ADVANCED DATA SCIENCE

TERM PROJECT
AI /ML BASED CRYPTOCURRENCY(BITCOIN) PRICE FORECASTING SYSTEM.

UNIVERSITY OF THE WEST *of* SCOTLAND
UWS

Submitted by	Submitted to
Kiran babu Basnet	Prof. Zeeshan Parvez
Banner ID:B000728243	

School of Engineering and Computing

University of the West of Scotland

December 07, 2022

Table of Contents

ABSTRACT

This term project focuses on forecasting dominant cryptocurrency motivated by today s rapid market fluctua tion. Cryptocurrency is rising its presence on world's financial reshaping with geometrically increasing user base. Today's time is digital era and best possible each individual will be the part of this gigantic network con sider as stakeholders. Cryptos are tradable assets so financial market place has huge impact on it. Data driven statistical modelled with artificial intelligence specially machine learning approach have gained great import ance on financial marketplace. Finalize the most discriminating variable and use these variable's data use to f eed to advanced artificial intelligence (AI) frameworks of artificial neural network (ANN) and some machine lear ning models such as Long short-term memory (LSTM), regression, gradient boosting and random forest to predict the high-frequency time series of prices of highly impactful cryptocurrency such as Bitcoin (BTC). I use comprehensi ve feature set, including time series analysis focusing on closing price of bitcoins data to make prediction usi ng LSTM and Pycret . These most discriminating feature in cryptos market is closing value and using the clos ing value featured historical data, develop a mathematical representation that has acceptable fidelity to the da ta and an appropriate machine learning models are created to make forecasting for future. Moreover, it will e xplain the end-to-end process of performing forecasting, such as data sourcing and data preparation, explanatory data analysis, feature engineering, feature selection, model training techniques and model performance comparison and in terpretation in this project. Despite being significant downfall in the recent time, the predicted price over the closing price is seen some how accurate in the visual graph. The predicted value can say the accuracy of more then 90% which is also supported by theoretical and implemented model of LSTM. Regularization was evalu ated but deemed not necessary given the already high correlation, resulting in a coefficient of determination (R squared) accuracy of 96.8%.

Keywords: *Artificial Intelligence (AI), Machine Learning, Cryptocurrency, Financial Forecasting*

1. INTRODUCTION

1.1 Background

As of 2020, more than 7000 cryptocurrencies are actively trading in more than 20000 online exchanges. Thei r total market capitalization has exceeded USD 300 billion. Although these cryptocurrencies are not backed by any tangible assets, they gain trust from users by publicly disclosing the full creation and transaction histo ry in peerpeer blockchain networks. Each transaction in the blockchain consists of transferring a virtual valu e from a virtual identity, i.e., a blockchain address or a set of addresses, to another. Among these variety of cry ptos Bitcoin is a market leading and known to every individual. The sizes of transaction records are quickly e xpanding. The total transaction volumes of Bitcoin is highest at the middle of 2021. Although technically cha llenging in extracting, transforming, and analyzing, these transaction histories have given us an unprecedent ed opportunity to study the panorama of human behavior in a complex economic environment. Reid and Harr igan conducted the first study on the entire cryptocurrency transaction history (up to mid-2021), revealed emerging structure from the Bitcoin flow network, and demonstrated the transaction history and being a futuristic technological behaviors data mining from cryptocurrency transactions has emphasize d into a large body of research and been successfully applied in assisting multiple law enforcement actions, in cluding ceasing the thenlargest darknet market Silk Road in late era and arresting suspects in a major theft fro m the then-largest cryptocurrency exchange.

1.2 Motivation

The main motive of the generation of this report is to study of bitcoins and make prediction on the basis of the available data. To know why bitcoin is being so popular and the first crypto coin consider as blockchain 1.0 B y employing machine learning , make a reasonable prediction so that stakeholder benefitted using the result a nd to know how people are predicting the future and investing. Direct access to the data is going to help, So go ing through the methodical approach of data science and draw a meaningful conclusion based on data. Data s cience and its tool are not only the theoretical approach, but it is a proven facts with mathematical proof, It is n ot only the probability, but its actuality. Following are some aim of generating the report.

- The main aim of this project is to make a reasonable price forecasting for the top-rated cryptocurrency called bitcoin by considering most discriminating factor's data built and train t he model by employing artificial intelligence (AI)and machine learning(ML)

- Understand the working of new technology called Blockchain and its biggest application till now cal led cryptocurrency (The earliest one "Bitcoin") and its influence in financial marketplace and future prediction.

1.3 Problem

This is a kind of transitional time for the entire world economically i.e., the transition from our conventional monetary system to new digital, technology based crypto economic demography. The entire world is facing economic recession. It has only the history of a decade, but its impact is like a history of thousand's year. Cry ptocurrency(bitcoin) and its parent technology viz Blockchain has started in 2008 AD and introduced its first currency in 2009 AD viz Bitcoin is considered as Blockchain 1.0 after that its gradual development and opti mization found that will become the hot ball in the hand of today's world. In the specific circumstances I took consider the crypto history from 2013 AD to 2021 AD, so that we will able to figure out the each ups and dow n to the crypto world and also help to visualize the upcoming days for the world's digital economy.

1.4 General Interest

Though the cryptocurrency has not been fully understood by all the people, only a handful of people has some how deep understanding of it but everyone has interest to know about it and it is becoming a way of life in futu re for every citizen. Some early adopter of cryptos with good understanding get rich but some lost almost eve rything. The make general understanding of crypto, it's working, impact, popularity, flexibility, investment, profitability and many other unseen dimensions on futuristic aspect in multi-
disciplinary view for the stakeholders.

2.Literature

2.1Forecast ability?

The data features of bitcoin, such as its high volatility, high nonlinearity, no stationarity, nonlinear dynamics , lack of periodicity, presence of a spectrum of scaling components, noisy data, and unpredictability, necessit ate special attention when predicting their price. A variety of statistical, machine learning, and deep learning methods and techniques, including liner regression (LR), autoregressive integrated moving average (ARIM A), Linear discriminant analysis (LDA),Data transmission(DT), RF, XGBoost, Quantitative descriptive an alysis(QDA),support vector machine(SVM), and long short term memory(LSTM), have been utilized by re searchers all around the world to forecast cryptos price. The ability to forecast the price of bitcoin using hybri d models like Convolution neural network (CNN)and new generation of recurrent neural network called GR U has been demonstrated in research (Edwards, 2022). This specific report is developed on the basis of using LSTM which is the special modification of Recurrent Neural Network(RNN) and a well known python libra ry called Pycret.

2.2Artificial Intelligence (AI) and Machine Learning (ML)?

The simulation of human intellect by computers is known as artificial intelligence (AI). It provides excellent chances to boost client satisfaction, democratize financial services, guarantee consumer protection, and vast ly enhance risk management. The degree to which the internal workings of an AI system can be described in t erms of human language is represented by AI-based prediction models.

In the field of AI known as machine learning (ML), algorithms are used to learn from data in order to make pr edictions or judgments in the future. Different algorithms can assist traders in predicting the price patterns of cryptocurrencies. Forecasting research is, of course, in the forefront. Among all baselines for the price predic tion and fluctuation prediction problem, attentive LSTM network and an Embedding Network produce high er state-of-the-
art performance. compares well to the conventional fully connected deep neural networks in that it accepts va rious cryptocurrency data as inputs and manages them individually to leverage useful information from each cryptocurrency separately (Jhfree, 2022).

Unquestionably one of the most difficult and unstable investment types is cryptocurrency. Fortunately, user are now able to diversify their portfolios and new investors are appearing owing to the surge in the developm ent of cryptocurrencies. Additionally, as information technology has lately advanced, consumers are now ab le to browse websites and utilise a variety of applications on portable devices in addition to PCs, such as smart phones and tablets.

The created technique enables forecasting of the price of cryptocurrencies. This rate is predicted using machi ne learning and data mining. In order to train bitcoin prices as time series data effectively, LSTM, RNN, Deci sion tree, ANN, and Linear regression are employed. In this technique, the movement of a cryptocurrency ac ross a range of time periods may be predicted. Different algorithms need different amounts of time to build m odels and have varying degrees of predictive accuracy. There have been a few research that seek to comprehe nd the cryptocurrency time series and develop statistical models to recreate and anticipate price dynamics, de spite the fact that there are currently little efforts on cryptocurrency price analysis and prediction.

3.Method of Solving

Data acquisition, Performing the Data Pre-processing like importing the required libraries and loading them, importing the csv files, Handling the missi ng and perform the cleaning process. Deploy some inbuilt function to analyze the data frames called explorat ory data analysis to know data inside,, Visualizing the data, finding the relationship between the variables, sp litting it into train and test data for a model. The graphical depiction of information and data is known as data visualization. Data visualization tools make it easy to examine and comprehend trends, outliers, and patterns in data by employing visual components like charts, graphs, and maps. Employ the machine learning algorit hm to make prediction .Professionals' ability to utilize data to make choices and use graphics to communicat e stories about when data informs the who, what, when, where, and how is becoming increasingly valuable.

4.Data/ Data sets

All the data sets has taken from following source .

https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory/metadata

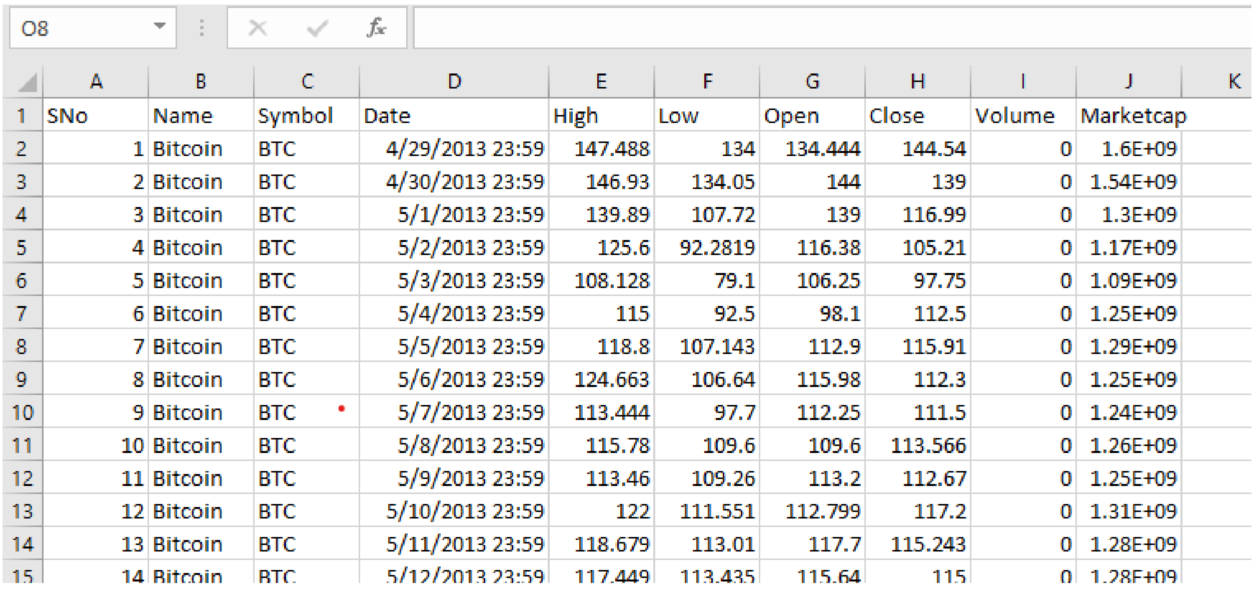
A properly formatted csv formatted data secondary source from data community is consider as genuine and dependable one. Clearly formatted 2 Mega Bytes of csv archived data is used for the calculation. The open da ta set is legal and ethical . The data set is some how clean ,all the data set may some missing values . The data s et has already from the trustable source with large community so it has gathered via variety of source and also formatted to public use. These data sets are widely used to know more about crypto currency in multi-disciplinary aspect and timely updated.

According to my need I only consider Bitcoin data and work on it , Rest of the data are overlooked.

All the formatted data set has mainly six calculative, alphanumeric value under consideration. which are liste d below

- Highest value
- Lowest value
- Opening value
- Closing value
- Volume of transaction
- Total market capitalization

All the data are with their respective dates and time . A properly classified data sets with distinct numeric al values of each coins makes the users easier to perform analysis over it and draw a meaningful conclusi on.



SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
1	Bitcoin	BTC	4/29/2013 23:59	147.488	134	134.444	144.54	0	1.6E+09
2	Bitcoin	BTC	4/30/2013 23:59	146.93	134.05	144	139	0	1.54E+09
3	Bitcoin	BTC	5/1/2013 23:59	139.89	107.72	139	116.99	0	1.3E+09
4	Bitcoin	BTC	5/2/2013 23:59	125.6	92.2819	116.38	105.21	0	1.17E+09
5	Bitcoin	BTC	5/3/2013 23:59	108.128	79.1	106.25	97.75	0	1.09E+09
6	Bitcoin	BTC	5/4/2013 23:59	115	92.5	98.1	112.5	0	1.25E+09
7	Bitcoin	BTC	5/5/2013 23:59	118.8	107.143	112.9	115.91	0	1.29E+09
8	Bitcoin	BTC	5/6/2013 23:59	124.663	106.64	115.98	112.3	0	1.25E+09
9	Bitcoin	BTC	5/7/2013 23:59	113.444	97.7	112.25	111.5	0	1.24E+09
10	Bitcoin	BTC	5/8/2013 23:59	115.78	109.6	109.6	113.566	0	1.26E+09
11	Bitcoin	BTC	5/9/2013 23:59	113.46	109.26	113.2	112.67	0	1.25E+09
12	Bitcoin	BTC	5/10/2013 23:59	122	111.551	112.799	117.2	0	1.31E+09
13	Bitcoin	BTC	5/11/2013 23:59	118.679	113.01	117.7	115.243	0	1.28E+09
14	Bitcoin	BTC	5/12/2013 23:59	117.449	113.435	115.64	115	0	1.28E+09

Fig 1: Data sets for Bitcoin

4.1.Feature Exploration

The goal here is to target features that have a high correlation to Bitcoin, but ideally reside outside the Bitcoin universe (an example of this would be cryptocurrency universe market capitalization, of which the Bitcoin market capitalization represents almost 35% per coinmarketcap.com at the time of this writing). Features considered include:

Bitcoin related:

- Cryptocurrency universe market capitalization
- Current price
- Volume
- Number of transactions
- Average block size
- Transaction fees
- Unique addresses
- Hash rate

5.Data Science Pipeline

The complete data science pipeline for this specific project involves following major steps.

- Importing the libraries
- Reading / loading the data
- Data pre-processing
- (Exploratory data analysis)EDA
- Data modelling
- Prediction

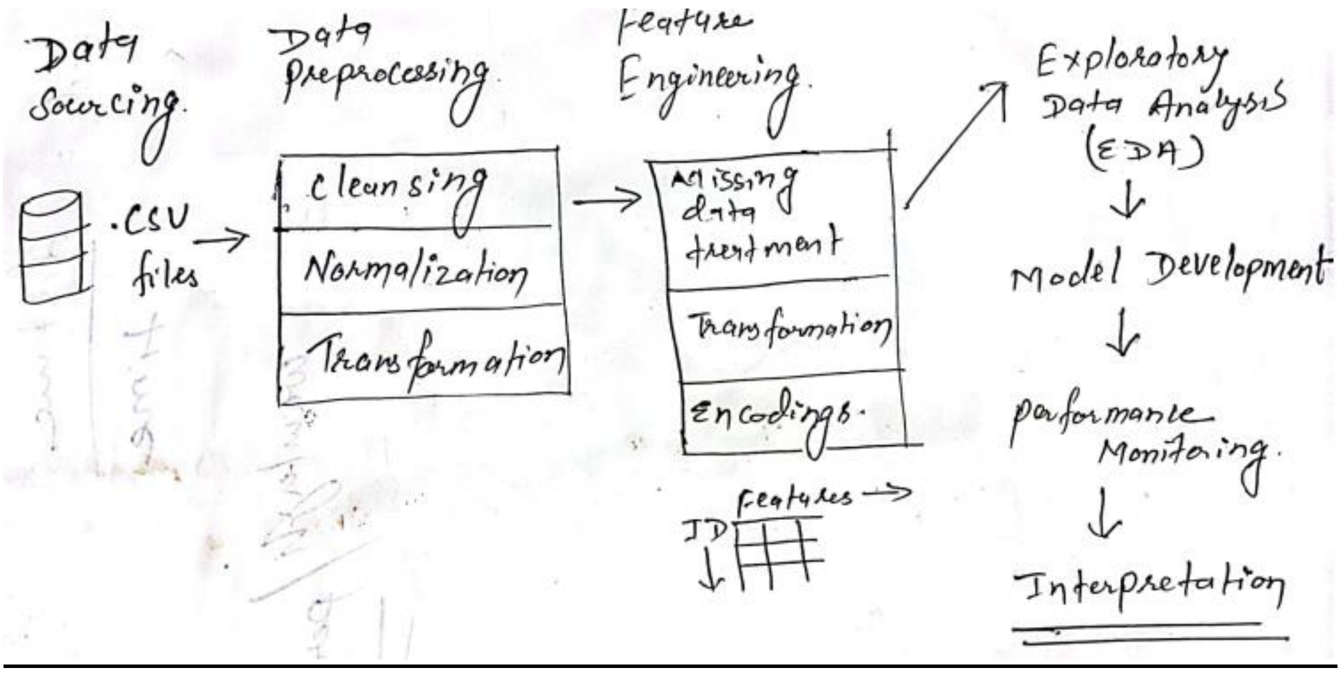
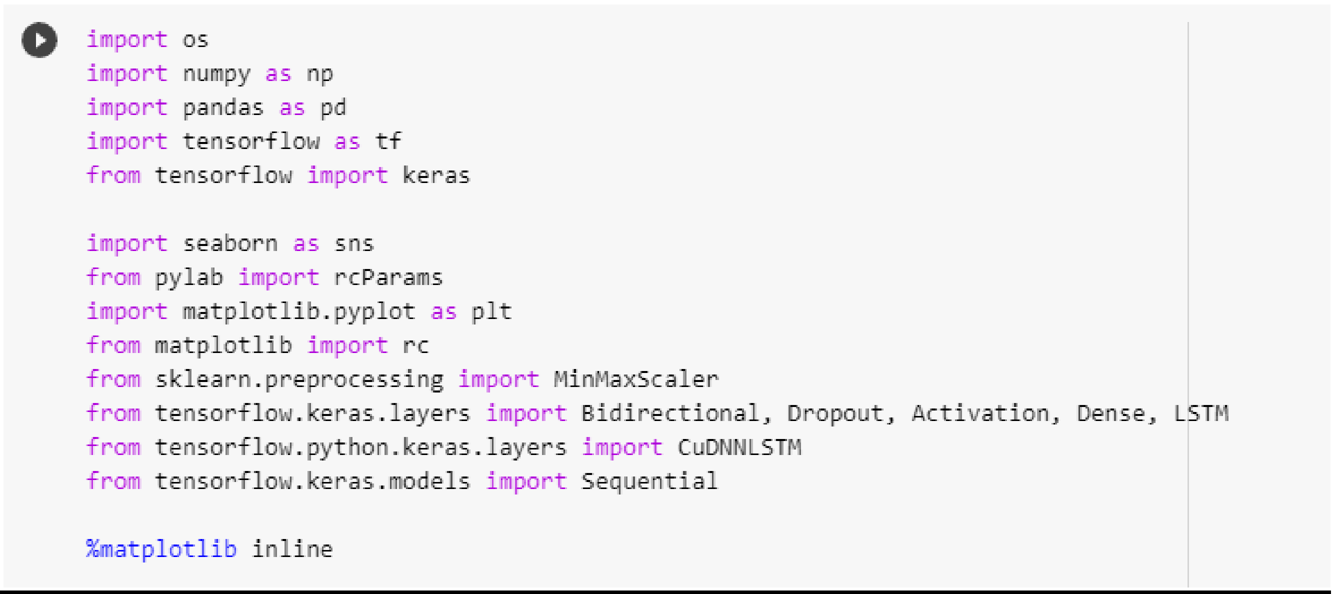


Fig1: Data science Pipeline / Work flow

6.Importing Required Libraries

6.1For LSTM Model



```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras

import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import Bidirectional, Dropout, Activation, Dense, LSTM
from tensorflow.python.keras.layers import CuDNNLSTM
from tensorflow.keras.models import Sequential

%matplotlib inline
```

Fig1 : Importing required library

Os module in python is for creating and managing directory structure, fetching the content and identify curre nt directory.

