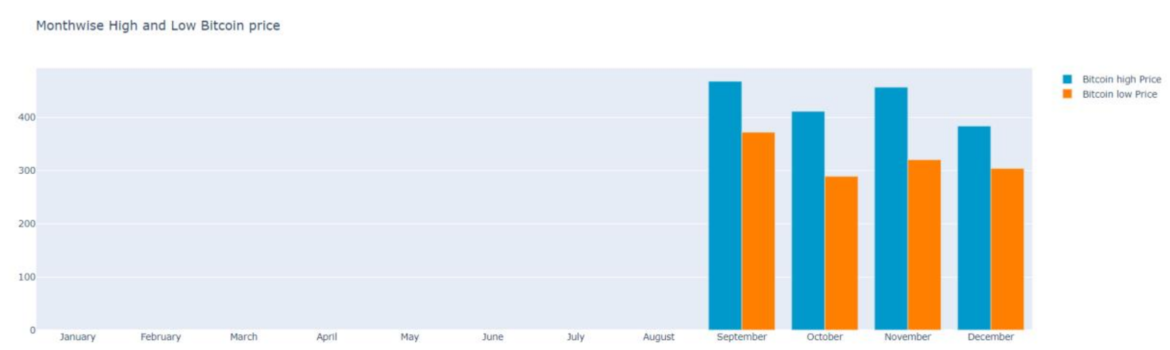
```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2014 = maindf.loc[(maindf['Date'] >= '2014-09-17')
                     & (maindf['Date'] < '2014-12-31')]

y_2014.drop(y_2014[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
0	2014-09-17	465.864014	468.174011	452.421997	457.334015
1	2014-09-18	456.859985	456.859985	413.104004	424.440002
2	2014-09-19	424.102997	427.834991	384.532013	394.795990
3	2014-09-20	394.673004	423.295990	389.882996	408.903992
4	2014-09-21	408.084991	412.425995	393.181000	398.821014
...	...	...	...	...	...
100	2014-12-26	319.152008	331.424011	316.627014	327.924011
101	2014-12-27	327.583008	328.911011	312.630005	315.863007
102	2014-12-28	316.160004	320.028015	311.078003	317.239014
103	2014-12-29	317.700989	320.266998	312.307007	312.670013
104	2014-12-30	312.718994	314.808990	309.372986	310.737000

105 rows × 5 columns

```
monthvise= y_2014.groupby(y_2014['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Fig 4.6: Analysis of the year 2014



Note that we only have few months in 2014 so the rest of the months are not plotted since we do not have the data

Fig 4.7: The year 2014 Monthwise bitcoin price

```python
names = cycle(['Bitcoin Open Price','Bitcoin Close Price','Bitcoin High Price','Bitcoin Low Price'])

fig = px.line(y_2014, x=y_2014.Date, y=[y_2014['Open'], y_2014['Close'],
                                          y_2014['High'], y_2014['Low']],
             labels={'Date': 'Date','value':'Bitcoin value'})
fig.update_layout(title_text='Bitcoin analysis chart', font_size=15, font_color='black',legend_title_text='Bitcoin Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.8: Bitcoin analysis chart 2014

Analysis of Year 2015

5 cells hidden

Analysis of Year 2016

```python
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2016 = maindf.loc[(maindf['Date'] >= '2016-01-01')
                     & (maindf['Date'] < '2017-01-01')]

y_2016.drop(y_2016[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
471	2016-01-01	430.721008	436.246002	427.515015	434.334015
472	2016-01-02	434.622009	436.062012	431.869995	433.437988
473	2016-01-03	433.578003	433.743011	424.705994	430.010986
474	2016-01-04	430.061005	434.516998	429.084015	433.091003
475	2016-01-05	433.069000	434.182007	429.675995	431.959991
...	...	...	...	...	...
832	2016-12-27	908.354004	940.047974	904.255005	933.197998
833	2016-12-28	934.830994	975.921021	934.830994	975.921021
834	2016-12-29	975.125000	979.396973	954.502991	973.497009
835	2016-12-30	972.534973	972.534973	934.833008	961.237976
836	2016-12-31	960.627014	963.742981	947.236023	963.742981