NumPy is library in python for large multi dimensional array and matrix and for mathematical function as w ell.

Pandas is library in python for manipulating and analysis the data mainly for the numerical calculation.

Matplptlib is a plotting library in python for visualization.

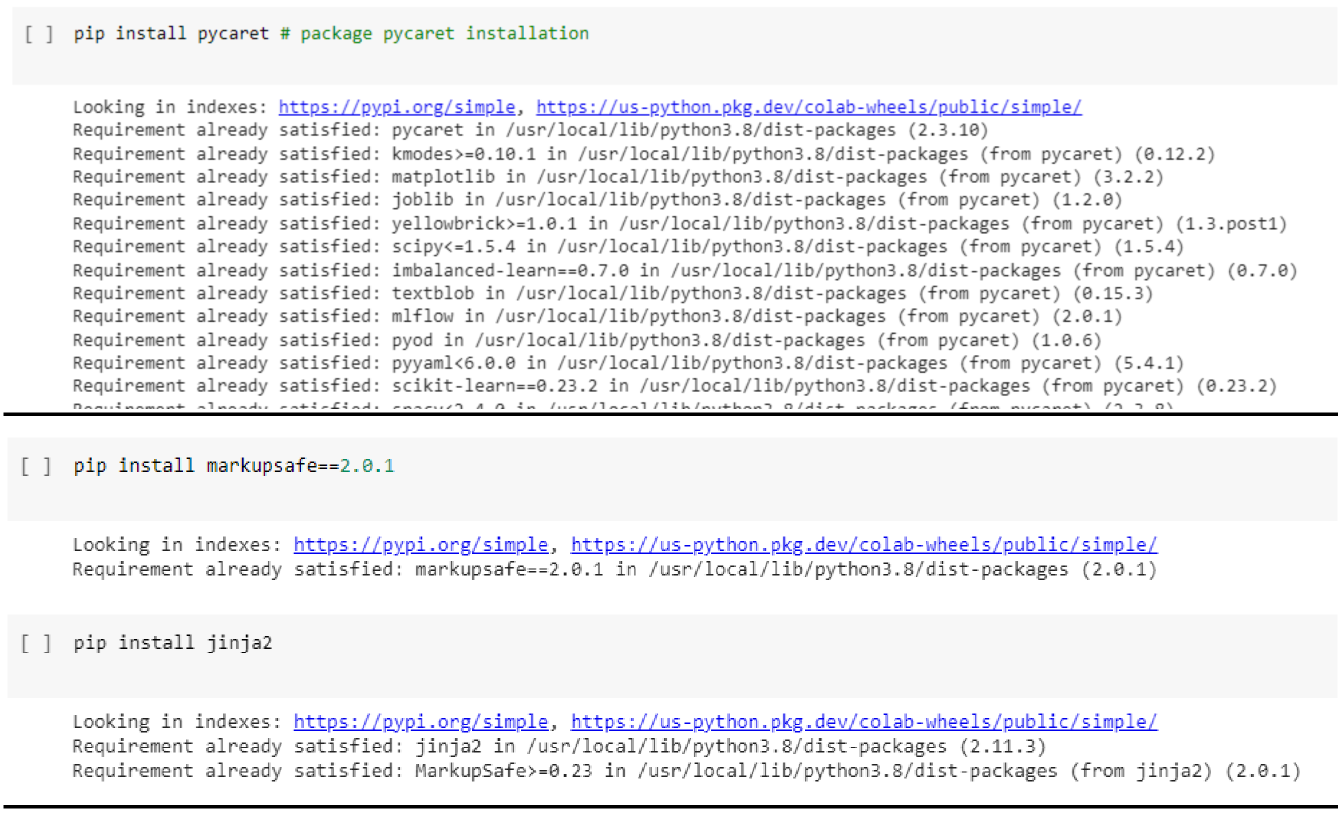
Seaborn is a data visualization library based on matplotlib.

Scikit-learn is a ML librry specially for classification, regression and clustering.

Tensorflow is a ML library for training and inference of neural network.

Keras is python library for AI/ML act as inference for tensorflow.

<u>For Pycret Model</u>



```
[ ]  pip install pycaret # package pycaret installation

     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Requirement already satisfied: pycaret in /usr/local/lib/python3.8/dist-packages (2.3.10)
     Requirement already satisfied: kmodes>=0.10.1 in /usr/local/lib/python3.8/dist-packages (from pycaret) (0.12.2)
     Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from pycaret) (3.2.2)
     Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from pycaret) (1.2.0)
     Requirement already satisfied: yellowbrick>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from pycaret) (1.3.post1)
     Requirement already satisfied: scipy<=1.5.4 in /usr/local/lib/python3.8/dist-packages (from pycaret) (1.5.4)
     Requirement already satisfied: imbalanced-learn==0.7.0 in /usr/local/lib/python3.8/dist-packages (from pycaret) (0.7.0)
     Requirement already satisfied: textblob in /usr/local/lib/python3.8/dist-packages (from pycaret) (0.15.3)
     Requirement already satisfied: mlflow in /usr/local/lib/python3.8/dist-packages (from pycaret) (2.0.1)
     Requirement already satisfied: pyod in /usr/local/lib/python3.8/dist-packages (from pycaret) (1.0.6)
     Requirement already satisfied: pyyaml<6.0.0 in /usr/local/lib/python3.8/dist-packages (from pycaret) (5.4.1)
     Requirement already satisfied: scikit-learn==0.23.2 in /usr/local/lib/python3.8/dist-packages (from pycaret) (0.23.2)
```

```
[ ]  pip install markupsafe==2.0.1

     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Requirement already satisfied: markupsafe==2.0.1 in /usr/local/lib/python3.8/dist-packages (2.0.1)
```

```
[ ]  pip install jinja2

     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (2.11.3)
     Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from jinja2) (2.0.1)
```

Fig2: Installing Package for Pycret Model

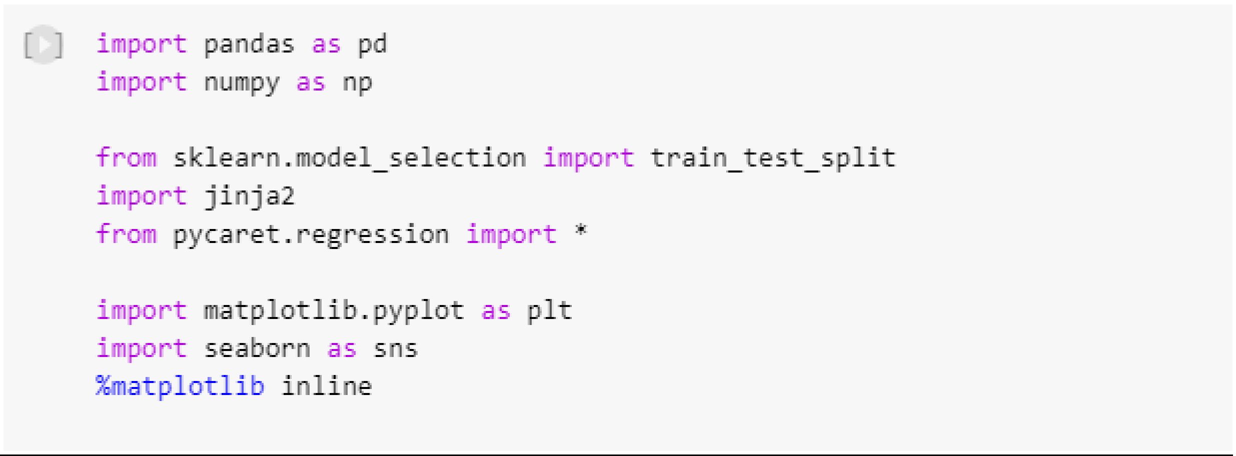
To access the functionality of Pycret , need to install the Pycret package along with markupsafe and jinja2.

Pycret is low code ML library for EDA, Pre-processing, Modelling , Training and MLOPS.

Markupsafe is for dealing with text and special character to wrap in markup.

Jinja2 is fast templating engine.

All others libraries functionality is same as LSTM model.



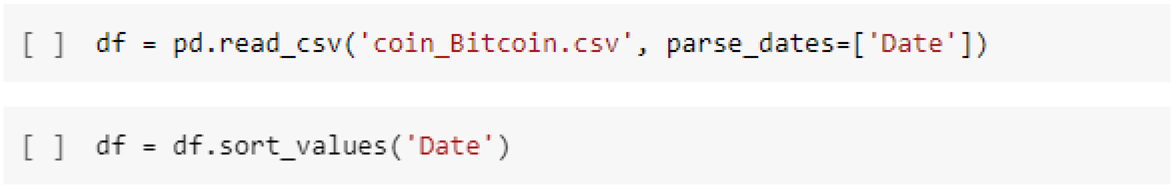
```
[ ] import pandas as pd
    import numpy as np

    from sklearn.model_selection import train_test_split
    import jinja2
    from pycaret.regression import *

    import matplotlib.pyplot as plt
    import seaborn as sns
    %matplotlib inline
```

Fig3: Required library for Pycret Model

7. Data pre-processing

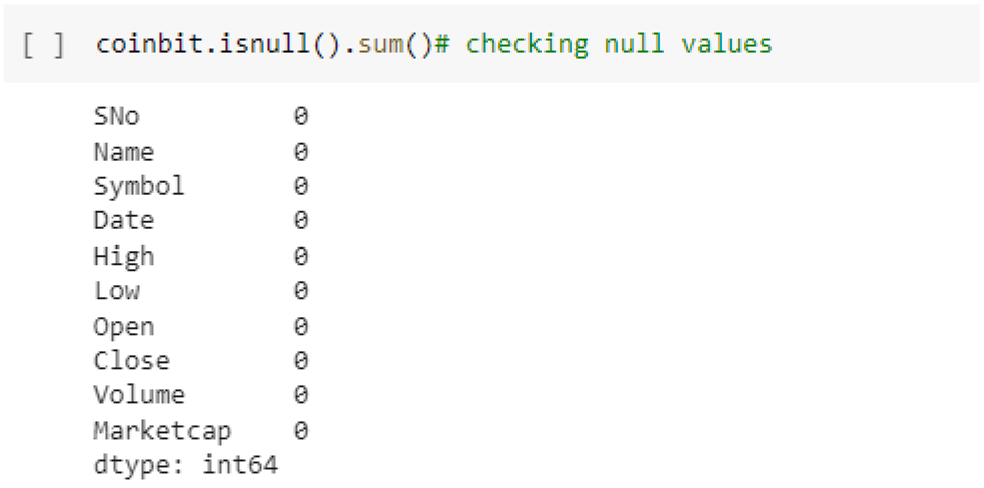
Though the data has been retrieved from trusted source, while studying and understanding the data I figure ou t some errors, so I removes those errors basically I perform cleansing . the error may be typing error are I also delete incomplete and repeated data. I also need to round up to ceiling or in floor according to the nature of dat a.

The data sets have already been classified; each coin has its own respective csv file so not much integration ne ed to perform to find final data sets. As there are similar data row consistently which I reduced to limited rows which can consider as data transformation. It mainly focuses on data cleansing and transformation to gain fin al data sets.



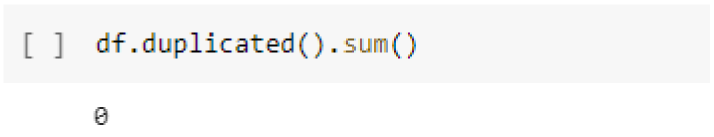
```
[ ] df = pd.read_csv('coin_Bitcoin.csv', parse_dates=['Date'])

[ ] df = df.sort_values('Date')
```

Fig4 : Reading the data and sorting it according to data .



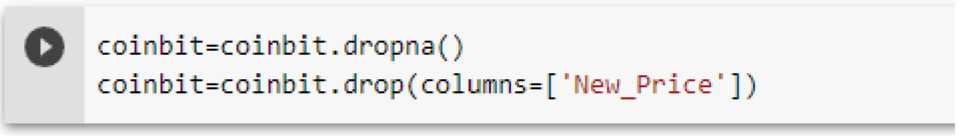
```
[ ]  coinbit.isnull().sum()# checking null values

     SNo          0
     Name         0
     Symbol       0
     Date         0
     High         0
     Low          0
     Open         0
     Close        0
     Volume       0
     Marketcap    0
     dtype: int64
```

Fig5: Checking for null values in the data set



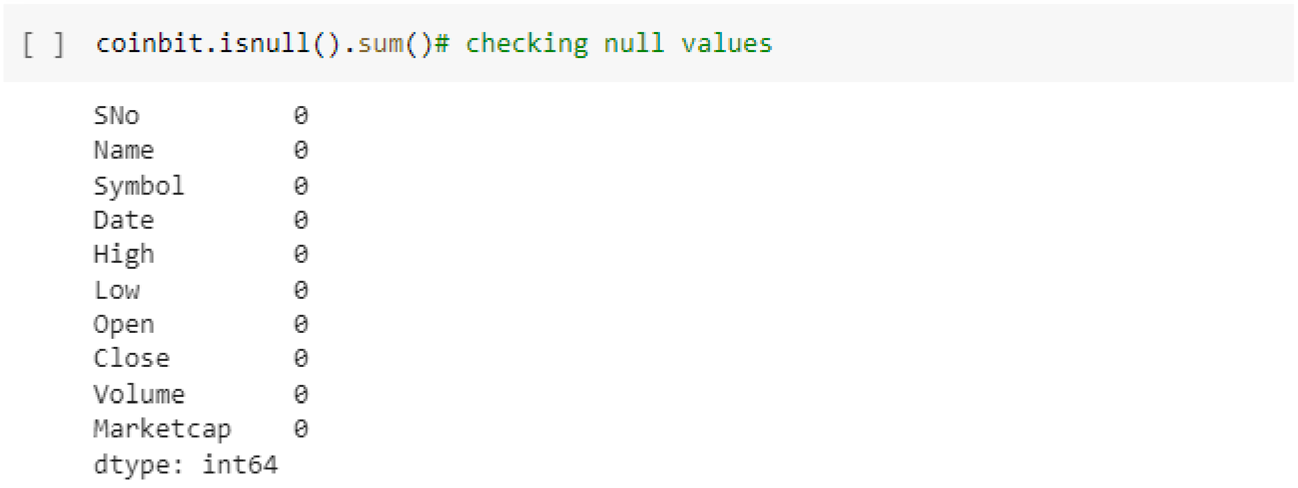
```
[ ]  df.duplicated().sum()

     0
```

Fig 6: Checking for Duplicate row



```
coinbit=coinbit.dropna()
coinbit=coinbit.drop(columns=['New_Price'])
```

Fig 7: Dropping the unused column



```
[ ]  coinbit.isnull().sum()# checking null values

     SNo          0
     Name         0
     Symbol       0
     Date         0
     High         0
     Low          0
     Open         0
     Close        0
     Volume       0
     Marketcap    0
     dtype: int64
```

Fig 8: Checking Null values

While loading the data then the preliminary step is sorting for sequential execution. Checking for the null val ues and dropping the unused columns. Data pre-processing is same for both LSTM and Pycret model.

8.EDA (Exploratory Data Analysis) -Qualitative

The first step is to collect the relevant data. Thereby, I implement data gathering, pre-processing, and model building using the Python programming and the libraries like TensorFlow, scikit-learn, numpy and pandas. After that actual EDA initiate to look not the data for getting the details . Following picture show detail EDA process.

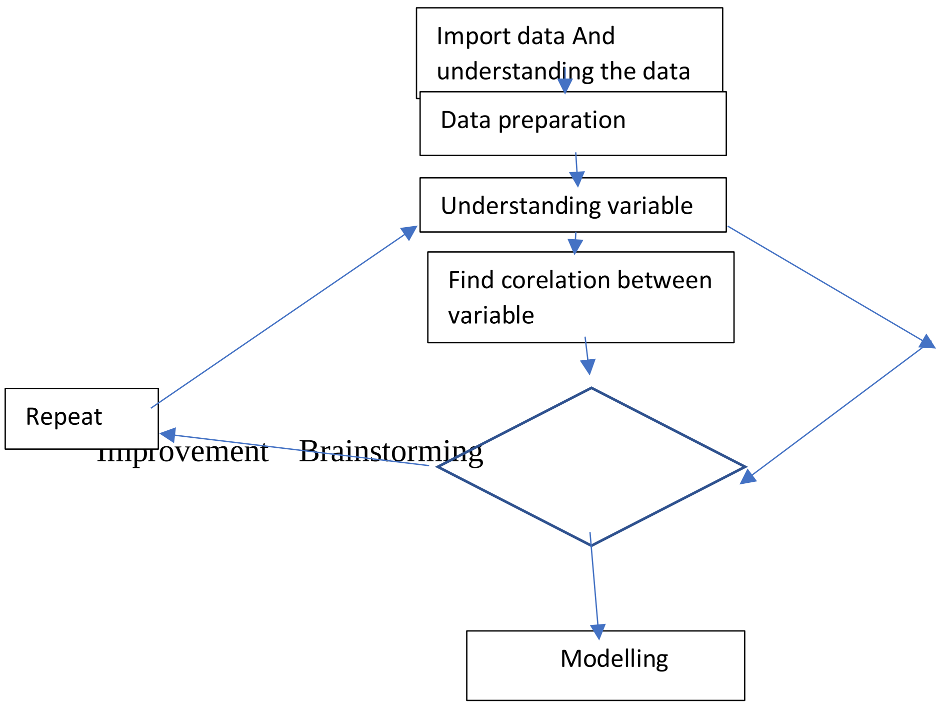
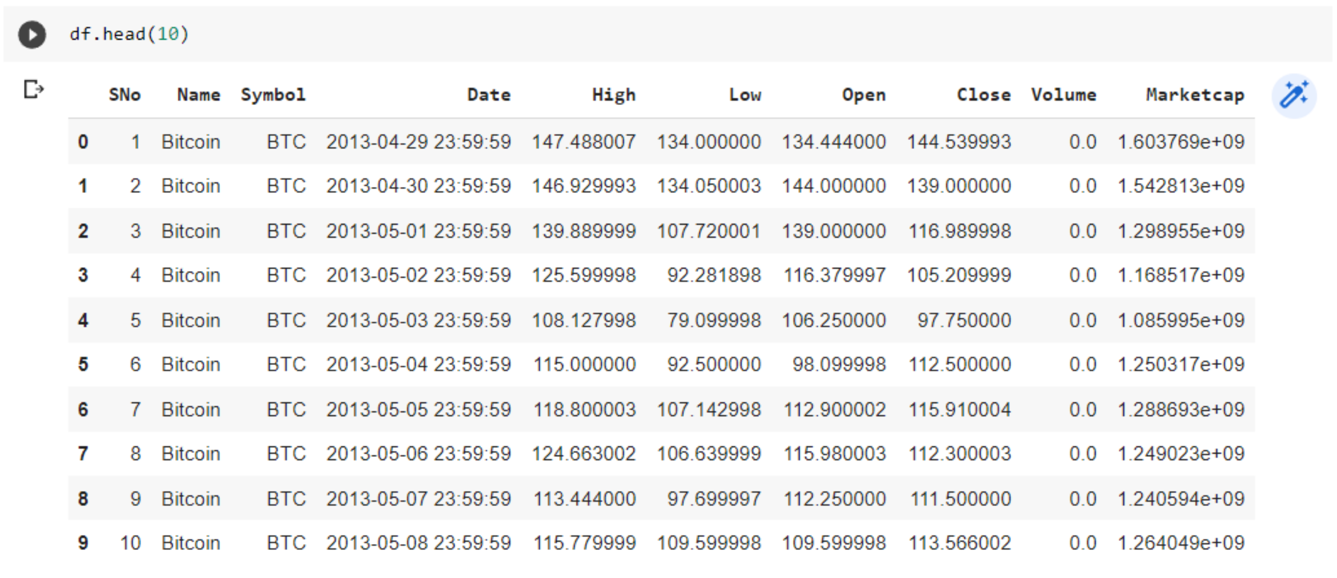


Fig9: EDA Process

<u>8.1 EDA Process for LSTM</u>

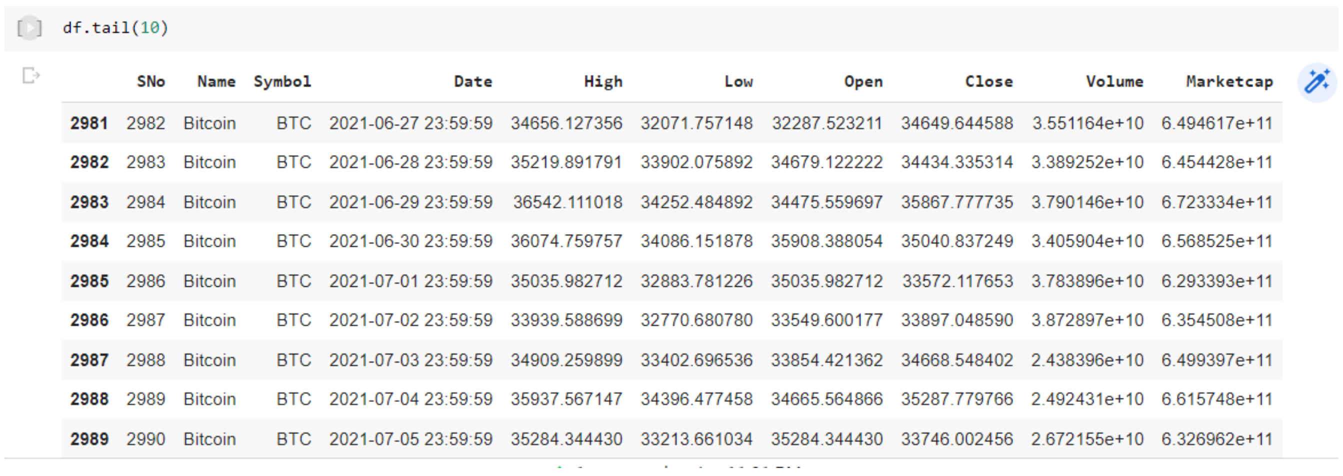
<u>8.1.1 Statistical analysis</u>



```
df.head(10)
```

```
   SNo     Name  Symbol                 Date        High         Low        Open       Close  Volume     Marketcap
0    1  Bitcoin     BTC  2013-04-29 23:59:59  147.488007  134.000000  134.444000  144.539993     0.0  1.603769e+09
1    2  Bitcoin     BTC  2013-04-30 23:59:59  146.929993  134.050003  144.000000  139.000000     0.0  1.542813e+09
2    3  Bitcoin     BTC  2013-05-01 23:59:59  139.889999  107.720001  139.000000  116.989998     0.0  1.298955e+09
3    4  Bitcoin     BTC  2013-05-02 23:59:59  125.599998   92.281898  116.379997  105.209999     0.0  1.168517e+09
4    5  Bitcoin     BTC  2013-05-03 23:59:59  108.127998   79.099998  106.250000   97.750000     0.0  1.085995e+09
5    6  Bitcoin     BTC  2013-05-04 23:59:59  115.000000   92.500000   98.099998  112.500000     0.0  1.250317e+09
6    7  Bitcoin     BTC  2013-05-05 23:59:59  118.800003  107.142998  112.900002  115.910004     0.0  1.288693e+09
7    8  Bitcoin     BTC  2013-05-06 23:59:59  124.663002  106.639999  115.980003  112.300003     0.0  1.249023e+09
8    9  Bitcoin     BTC  2013-05-07 23:59:59  113.444000   97.699997  112.250000  111.500000     0.0  1.240594e+09
9   10  Bitcoin     BTC  2013-05-08 23:59:59  115.779999  109.599998  109.599998  113.566002     0.0  1.264049e+09
```

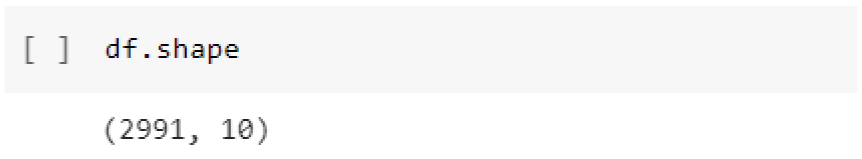
Fig 10: Reading 10 rows from head



```
df.tail(10)
```

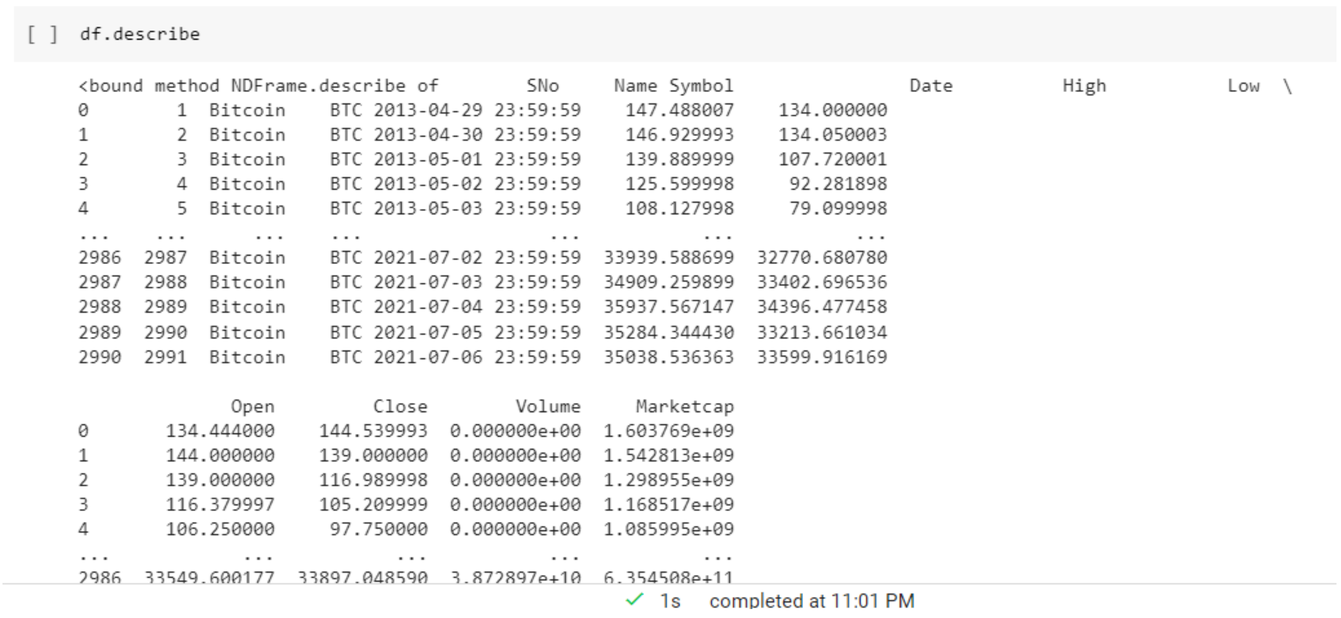
```
       SNo     Name  Symbol                 Date          High           Low          Open         Close        Volume     Marketcap
2981  2982  Bitcoin     BTC  2021-06-27 23:59:59  34656.127356  32071.757148  32287.523211  34649.644588  3.551164e+10  6.494617e+11
2982  2983  Bitcoin     BTC  2021-06-28 23:59:59  35219.891791  33902.075892  34679.122222  34434.335314  3.389252e+10  6.454428e+11
2983  2984  Bitcoin     BTC  2021-06-29 23:59:59  36542.111018  34252.484892  34475.559697  35867.777735  3.790146e+10  6.723334e+11
2984  2985  Bitcoin     BTC  2021-06-30 23:59:59  36074.759757  34086.151878  35908.388054  35040.837249  3.405904e+10  6.568525e+11
2985  2986  Bitcoin     BTC  2021-07-01 23:59:59  35035.982712  32883.781226  35035.982712  33572.117653  3.783896e+10  6.293393e+11
2986  2987  Bitcoin     BTC  2021-07-02 23:59:59  33939.588699  32770.680780  33549.600177  33897.048590  3.872897e+10  6.354508e+11
2987  2988  Bitcoin     BTC  2021-07-03 23:59:59  34909.259899  33402.696536  33854.421362  34668.548402  2.438396e+10  6.499397e+11
2988  2989  Bitcoin     BTC  2021-07-04 23:59:59  35937.567147  34396.477458  34665.564866  35287.779766  2.492431e+10  6.615748e+11
2989  2990  Bitcoin     BTC  2021-07-05 23:59:59  35284.344430  33213.661034  35284.344430  33746.002456  2.672155e+10  6.326962e+11
```

Fig 11: Reading 10 rows from tail of data set



```
[ ]  df.shape

(2991, 10)
```

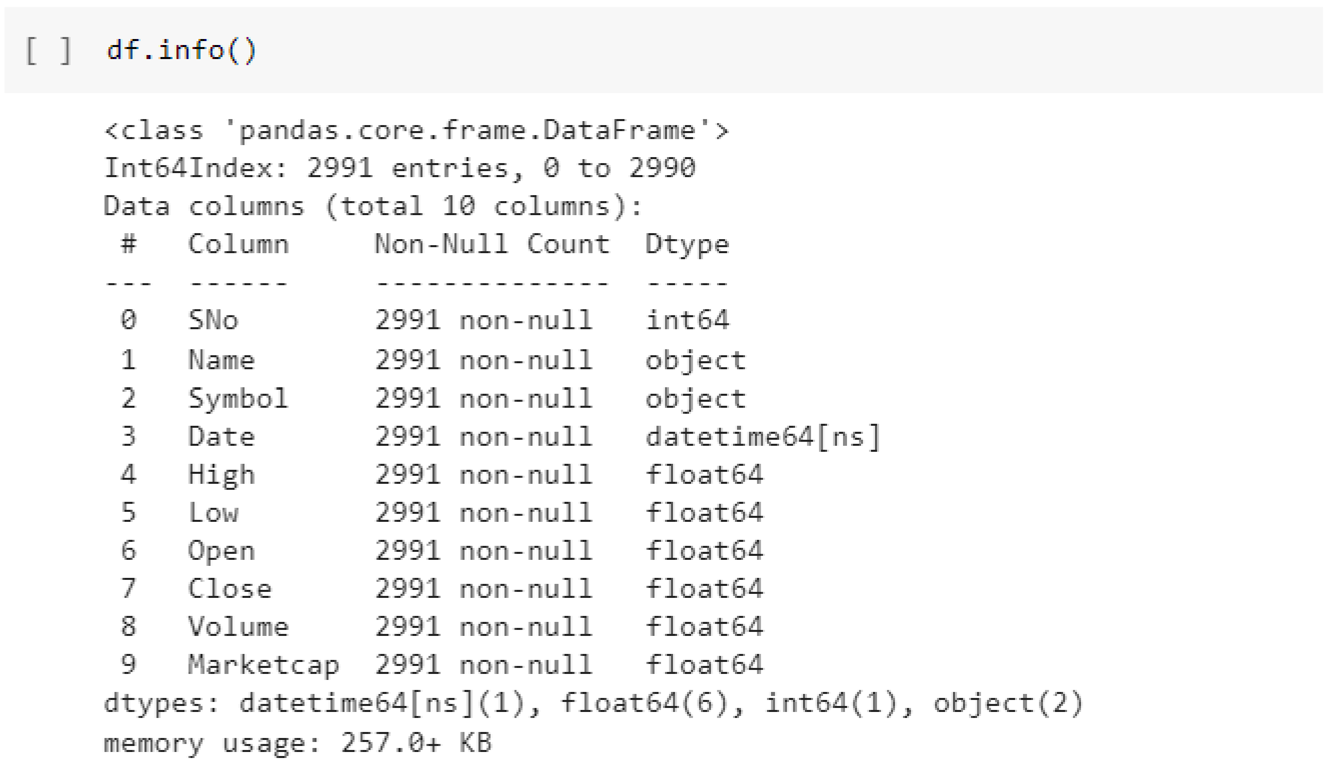
Fig12: Counting rows and column of data set



```
[ ]  df.describe

<bound method NDFrame.describe of        SNo     Name Symbol                Date          High           Low  \
0        1  Bitcoin    BTC 2013-04-29 23:59:59    147.488007    134.000000
1        2  Bitcoin    BTC 2013-04-30 23:59:59    146.929993    134.050003
2        3  Bitcoin    BTC 2013-05-01 23:59:59    139.889999    107.720001
3        4  Bitcoin    BTC 2013-05-02 23:59:59    125.599998     92.281898
4        5  Bitcoin    BTC 2013-05-03 23:59:59    108.127998     79.099998
...    ...      ...    ...                 ...           ...           ...
2986  2987  Bitcoin    BTC 2021-07-02 23:59:59  33939.588699  32770.680780
2987  2988  Bitcoin    BTC 2021-07-03 23:59:59  34909.259899  33402.696536
2988  2989  Bitcoin    BTC 2021-07-04 23:59:59  35937.567147  34396.477458
2989  2990  Bitcoin    BTC 2021-07-05 23:59:59  35284.344430  33213.661034
2990  2991  Bitcoin    BTC 2021-07-06 23:59:59  35038.536363  33599.916169

              Open         Close        Volume     Marketcap
0       134.444000    144.539993  0.000000e+00  1.603769e+09
1       144.000000    139.000000  0.000000e+00  1.542813e+09
2       139.000000    116.989998  0.000000e+00  1.298955e+09
3       116.379997    105.209999  0.000000e+00  1.168517e+09
4       106.250000     97.750000  0.000000e+00  1.085995e+09
...            ...           ...           ...           ...
2986  33549.600177  33897.048590  3.872897e+10  6.354508e+11
```