366 rows × 5 columns

Fig 4.9: Analysis of the year 2015 and 2016

```python
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```

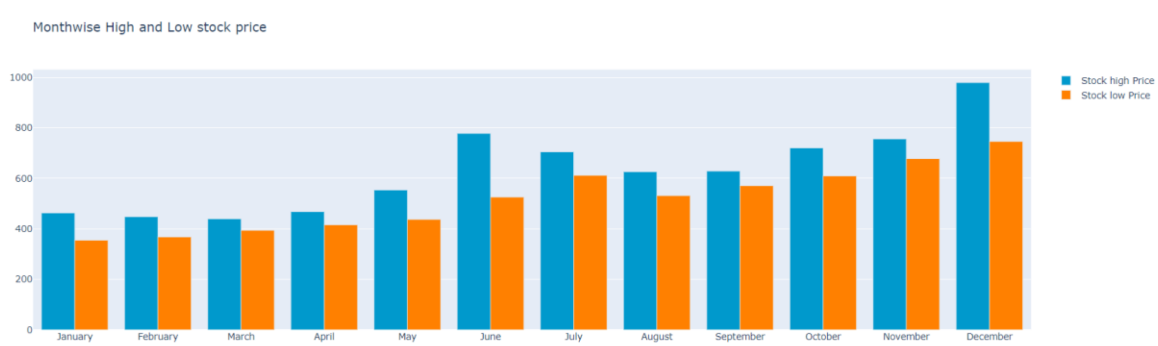


Fig 4.10: The year 2016 month-wise price

```
fig = px.line(y_2016, x=y_2016.Date, y=[y_2016['Open'], y_2016['Close'],
                                         y_2016['High'], y_2016['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.11: Bitcoin analysis chart 2016

Analysis of Year 2017

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2017 = maindf.loc[(maindf['Date'] >= '2017-01-01')
                     & (maindf['Date'] < '2018-01-01')]

y_2017.drop(y_2017[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
837	2017-01-01	963.658020	1003.080017	958.698975	998.325012
838	2017-01-02	998.617004	1031.390015	996.702026	1021.750000
839	2017-01-03	1021.599976	1044.079956	1021.599976	1043.839966
840	2017-01-04	1044.400024	1159.420044	1044.400024	1154.729980
841	2017-01-05	1156.729980	1191.099976	910.416992	1013.380005
...	...	...	...	...	...
1197	2017-12-27	16163.500000	16930.900391	15114.299805	15838.500000
1198	2017-12-28	15864.099609	15888.400391	13937.299805	14606.500000
1199	2017-12-29	14695.799805	15279.000000	14307.000000	14656.200195
1200	2017-12-30	14681.900391	14681.900391	12350.099609	12952.200195
1201	2017-12-31	12897.700195	14377.400391	12755.599609	14156.400391

365 rows × 5 columns

```
monthvise= y_2017.groupby(y_2017['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Fig 4.12: Analysis of the year 2017

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2017, x=y_2017.Date, y=[y_2017['Open'], y_2017['Close'],
                                         y_2017['High'], y_2017['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.13: Analysis of the year 2017

Analysis of Year 2018

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2018 = maindf.loc[(maindf['Date'] >= '2018-01-01')
                     & (maindf['Date'] < '2019-01-01')]

y_2018.drop(y_2018[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
1202	2018-01-01	14112.200195	14112.200195	13154.700195	13657.200195
1203	2018-01-02	13625.000000	15444.599609	13163.599609	14982.099609
1204	2018-01-03	14978.200195	15572.799805	14844.500000	15201.000000
1205	2018-01-04	15270.700195	15739.700195	14522.200195	15599.200195
1206	2018-01-05	15477.200195	17705.199219	15202.799805	17429.500000
...	...	...	...	...	...
1562	2018-12-27	3854.688477	3874.416992	3645.448486	3654.833496
1563	2018-12-28	3653.131836	3956.135986	3642.632080	3923.918701
1564	2018-12-29	3932.491699	3963.758789	3820.408691	3820.408691
1565	2018-12-30	3822.384766	3901.908936	3797.219238	3865.952637
1566	2018-12-31	3866.839111	3868.742920	3725.867432	3742.700439

365 rows × 5 columns

```
monthvise= y_2018.groupby(y_2018['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Fig 4.14: Analysis of the year 2018

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2018, x=y_2018.Date, y=[y_2018['Open'], y_2018['Close'],
                                          y_2018['High'], y_2018['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.15: Bitcoin analysis chart 2018

Analysis of Year 2019

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2019 = maindf.loc[(maindf['Date'] >= '2019-01-01')
                     & (maindf['Date'] < '2020-01-01')]

y_2019.drop(y_2019[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
1567	2019-01-01	3746.713379	3850.913818	3707.231201	3843.520020
1568	2019-01-02	3849.216309	3947.981201	3817.409424	3943.409424
1569	2019-01-03	3931.048584	3935.685059	3826.222900	3836.741211
1570	2019-01-04	3832.040039	3865.934570	3783.853760	3857.717529
1571	2019-01-05	3851.973877	3904.903076	3836.900146	3845.194580
...	...	...	...	...	...
1927	2019-12-27	7238.141113	7363.529297	7189.934082	7290.088379
1928	2019-12-28	7289.031250	7399.041016	7286.905273	7317.990234
1929	2019-12-29	7317.647461	7513.948242	7279.865234	7422.652832
1930	2019-12-30	7420.272949	7454.824219	7276.308105	7292.995117
1931	2019-12-31	7294.438965	7335.290039	7169.777832	7193.599121

365 rows × 5 columns

```
monthvise= y_2019.groupby(y_2019['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Fig 4.16: Analysis of the year 2019

```python
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2019, x=y_2019.Date, y=[y_2019['Open'], y_2019['Close'],
                                          y_2019['High'], y_2019['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