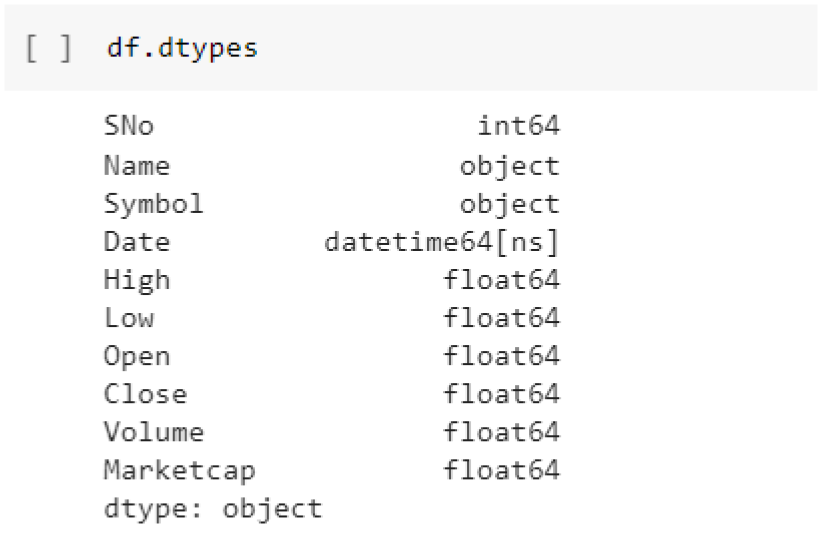
✓ 1s   completed at 11:01 PM

Fig 13: Describing the data and its property



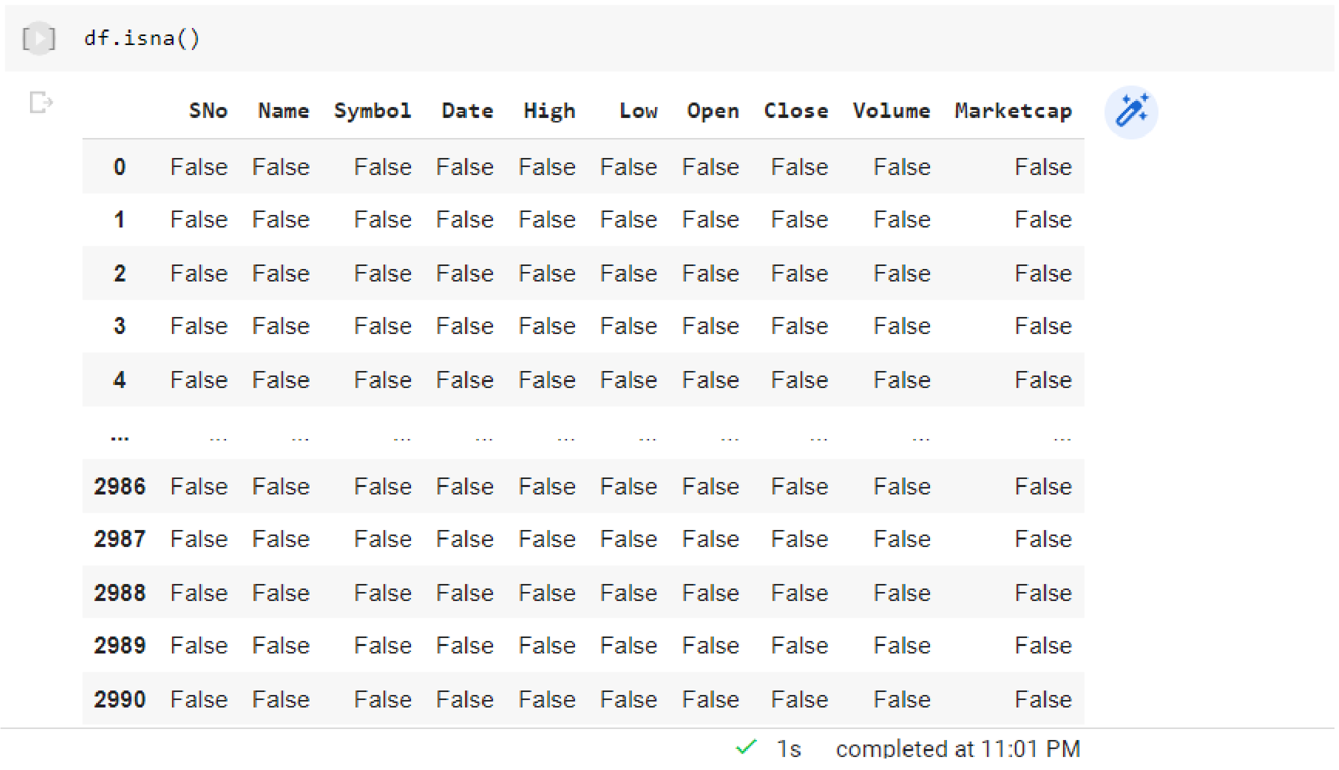
```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2991 entries, 0 to 2990
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SNo        2991 non-null   int64
 1   Name       2991 non-null   object
 2   Symbol     2991 non-null   object
 3   Date       2991 non-null   datetime64[ns]
 4   High       2991 non-null   float64
 5   Low        2991 non-null   float64
 6   Open       2991 non-null   float64
 7   Close      2991 non-null   float64
 8   Volume     2991 non-null   float64
 9   Marketcap  2991 non-null   float64
dtypes: datetime64[ns](1), float64(6), int64(1), object(2)
memory usage: 257.0+ KB
```

Fig 14: looking into the info of data for data type and null values



```
[ ]  df.dtypes

     SNo                    int64
     Name                  object
     Symbol                object
     Date          datetime64[ns]
     High                 float64
     Low                  float64
     Open                 float64
     Close                float64
     Volume               float64
     Marketcap            float64
     dtype: object
```

Fig 15: Looking into the datatype of dataset

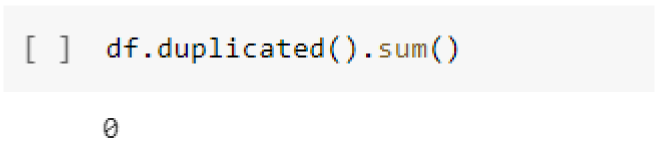


```
[ ]  df.isna()
```

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volume	Marketcap
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
2986	False	False	False	False	False	False	False	False	False	False
2987	False	False	False	False	False	False	False	False	False	False
2988	False	False	False	False	False	False	False	False	False	False
2989	False	False	False	False	False	False	False	False	False	False
2990	False	False	False	False	False	False	False	False	False	False

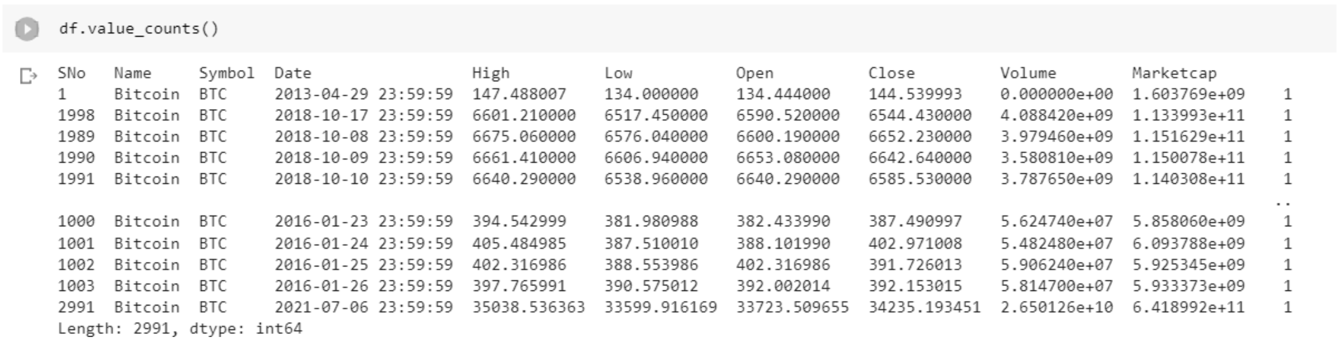
✓ 1s completed at 11:01 PM

Fig 16: Checking for NaN values



```
[ ]  df.duplicated().sum()

     0
```

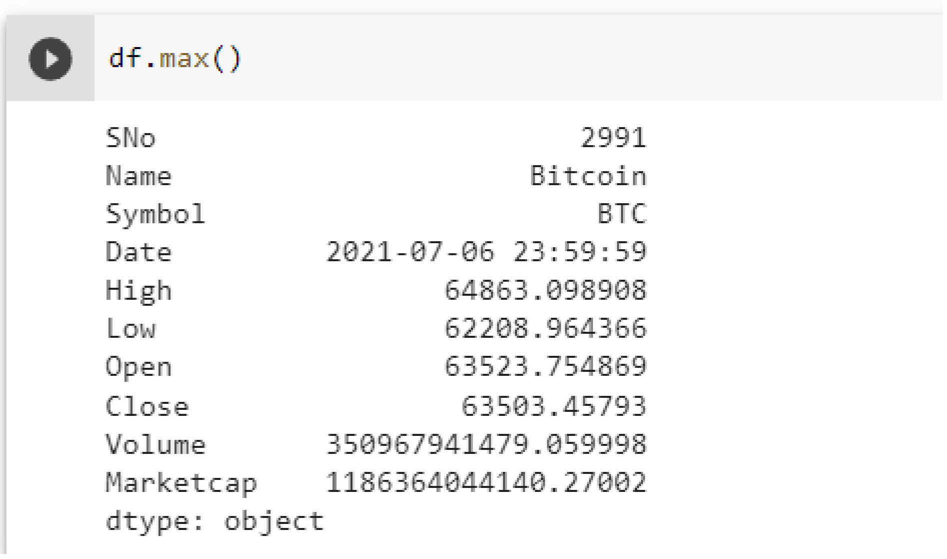
Fig 17: Checking for duplicate values



```
df.value_counts()
```

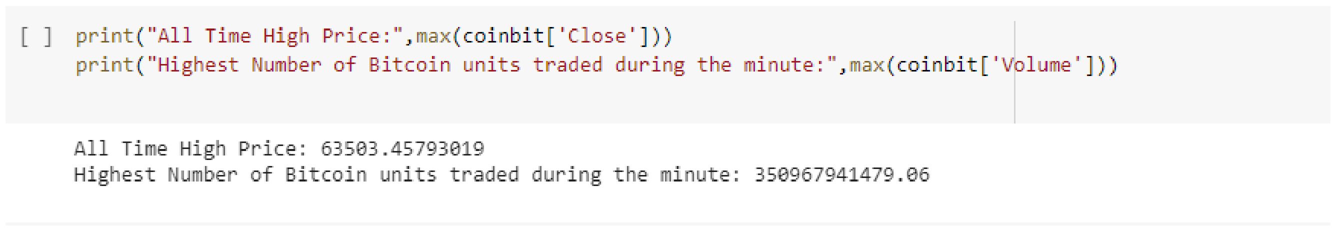
```
SNo   Name     Symbol  Date                 High          Low           Open          Close         Volume        Marketcap
1     Bitcoin  BTC     2013-04-29 23:59:59  147.488007    134.000000    134.444000    144.539993    0.000000e+00  1.603769e+09    1
1998  Bitcoin  BTC     2018-10-17 23:59:59  6601.210000   6517.450000   6590.520000   6544.430000   4.088420e+09  1.133993e+11    1
1989  Bitcoin  BTC     2018-10-08 23:59:59  6675.060000   6576.040000   6600.190000   6652.230000   3.979460e+09  1.151629e+11    1
1990  Bitcoin  BTC     2018-10-09 23:59:59  6661.410000   6606.940000   6653.080000   6642.640000   3.580810e+09  1.150078e+11    1
1991  Bitcoin  BTC     2018-10-10 23:59:59  6640.290000   6538.960000   6640.290000   6585.530000   3.787650e+09  1.140308e+11    1
                                                                                                                               ..
1000  Bitcoin  BTC     2016-01-23 23:59:59  394.542999    381.980988    382.433990    387.490997    5.624740e+07  5.858060e+09    1
1001  Bitcoin  BTC     2016-01-24 23:59:59  405.484985    387.510010    388.101990    402.971008    5.482480e+07  6.093788e+09    1
1002  Bitcoin  BTC     2016-01-25 23:59:59  402.316986    388.553986    402.316986    391.726013    5.906240e+07  5.925345e+09    1
1003  Bitcoin  BTC     2016-01-26 23:59:59  397.765991    390.575012    392.002014    392.153015    5.814700e+07  5.933373e+09    1
2991  Bitcoin  BTC     2021-07-06 23:59:59  35038.536363  33599.916169  33723.509655  34235.193451  2.650126e+10  6.418992e+11    1
Length: 2991, dtype: int64
```

Fig18: Counting for the entire data



```
df.max()
```

```
SNo                           2991
Name                       Bitcoin
Symbol                         BTC
Date           2021-07-06 23:59:59
High                  64863.098908
Low                   62208.964366
Open                  63523.754869
Close                  63503.45793
Volume          350967941479.059998
Marketcap      1186364044140.27002
dtype: object
```

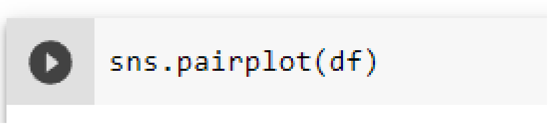


```
print("All Time High Price:",max(coinbit['Close']))
print("Highest Number of Bitcoin units traded during the minute:",max(coinbit['Volume']))
```

```
All Time High Price: 63503.45793019
Highest Number of Bitcoin units traded during the minute: 350967941479.06
```

Fig 19: looking for maximum values for each row

8.1.2 Visualization analysis



```
sns.pairplot(df)
```

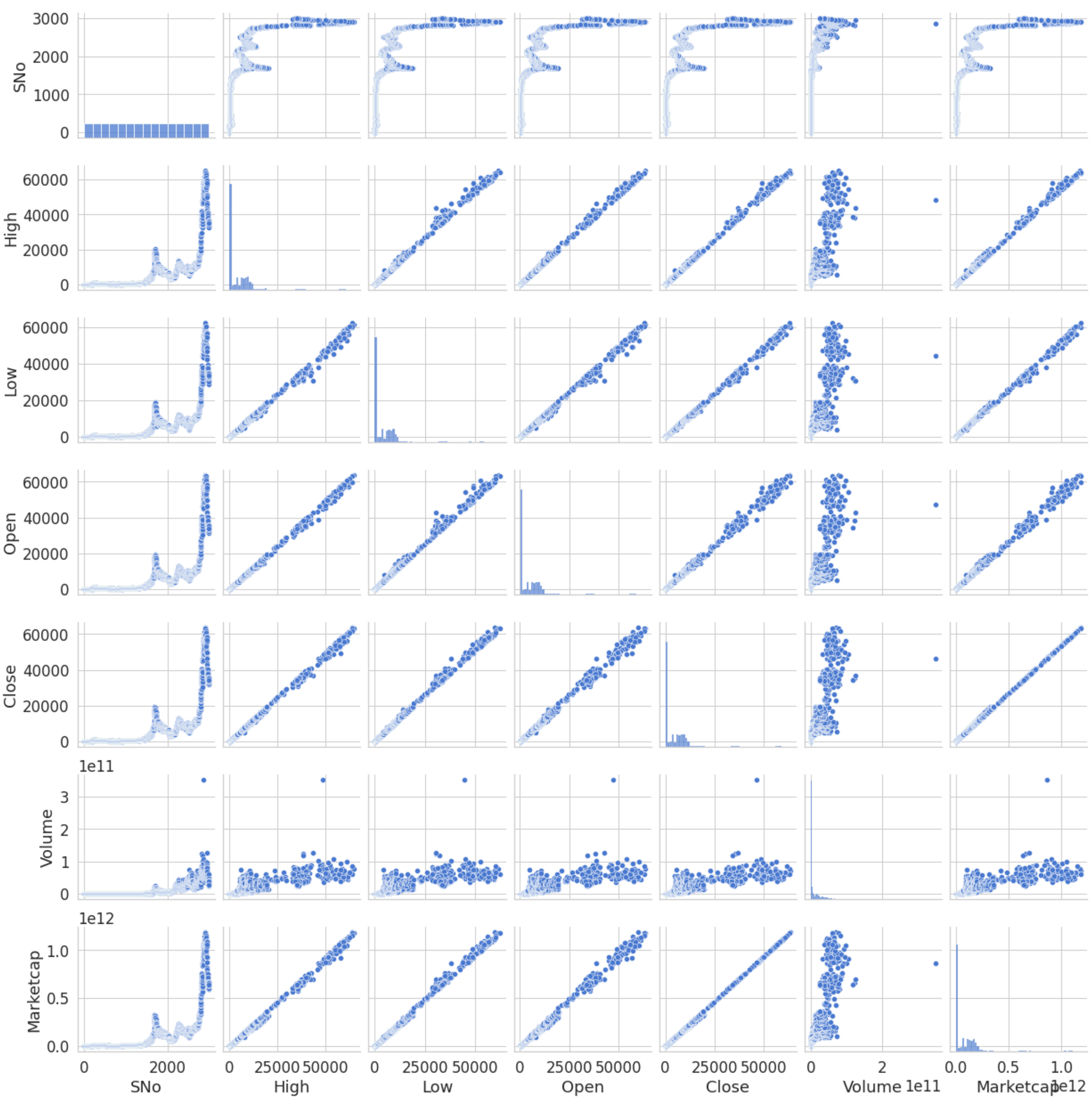


Fig 20 : Pair plot of each and every row

Since the data is for 17 currencies over a wide period of time, looking at the skew line plot of the data was notm eaningful as there were large variations for the same features across different currencies.