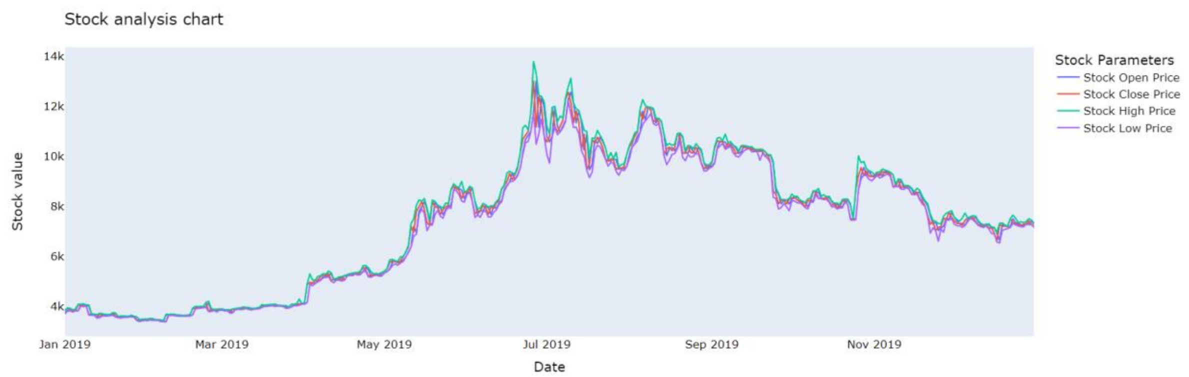


Fig 4.17: Bitcoin analysis chart 2019

Analysis of Year 2020

```python
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2020 = maindf.loc[(maindf['Date'] >= '2020-01-01')
                     & (maindf['Date'] < '2021-01-01')]

y_2020.drop(y_2020[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
1932	2020-01-01	7194.892090	7254.330566	7174.944336	7200.174316
1933	2020-01-02	7202.551270	7212.155273	6935.270020	6985.470215
1934	2020-01-03	6984.428711	7413.715332	6914.996094	7344.884277
1935	2020-01-04	7345.375488	7427.385742	7309.514160	7410.656738
1936	2020-01-05	7410.451660	7544.497070	7400.535645	7411.317383
...	...	...	...	...	...
2293	2020-12-27	26439.373047	28288.839844	25922.769531	26272.294922
2294	2020-12-28	26280.822266	27389.111328	26207.640625	27084.808594
2295	2020-12-29	27081.810547	27370.720703	25987.298828	27362.437500
2296	2020-12-30	27360.089844	28937.740234	27360.089844	28840.953125
2297	2020-12-31	28841.574219	29244.876953	28201.992188	29001.720703

366 rows × 5 columns

```python
monthvise= y_2020.groupby(y_2020['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
```

Fig 4.18: Analysis of the year 2020

```python
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2020, x=y_2020.Date, y=[y_2020['Open'], y_2020['Close'],
                                          y_2020['High'], y_2020['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

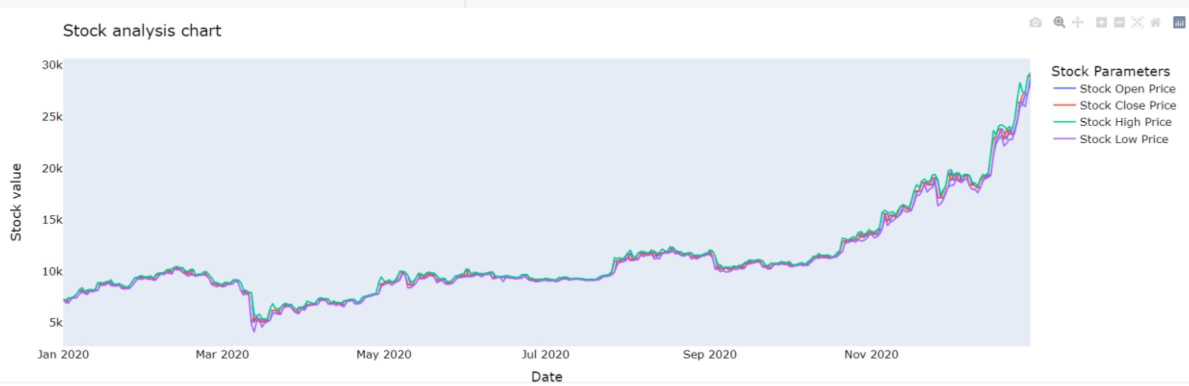


Fig 4.19: Bitcoin analysis chart 2020

Analysis of Year 2021

```python
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2021 = maindf.loc[(maindf['Date'] >= '2021-01-01')
                     & (maindf['Date'] < '2021-12-31')]

y_2021.drop(y_2021[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
2298	2021-01-01	28994.009766	29600.626953	28803.585938	29374.152344
2299	2021-01-02	29376.455078	33155.117188	29091.181641	32127.267578
2300	2021-01-03	32129.408203	34608.558594	32052.316406	32782.023438
2301	2021-01-04	32810.949219	33440.218750	28722.755859	31971.914063
2302	2021-01-05	31977.041016	34437.589844	30221.187500	33992.429688
...	...	...	...	...	...
2657	2021-12-26	50428.691406	51196.378906	49623.105469	50809.515625
2658	2021-12-27	50802.609375	51956.328125	50499.468750	50640.417969
2659	2021-12-28	50679.859375	50679.859375	47414.210938	47588.855469
2660	2021-12-29	47623.871094	48119.742188	46201.496094	46444.710938
2661	2021-12-30	46490.605469	47879.964844	46060.312500	47178.125000

364 rows × 5 columns

```python
monthvise= y_2021.groupby(y_2021['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Fig 4.20: Analysis of the year 2021

```python
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2021, x=y_2021.Date, y=[y_2021['Open'], y_2021['Close'],
                                          y_2021['High'], y_2021['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.21: Bitcoin analysis chart 2021

```python
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```

Fig 4.22: Month-wise comparison of stock open and close

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'])

fig = px.line(y_2022, x=y_2022.Date, y=[y_2022['Open'], y_2022['Close'],
                                          y_2022['High'], y_2022['Low']],
             labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.23:Stock analysis chart

Overall Analysis from 2014-2022

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_overall = maindf.loc[(maindf['Date'] >= '2014-09-17')
                     & (maindf['Date'] <= '2022-02-19')]

y_overall.drop(y_overall[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
0	2014-09-17	465.864014	468.174011	452.421997	457.334015
1	2014-09-18	456.859985	456.859985	413.104004	424.440002
2	2014-09-19	424.102997	427.834991	384.532013	394.795990
3	2014-09-20	394.673004	423.295990	389.882996	408.903992
4	2014-09-21	408.084991	412.425995	393.181000	398.821014
...	...	...	...	...	...
2708	2022-02-15	42586.464844	44667.218750	42491.035156	44575.203125
2709	2022-02-16	44578.277344	44578.277344	43456.691406	43961.859375
2710	2022-02-17	43937.070313	44132.972656	40249.371094	40538.011719
2711	2022-02-18	40552.132813	40929.152344	39637.617188	40030.976563
2712	2022-02-19	40022.132813	40246.027344	40010.867188	40126.429688

2713 rows × 5 columns

```
monthvise= y_overall.groupby(y_overall['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
```

Fig 4.24: Overall Analysis

```
fig = px.line(y_overall, x=y_overall.Date, y=[y_overall['Open'], y_overall['Close'],
                                              y_overall['High'], y_overall['Low']],
             labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```

Fig 4.25: Stock Analysis of prices

5. Building LSTM Model

- First Step is Preparing Data for Training and Testing
- Here we are just considering 1 year data for training data
- Since Bitcoin price has drastically flucated from 200 dollar in year 2014 to 15000 dollar in year 2018 to 3000 dollar in year 2019(theses values are apporx) so we will just consider 1 Year to avoid this type of flucation in the data.
- As we want to predict Close Price of the Bitcoin so we are just Considering Close and Date

```
# Lets First Take all the Close Price
closedf = maindf[['Date','Close']]
print("Shape of close dataframe:", closedf.shape)
```

```
Shape of close dataframe: (2713, 2)
```

```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Close Stock'})
fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')
fig.update_layout(title_text='Whole period of timeframe of Bitcoin close price 2014-2022', plot_bgcolor='white',
                  font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

Fig 4.26 : LSTM Model

Now we will Take data of just 1 Year

```
closedf = closedf[closedf['Date'] > '2021-02-19']
close_stock = closedf.copy()
print("Total data for prediction: ",closedf.shape[0])
```

```
Total data for prediction:  365
```