We would like to see how the closing values of all seventeen currencies correlate with each other. First we deci de to use pairplots to visualize this. The pairplots for Bitcoin show positive linear relationships for a number of features, such as *Open* and *Close*.

```
sns.boxplot(x=df["Close"])
```

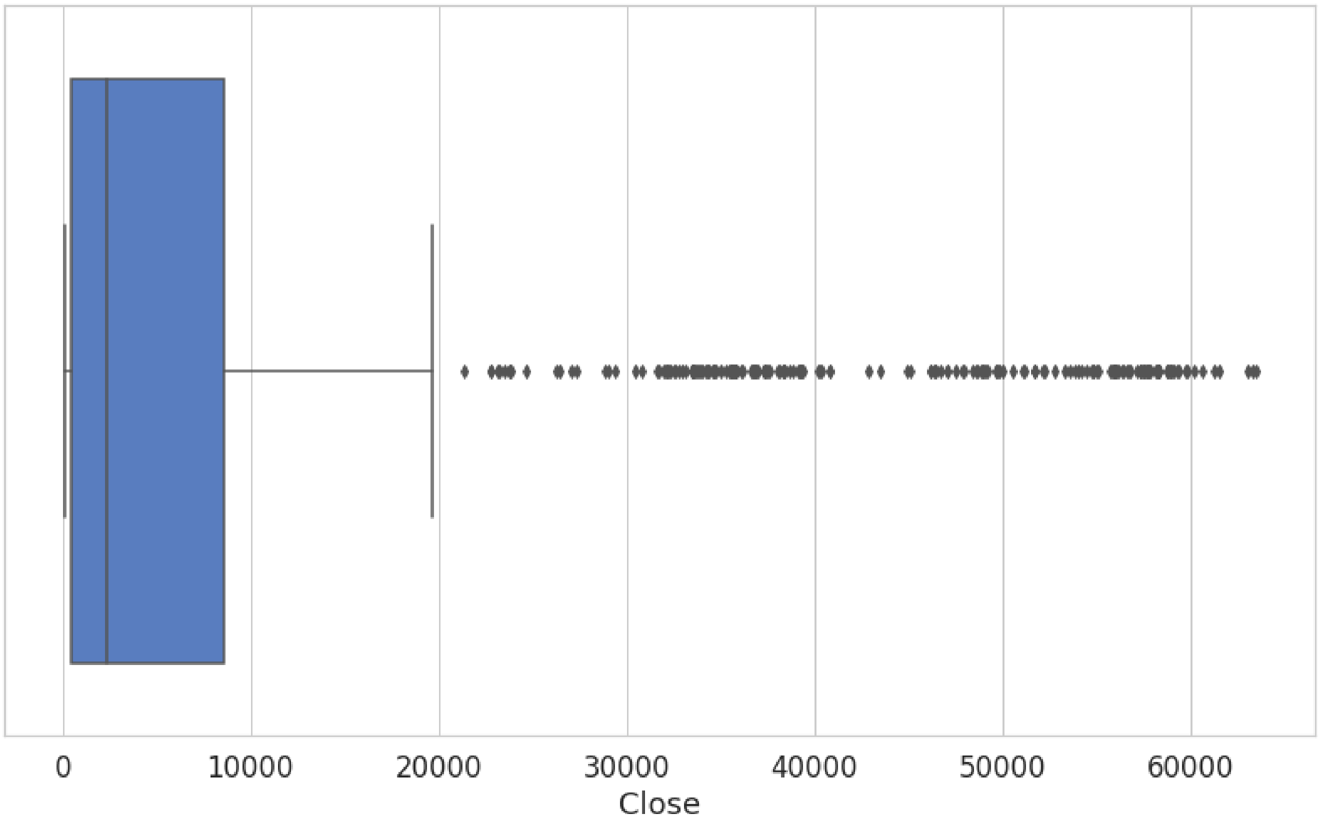


Fig 21: Boxplot for Close price

Draw a single horizontal boxplot, assigning the data directly to the coordinate variable:

```
[ ]  boxplot = df.boxplot(column=['Low','High','Open','Close'])
```

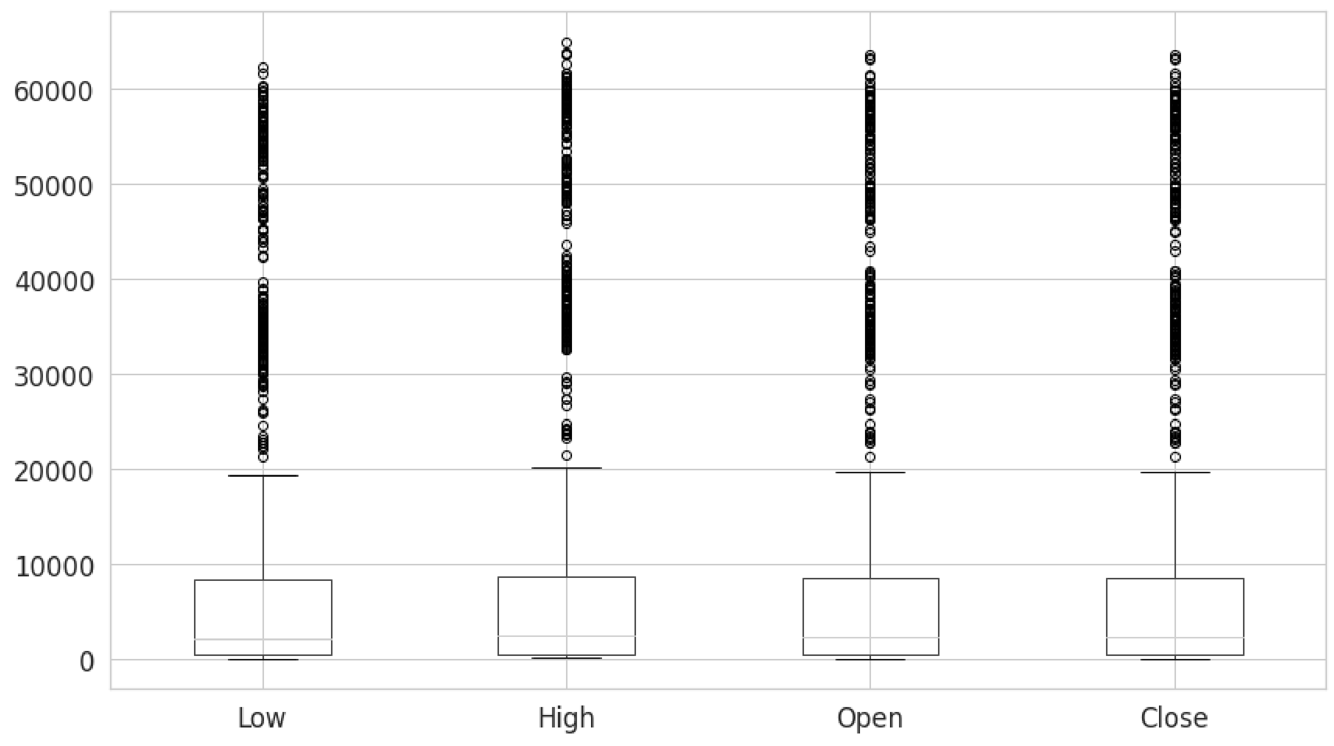


Fig 22: Boxplot for Low, High, Open and Close combined

Group by a categorical variable, referencing columns in a dataframe:

```
sns.boxplot(data=df[["Open", "Close" ,"High" ,"Low"]], orient="h")
```

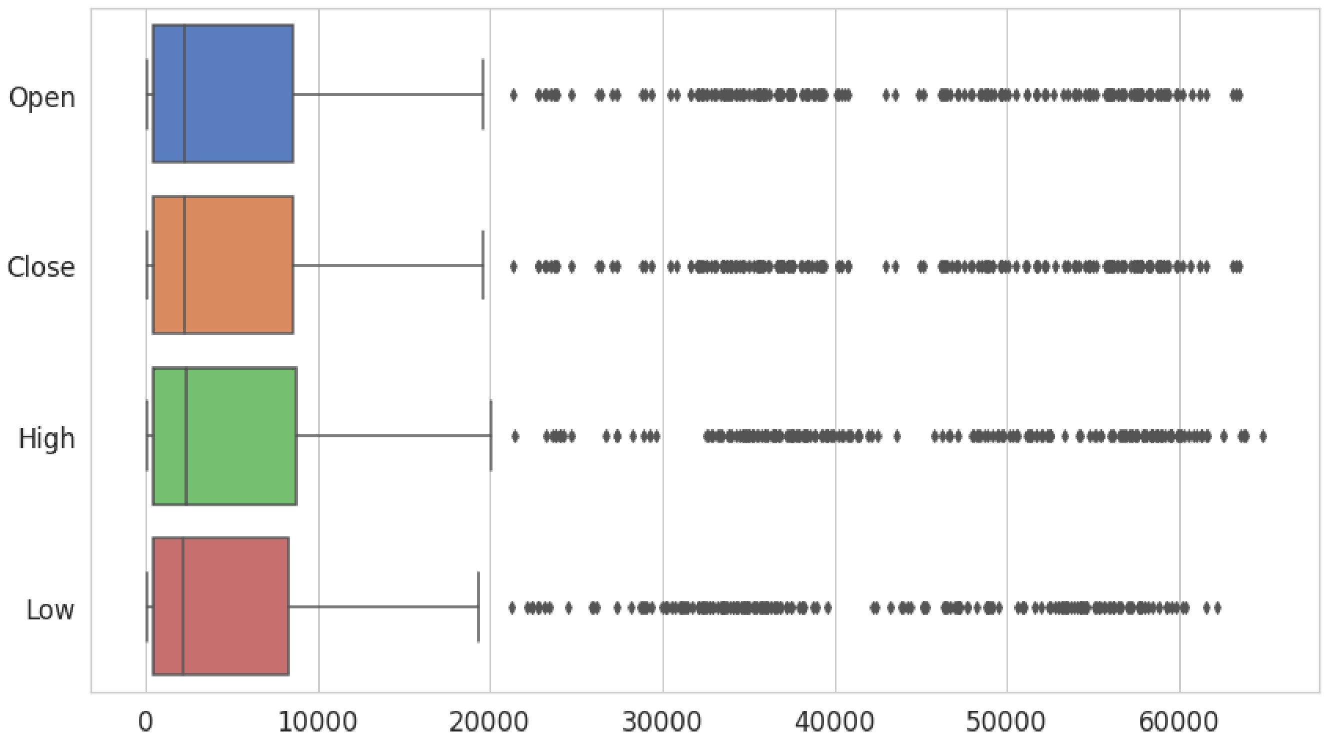
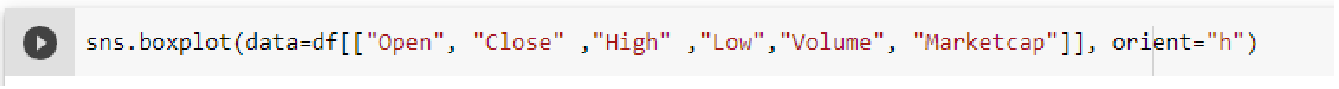


Fig 23: Boxplot for Low, High, Open and Close combined in horizontal orientation

Group by a categorical variable, referencing columns in a dataframe:



```
sns.boxplot(data=df[["Open", "Close" ,"High" ,"Low","Volume", "Marketcap"]], orient="h")
```

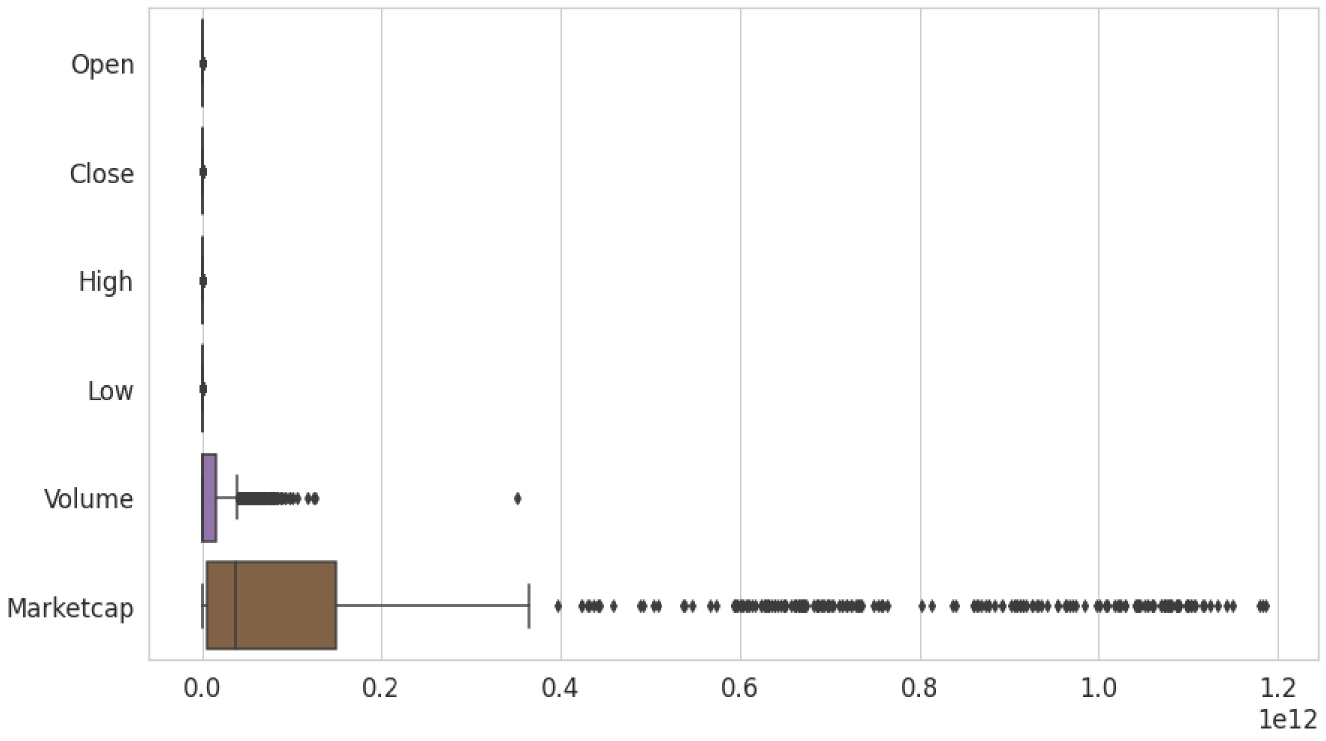
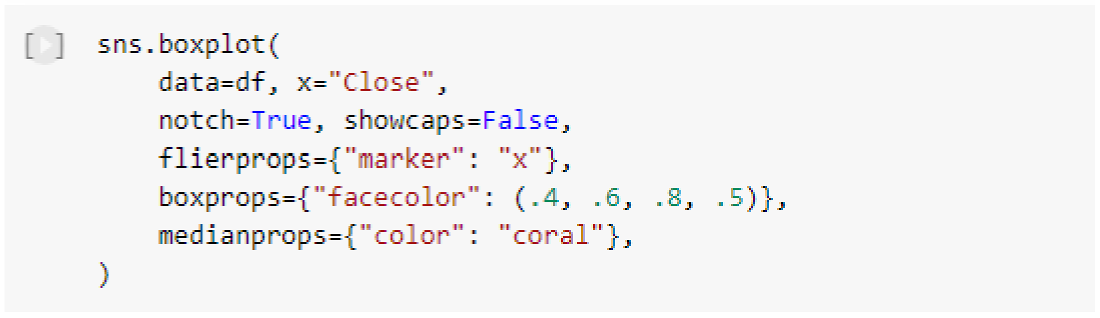


Fig 24: Boxplot for Low, High, Open , Close, Volume and Market Capitalization combined

Group by a categorical variable, referencing columns in a dataframe:



```
sns.boxplot(
    data=df, x="Close",
    notch=True, showcaps=False,
    flierprops={"marker": "x"},
    boxprops={"facecolor": (.4, .6, .8, .5)},
    medianprops={"color": "coral"},
)
```