Fig 4.27: Timeframe of bitcoin price

```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Close Stock'})
fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')
fig.update_layout(title_text='Considered period to predict Bitcoin close price',
                  plot_bgcolor='white', font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

Fig 4.28: Considered period to Predict bitcoin

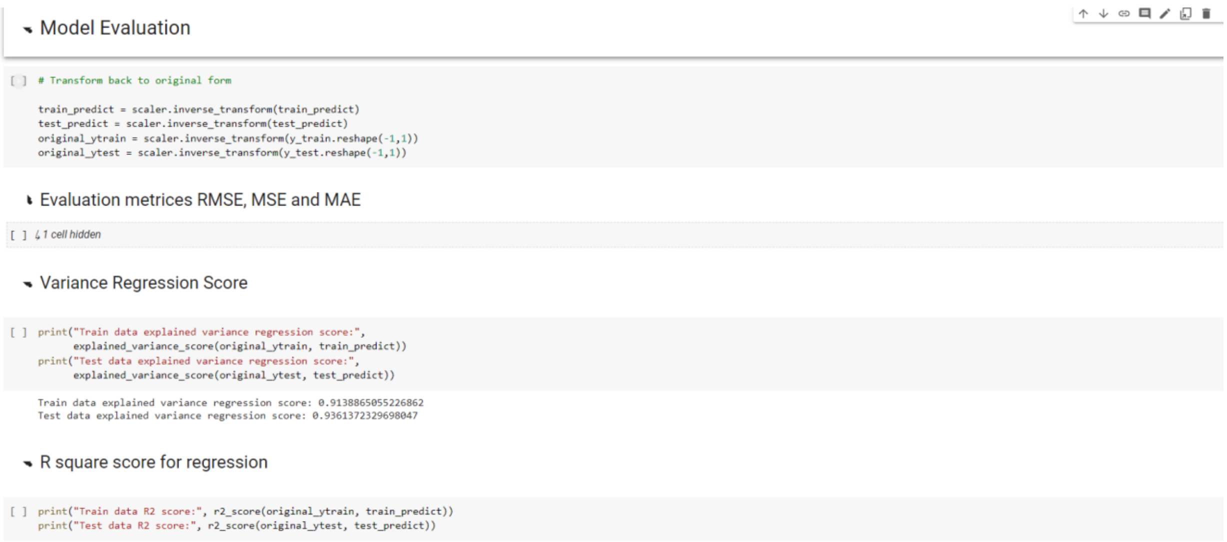


Fig 4.29: Model Evaluation

Fig 4.30: Comparison between original and predicted price

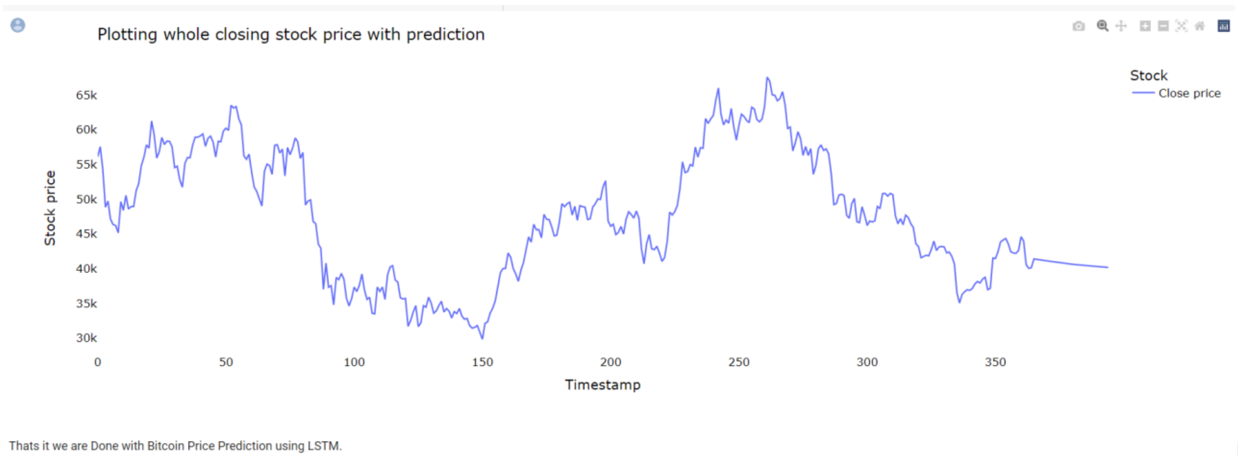


Fig 4.31: Closing Stock price with prediction

4. Conclusion

LSTM is an artificial recurrent neural network architecture for deep learning. Unlike standard neural networks, LSTMs have feedback connections. It can handle not just single data points but entire data ranges. A long-short-term memory (LSTM) network is a recurrent neural network capable of learning sequence dependencies in prediction problems. This is the desired behaviour in complex problem domains like machine translation, speech recognition, etc. LSTMs are a complex area of deep learning. LSTMs are often referred to as sophisticated RNNs. Vanilla RNN has no cellular state. They only have hidden states, which serve as the memory of the RNN. Meanwhile, LSTM has both a cell state and a hidden state.

5. References

[1] Sina E. Charandabi, Kamyar Kamyar(2021) -Prediction of Cryptocurrency Price Index Using Artificial Neural Networks, Proposed hybrid artificial neural network models to predict prices of cryptocurrency.
[2]Ahmed M. Khedr, Ifra Arif, Pravija Raj P V(2021) – Traditional statistical methods require a lot of statistical assumptions that could be unrealistic, leaving machine learning as the best technology in this field.
[3]Edwin, Lipo Wang (2017) - Described the features of Bitcoin and the following day's change in the price of Bitcoin using an Artificial Neural Network ensemble.
[4] Yeray Mezquita, Ana Belén Gil-González, Javier Prieto, and Juan Manuel Corchado (2021)-described that the solutions proposed are for detecting patterns and anomalies in cryptocurrency ecosystems.
[5] Lloyd Kasal, Mihir Shetty, Tanmay Nayak, Ramanath Pai, and Shilpa B(2022) proposed that numerous different kinds of neural networks may be utilized to analyse cryptocurrency values. The most successful of them all has been determined to be LSTM.
[6]Ahmed M. Khedr Ifra Arif Pravija Raj P V Magdi El-Bannany (2021)- proposed that traditional statistical methods, although simple to implement and interpret, require a lot of statistical assumptions
[7]Caporale, Guglielmo Maria Plastun, Alex (2018),-Cryptocurrency market Bitcoin overreaction momentum abnormal returns contrarian strategy trading robot.
[8]Al-Yahyaee KH, Mensi W(2020) -Multifractality, long-memory process, and efficiency hypothesis of six major cryptocurrencies using the time-rolling MF-DFA approach.
[9]Magdi El-Bannany, Saadat M. Alhashmi, Meenu Sreedharan(2016) - Cryptocurrency price prediction using traditional statistical and machine-learning techniques, this type of model approach called Genetic Algorithm based Selective Neural Network Ensemble.
[10]Ujan Mukhopadhyay; Anthony Skjellum; Oluwakemi; Hambolu; Jon Oakley; Richard Brooks(2022) proposed that Cryptocurrencies require robust and secure mining algorithms. In this paper, we survey and compare and contrast current mining techniques used by major Cryptocurrencies.
[11] Carmine Ventre, Michail Basios, Leslie Kanthan, David Martinez-Rego(2022), This survey analyzes the research distribution that characterizes cryptocurrency Research and distribution among properties.
[12] Sina E. Charandabi; Kamyar Kamyar (2021), This survey paper aims to present and compare multiple research papers that employed multiple neural networks.
[13] Saeed Alzahrani; Tugrul U. Daim(2019) -Paper suggests that the main factors driving the adoption decision are the acceptance by businesses as a payment method and the fast transfer of funds.
[14]Pravija Raj, Magdi El-Bannany, Saadat M.Alhashmi, Meenu Sreedharan(2021) –Research in this field uses traditional statistical and machine-learning techniques, making it hard to predict using a statistical approach.
[15] George S.Atsalakis(2021) –Fuzzy modelling demonstrating that the closed-loop or feedback control technique can cope with uncertainties associated with the dynamic behaviour of the price of Bitcoin, and achieve positive returns.