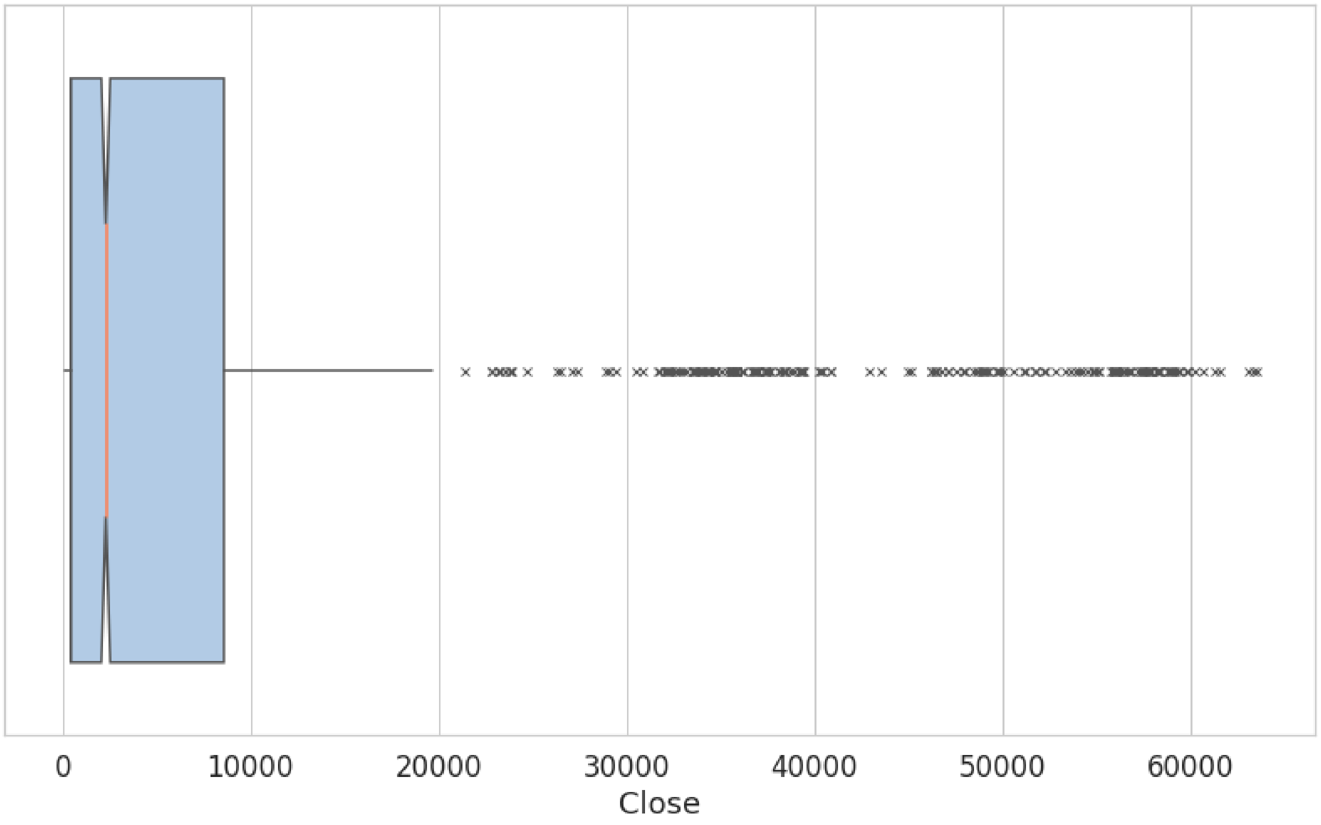
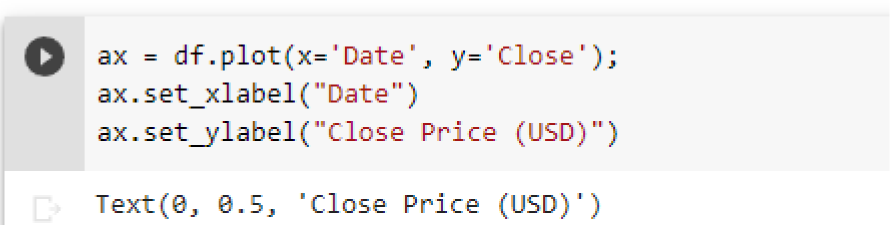


Fig 25: Boxplot Close value with additional information

Pass additional keyword arguments to matplotlib:



```
ax = df.plot(x='Date', y='Close');
ax.set_xlabel("Date")
ax.set_ylabel("Close Price (USD)")
```

```
Text(0, 0.5, 'Close Price (USD)')
```

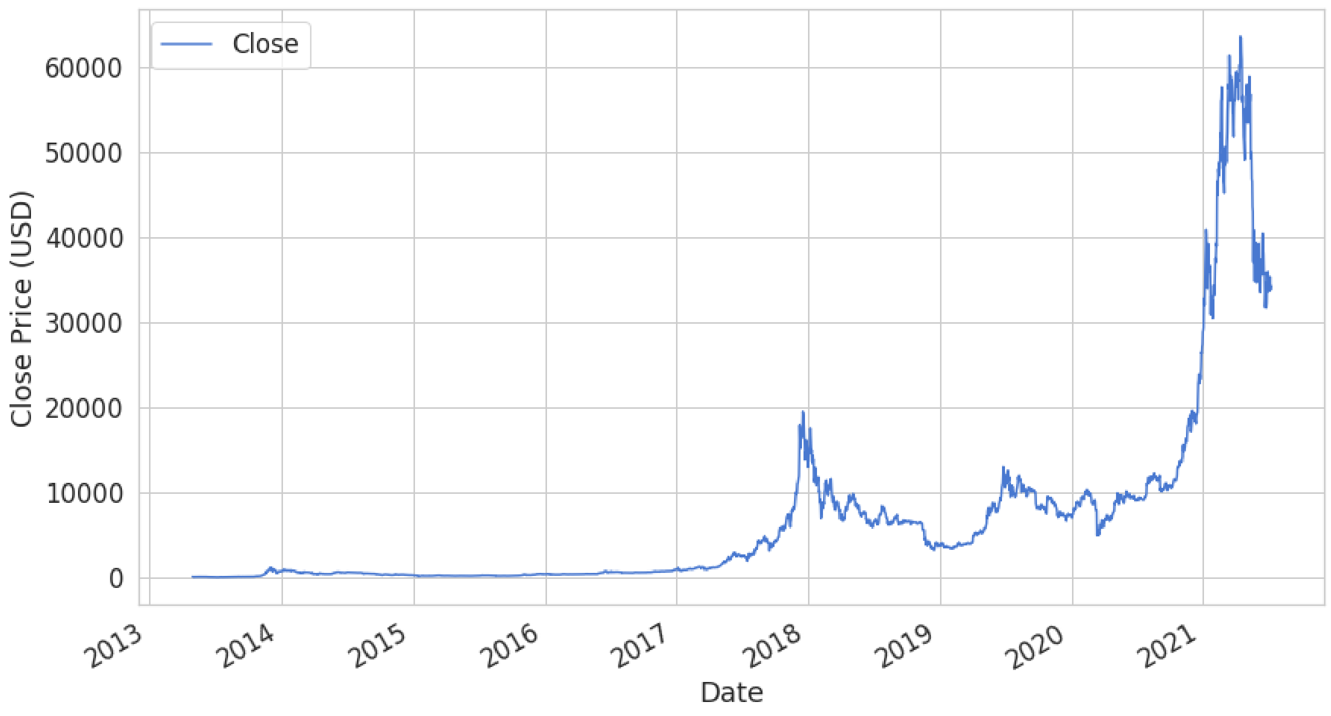
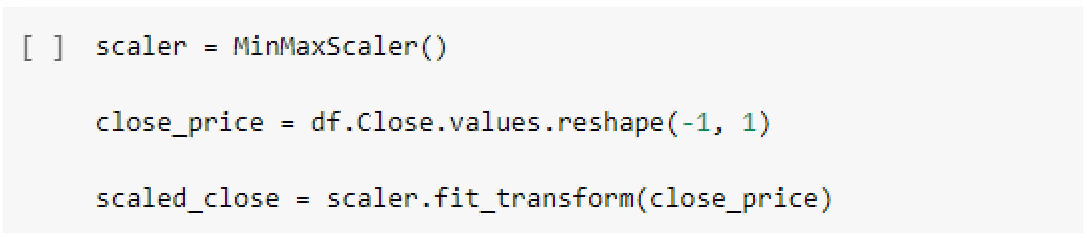


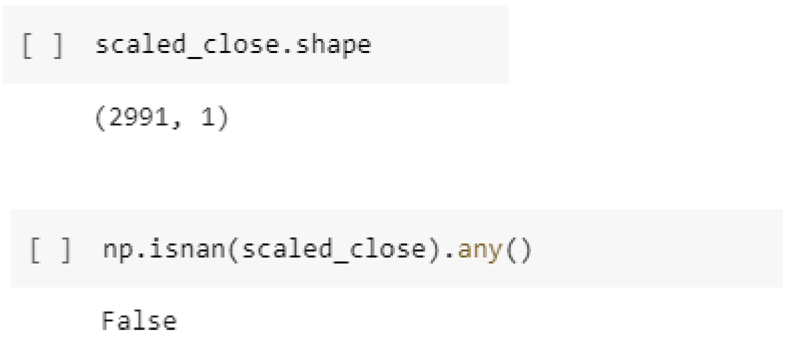
Fig 26: Plotting Close value (As training and testing data)

9.Normalization



```
[ ]  scaler = MinMaxScaler()

     close_price = df.Close.values.reshape(-1, 1)

     scaled_close = scaler.fit_transform(close_price)
```

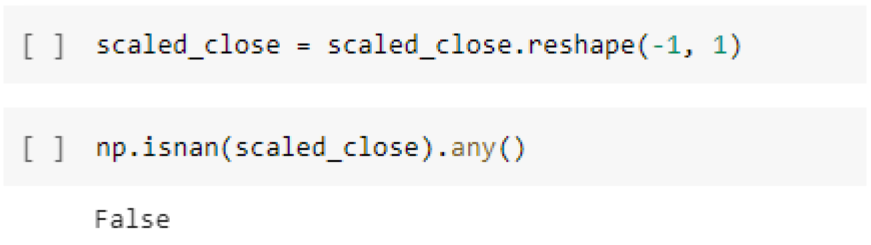
Fig 27: Min Max scaler for reshaping the data



```
[ ]  scaled_close.shape

     (2991, 1)
```

```
[ ]  np.isnan(scaled_close).any()

     False
```

Fig 28: scaling and additional filter for NaN values



```
[ ]  scaled_close = scaled_close.reshape(-1, 1)
```

```
[ ]  np.isnan(scaled_close).any()

     False
```

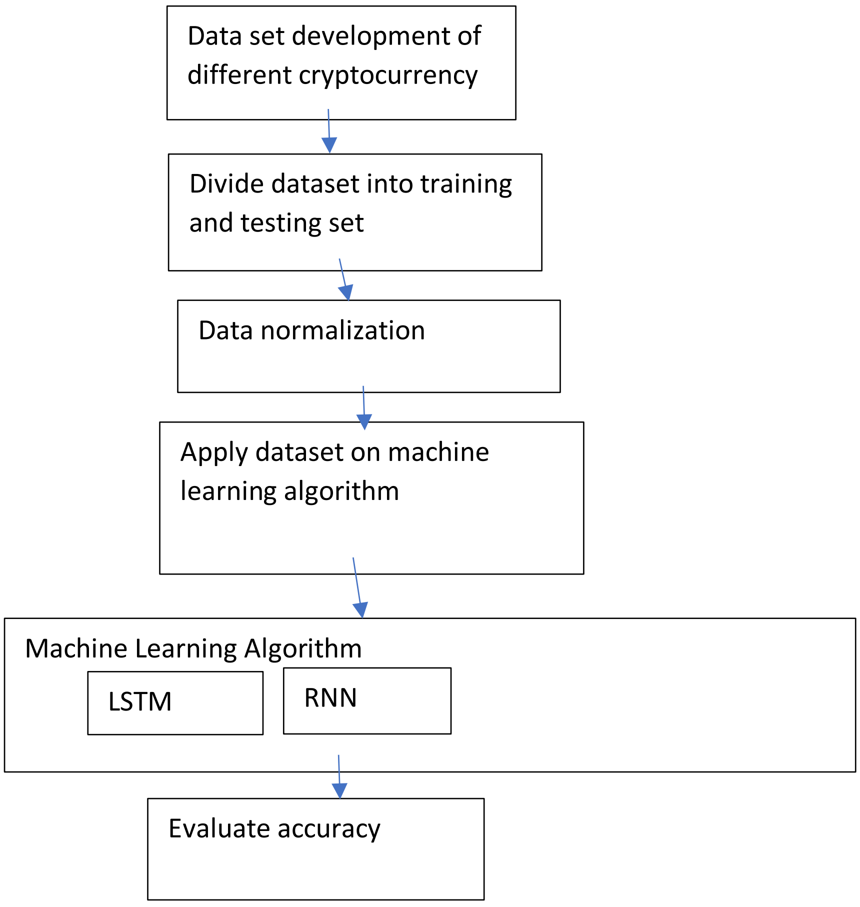
Fig 29: scaling and additional filter for NaN values

Continue by normalizing the values to floats between -
1 to 1. We are using sklearn's MinMaxScaler. We need to be careful to fit the scaler on our entire data range or else we will end up with a messed up scale between our test data and our train data. After we fit it we then trans form both our test and training data.

10.ML (Machine Learning)-Qualitative

Trained distinct models for different forms of Bitcoins price prediction using historical cryptocurrency price s. After data collection and preliminary processing following task will be performed.

- Data exploration and visualization.
- Training the models.
- Testing the models.
- Make Prediction
- Extracting and comparing the results.


Data set development of different cryptocurrency
Divide dataset into training and testing set
Data normalization
Apply dataset on machine learning algorithm
Machine Learning Algorithm
LSTM
RNN
Evaluate accuracy

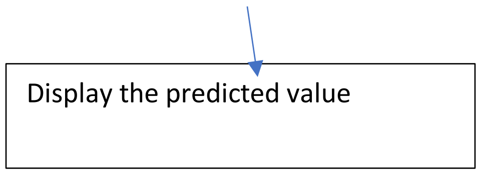


Fig 30: ML deployment process

11.Core Solution

It is said that "Human mind can only be able to do visualization" Our mind is not for any sorts of theoretical letter and descriptive data. Performing the Data Preprocessing like importing the required libraries and loading them, importing the csv files to analyze the data frames, Handling the missing data, Visualizing the data, finding the relationship between the variables, splitting it into train and test data for a model. Every parameter should properly clean so we follow theprocess of data cleaning present it into the best format and visualize it and apply the regression more specifically linear regression so see which method suits our data sets. Different timeseries methods such as the Exponential Weighted Averages, Generalized Autoregressive Conditional Heteroskedastic Model, Structural Time Series models were used to observe the future of coins. Various uncertainty quantification methods such as Mean Absolute Error and Root Mean Square Error were used to determine the accuracy of the predictions(Seaborn.boxplot — Seaborn 0.11.1 Documentation, n.d.)

.

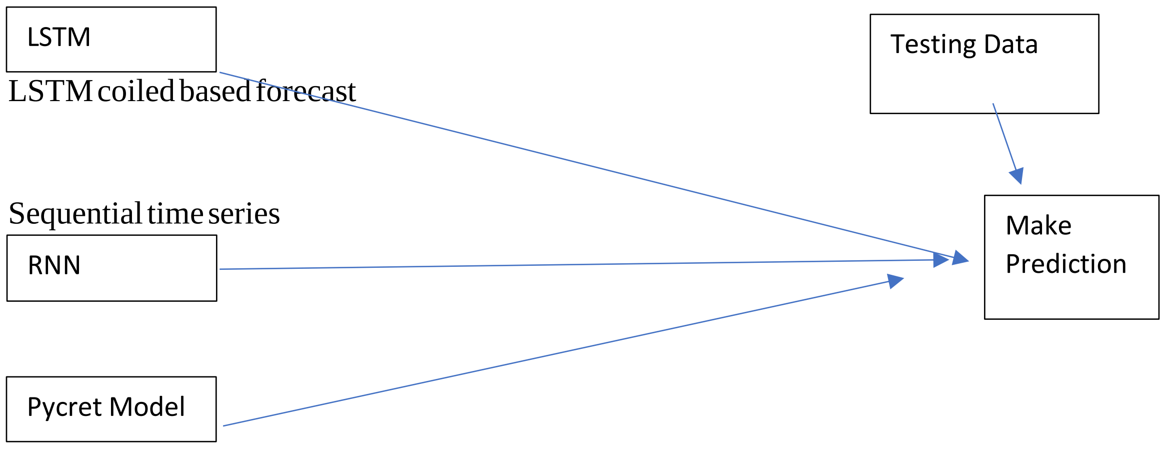
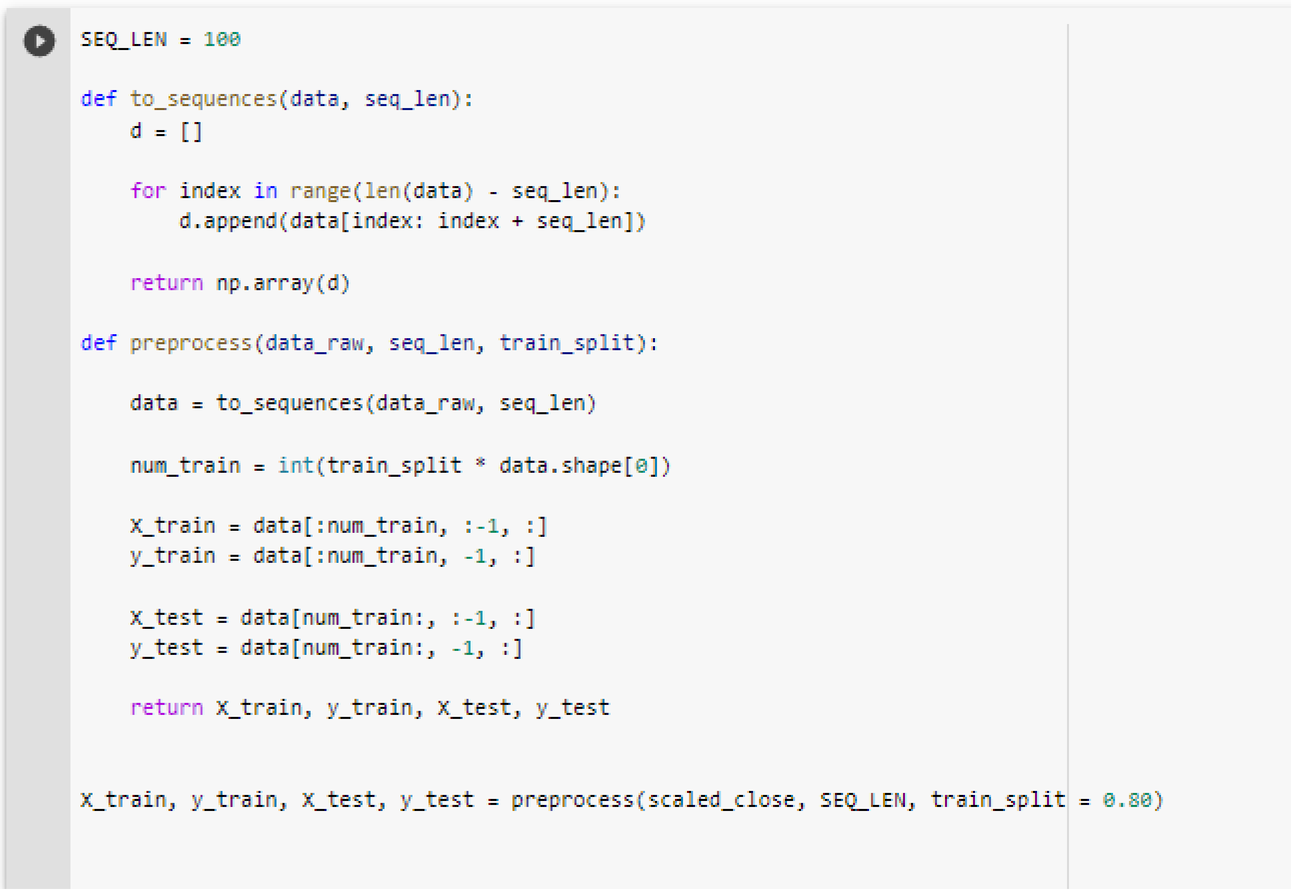


Fig 31: Forecasting Process

12.Data Pre-processing for Training and Testing

12.1 For LSTM Model



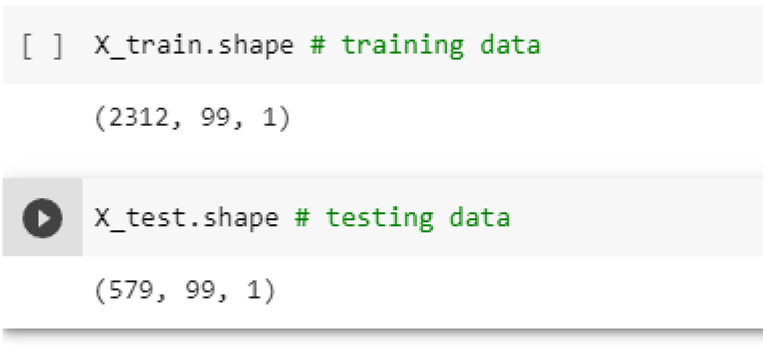
```
SEQ_LEN = 100

def to_sequences(data, seq_len):
    d = []

    for index in range(len(data) - seq_len):
        d.append(data[index: index + seq_len])

    return np.array(d)

def preprocess(data_raw, seq_len, train_split):

    data = to_sequences(data_raw, seq_len)

    num_train = int(train_split * data.shape[0])

    X_train = data[:num_train, :-1, :]
    y_train = data[:num_train, -1, :]

    X_test = data[num_train:, :-1, :]
    y_test = data[num_train:, -1, :]

    return X_train, y_train, X_test, y_test


X_train, y_train, X_test, y_test = preprocess(scaled_close, SEQ_LEN, train_split = 0.80)
```

Fig 32: data pre-processing for training and testing

Now we seperate our training data into our inputs and our outputs in time steps of time_steps. Where we will l ook at time_steps amount of data before we make our prediction of what the output for y will be.

Splitting our training data into training and validation.

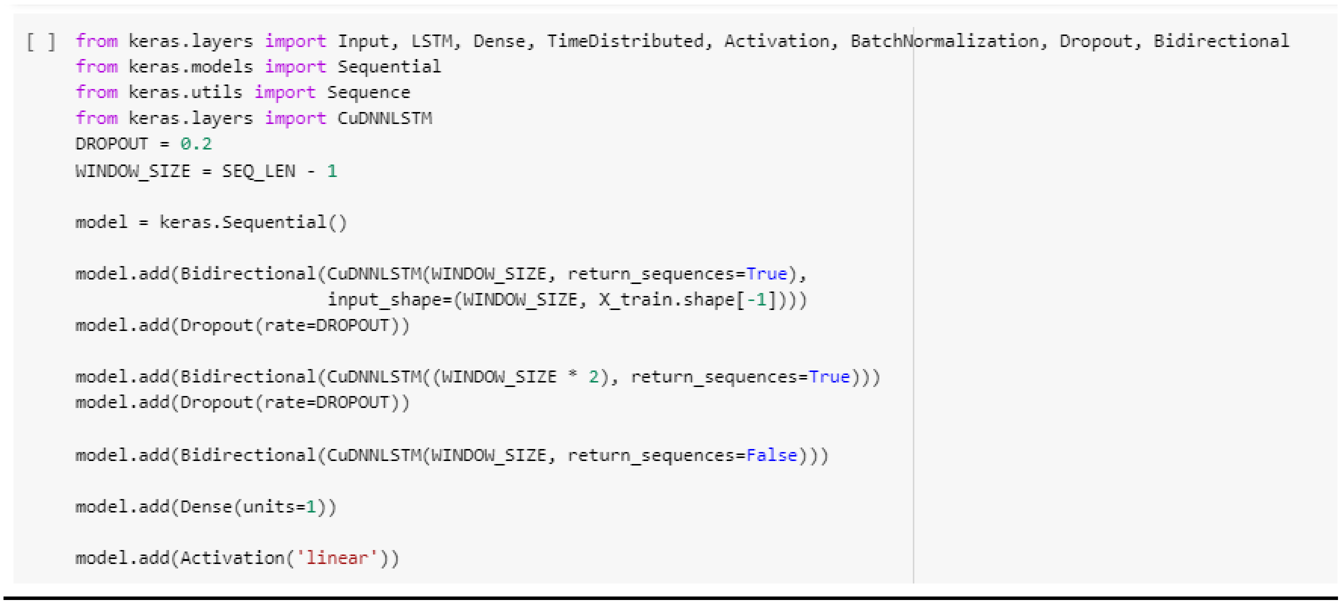
Now we implement our actual model. We start with an LSTM input layer with 100 hidden units. We add a dro pout of 0.2 before our Dense output layer with a linear activation and a shape of 1 (as we are outputting our ex pected price). We are using mean squared error to calculate our loss and adam as our optimizer.



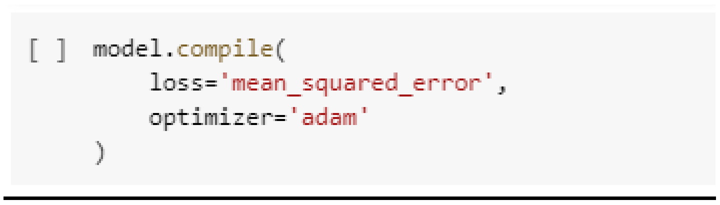
```
[ ] X_train.shape # training data

    (2312, 99, 1)

    X_test.shape # testing data

    (579, 99, 1)
```

Fig 33: training and testing data set

12.2 Model Building



```
[ ] from keras.layers import Input, LSTM, Dense, TimeDistributed, Activation, BatchNormalization, Dropout, Bidirectional
    from keras.models import Sequential
    from keras.utils import Sequence
    from keras.layers import CuDNNLSTM
    DROPOUT = 0.2
    WINDOW_SIZE = SEQ_LEN - 1

    model = keras.Sequential()

    model.add(Bidirectional(CuDNNLSTM(WINDOW_SIZE, return_sequences=True),
                            input_shape=(WINDOW_SIZE, X_train.shape[-1])))
    model.add(Dropout(rate=DROPOUT))

    model.add(Bidirectional(CuDNNLSTM((WINDOW_SIZE * 2), return_sequences=True)))
    model.add(Dropout(rate=DROPOUT))

    model.add(Bidirectional(CuDNNLSTM(WINDOW_SIZE, return_sequences=False)))

    model.add(Dense(units=1))

    model.add(Activation('linear'))
```



```
[ ] model.compile(
        loss='mean_squared_error',
        optimizer='adam'
    )
```

Fig 34: Model Building with necessary layers.

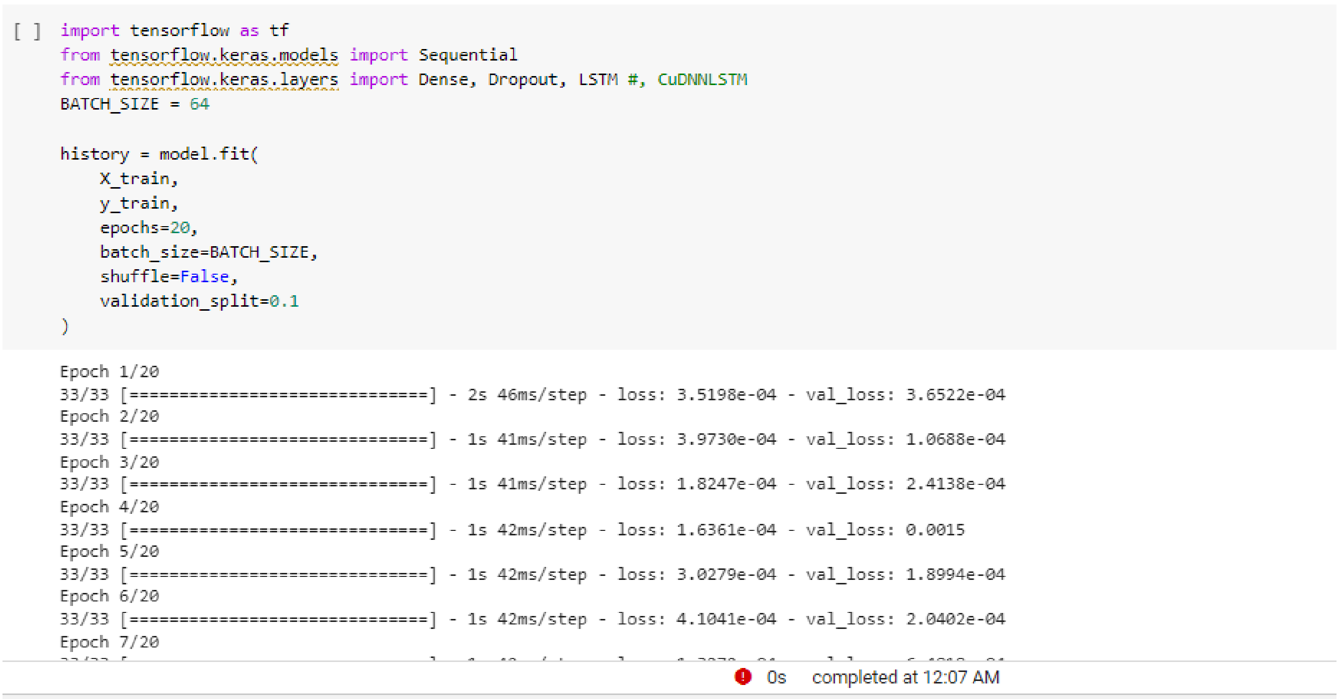
All models I have built so far do not allow for operating on sequence data. Fortunately, I have use a special clas s of Neural Network models known as **Recurrent Neural Networks (RNNs)** just for this purpose. RNNs allo w using the output from the model as a new input for the same model. The process can be repeated indefinitely.

One serious limitation of RNNs is the inability of capturing long-term dependencies in a sequence One way to handle the situation is by using an **Long short-term memory (LSTM)** variant of RNN.The default LSTM behavior is remembering information for prolon ged periods of time. Let's see how to use LSTM in Keras.

Bidirectional RNNs allows you to train on the sequence data in forward and backward (reversed) direction. In practice, this approach works well with LSTMs.

CuDNNLSTM is a “Fast LSTM implementation backed by cuDNN”. Personally, I think it is a good example of leaky abstraction, but it is crazy fast!

Our output layer has a single neuron (predicted Bitcoin price). We use Linear activation function which activa tion is proportional to the input.



```
[ ] import tensorflow as tf
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Dropout, LSTM #, CuDNNLSTM
    BATCH_SIZE = 64

    history = model.fit(
        X_train,
        y_train,
        epochs=20,
        batch_size=BATCH_SIZE,
        shuffle=False,
        validation_split=0.1
    )

    Epoch 1/20
    33/33 [==============================] - 2s 46ms/step - loss: 3.5198e-04 - val_loss: 3.6522e-04
    Epoch 2/20
    33/33 [==============================] - 1s 41ms/step - loss: 3.9730e-04 - val_loss: 1.0688e-04
    Epoch 3/20
    33/33 [==============================] - 1s 41ms/step - loss: 1.8247e-04 - val_loss: 2.4138e-04
    Epoch 4/20
    33/33 [==============================] - 1s 42ms/step - loss: 1.6361e-04 - val_loss: 0.0015
    Epoch 5/20
    33/33 [==============================] - 1s 42ms/step - loss: 3.0279e-04 - val_loss: 1.8994e-04
    Epoch 6/20
    33/33 [==============================] - 1s 42ms/step - loss: 4.1041e-04 - val_loss: 2.0402e-04
    Epoch 7/20
```

0s completed at 12:07 AM

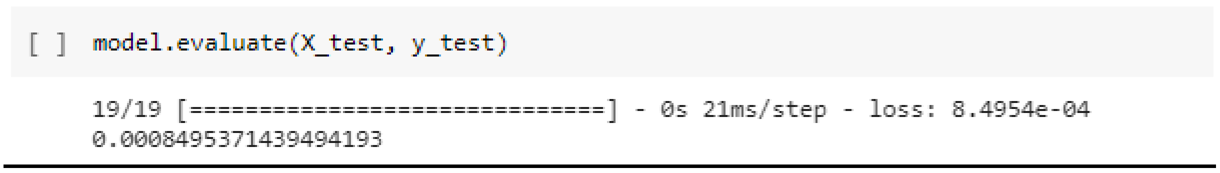
Fig35: Training and model fitting

Fit our model now on our X and y training/validation data. We take advantage of keras' early stopping class s o that once we are no longer recieving improvements the model will take its best weights and stop.

Now we are breaking our testing data up into time steps and splitting it again into our inputs and our expected outputs.

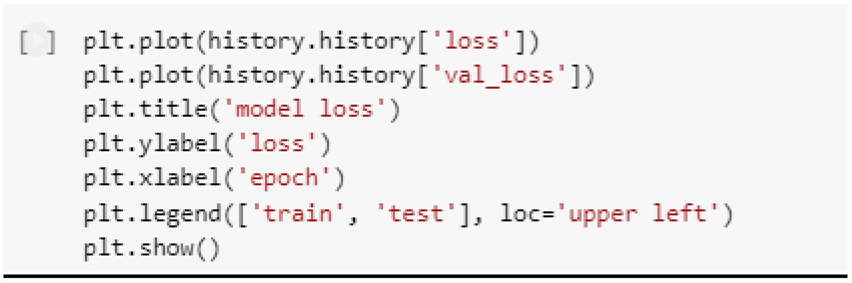
We then predict on the inputs and then scale both the inputs and outputs back up, now were ready to see how w e did!

12.3Model evaluation



```
[ ]  model.evaluate(X_test, y_test)

     19/19 [==============================] - 0s 21ms/step - loss: 8.4954e-04
     0.0008495371439494193
```

Fig 36: Evaluating model



```
[ ]  plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('model loss')
     plt.ylabel('loss')
     plt.xlabel('epoch')
     plt.legend(['train', 'test'], loc='upper left')
     plt.show()
```

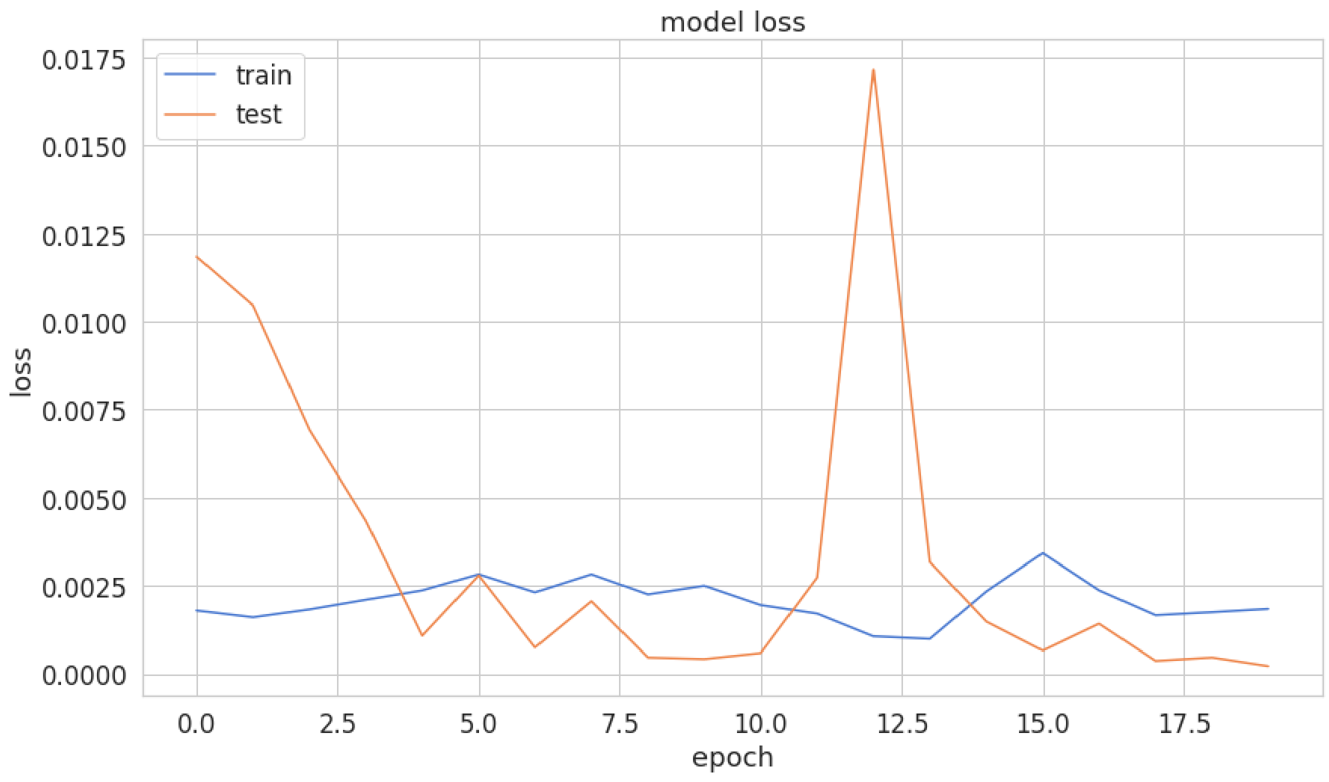
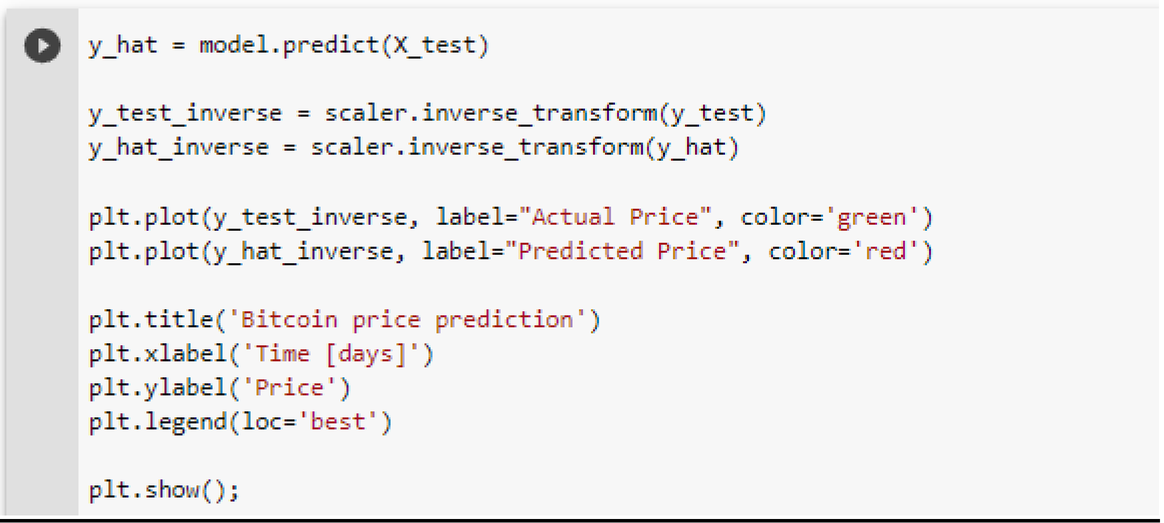


Fig 37: Plotting training and testing data



```
y_hat = model.predict(X_test)

y_test_inverse = scaler.inverse_transform(y_test)
y_hat_inverse = scaler.inverse_transform(y_hat)

plt.plot(y_test_inverse, label="Actual Price", color='green')
plt.plot(y_hat_inverse, label="Predicted Price", color='red')

plt.title('Bitcoin price prediction')
plt.xlabel('Time [days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.show();
```

Fig 38: Prediction for LSTM Model

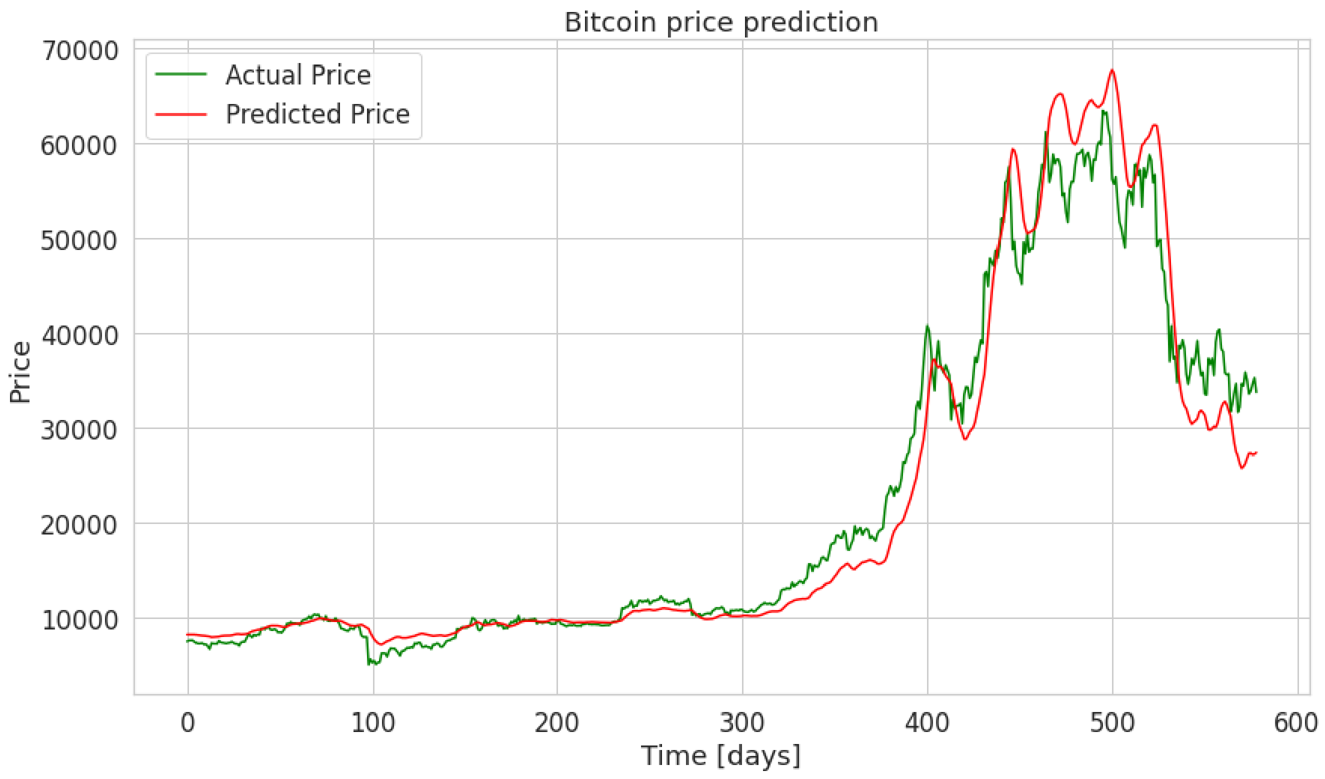
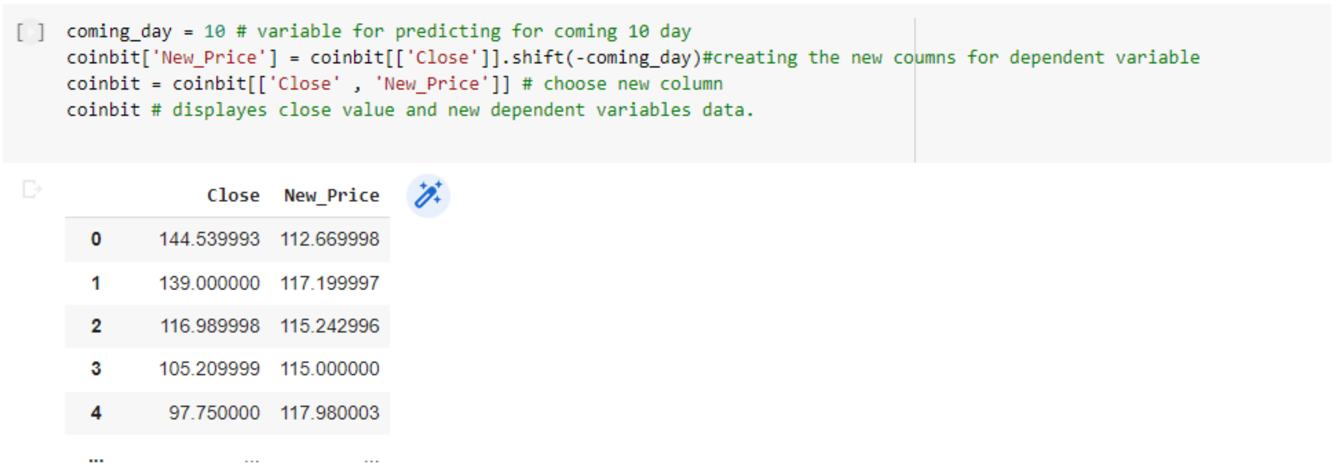


Fig 39: Predicted price visualization along with close price for LSTM model

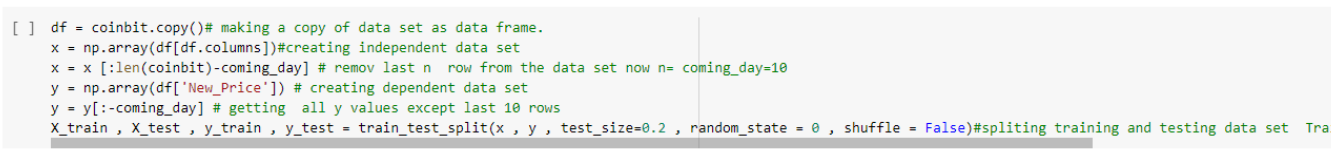
12.2 For Pycret Model



```
coming_day = 10 # variable for predicting for coming 10 day
coinbit['New_Price'] = coinbit[['Close']].shift(-coming_day)#creating the new coumns for dependent variable
coinbit = coinbit[['Close' , 'New_Price']] # choose new column
coinbit # displayes close value and new dependent variables data.
```

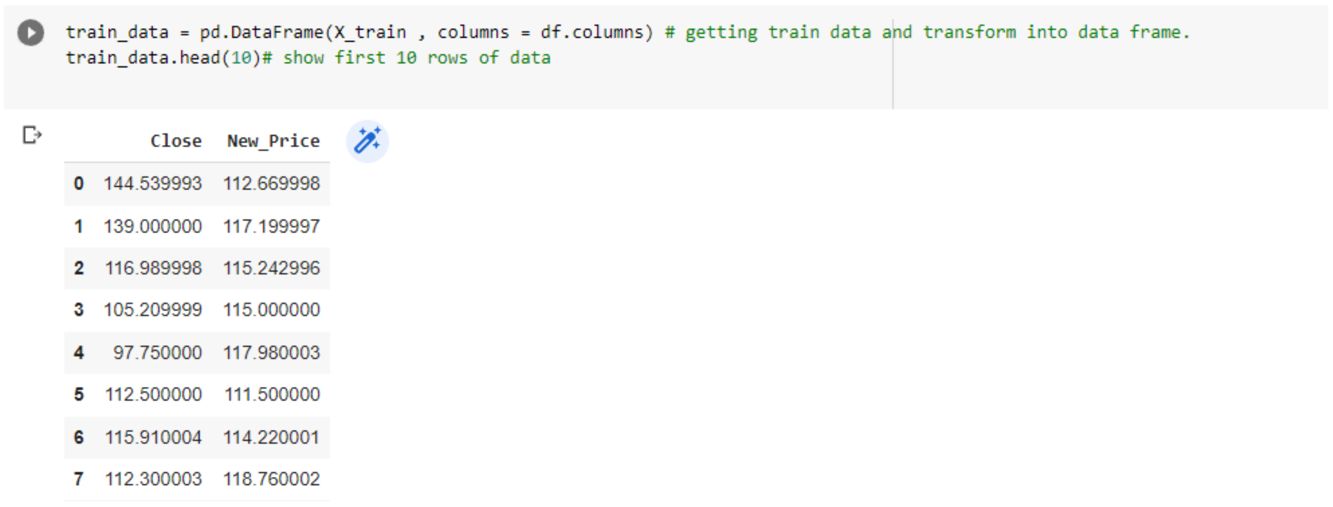
```
        Close   New_Price
0  144.539993  112.669998
1  139.000000  117.199997
2  116.989998  115.242996
3  105.209999  115.000000
4   97.750000  117.980003
...       ...         ...
```

Fig 40: Creating new dependent variable



```
df = coinbit.copy()# making a copy of data set as data frame.
x = np.array(df[df.columns])#creating independent data set
x = x [:len(coinbit)-coming_day] # remov last n  row from the data set now n= coming_day=10
y = np.array(df['New_Price']) # creating dependent data set
y = y[:-coming_day] # getting  all y values except last 10 rows
X_train , X_test , y_train , y_test = train_test_split(x , y , test_size=0.2 , random_state = 0 , shuffle = False)#spliting training and testing data set  Tra
```

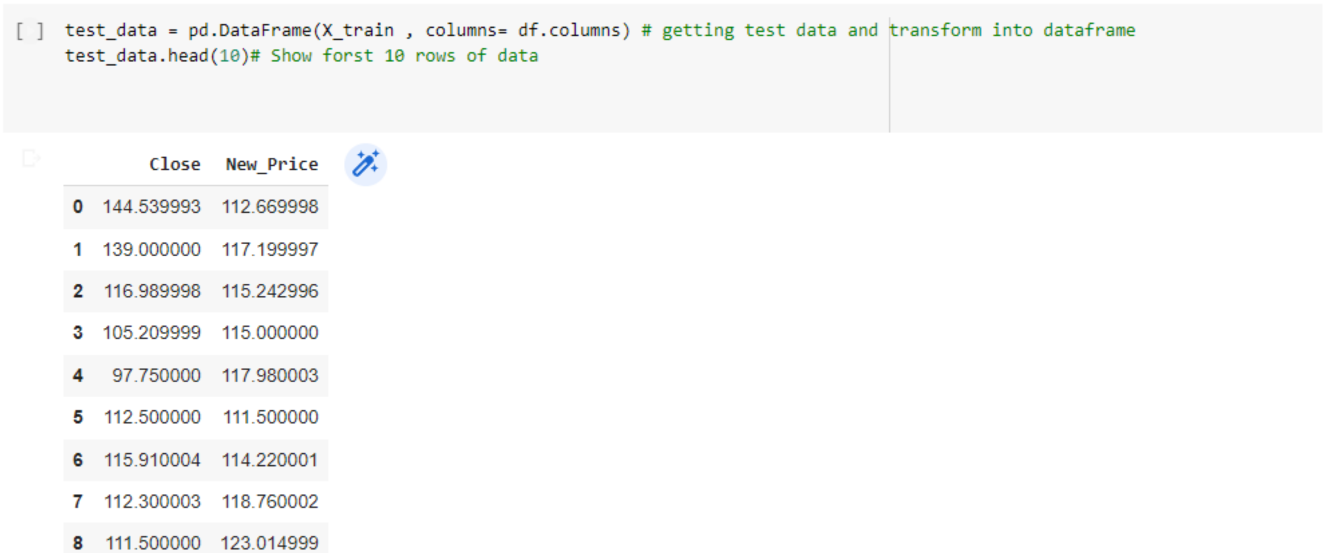
Fig 41: Splitting dataset for training and testing



```
train_data = pd.DataFrame(X_train , columns = df.columns) # getting train data and transform into data frame.
train_data.head(10)# show first 10 rows of data
```

```
        Close   New_Price
0  144.539993  112.669998
1  139.000000  117.199997
2  116.989998  115.242996
3  105.209999  115.000000
4   97.750000  117.980003
5  112.500000  111.500000
6  115.910004  114.220001
7  112.300003  118.760002
```

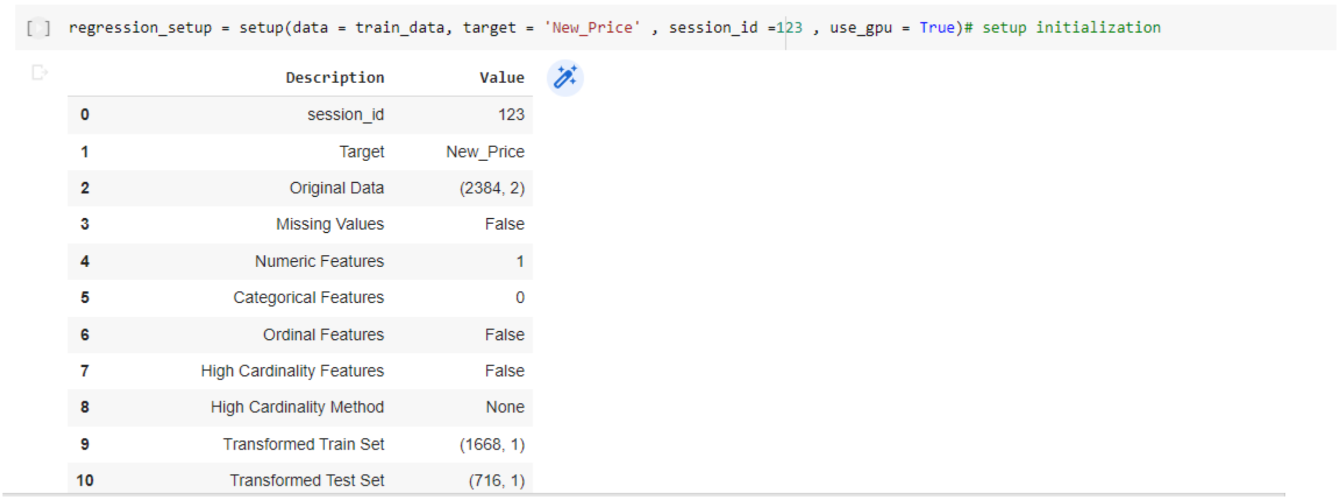
Fig 42: Training data set



```
test_data = pd.DataFrame(X_train , columns= df.columns) # getting test data and transform into dataframe
test_data.head(10)# Show forst 10 rows of data
```

	Close	New_Price
0	144.539993	112.669998
1	139.000000	117.199997
2	116.989998	115.242996
3	105.209999	115.000000
4	97.750000	117.980003
5	112.500000	111.500000
6	115.910004	114.220001
7	112.300003	118.760002
8	111.500000	123.014999

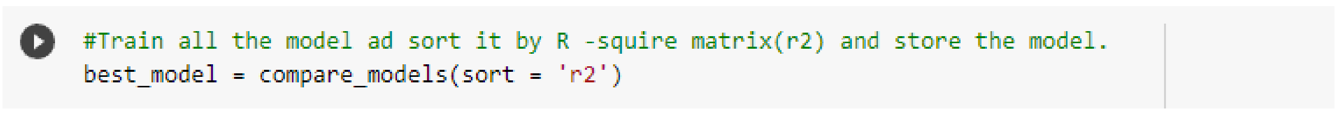
Fig 43:Testing data set



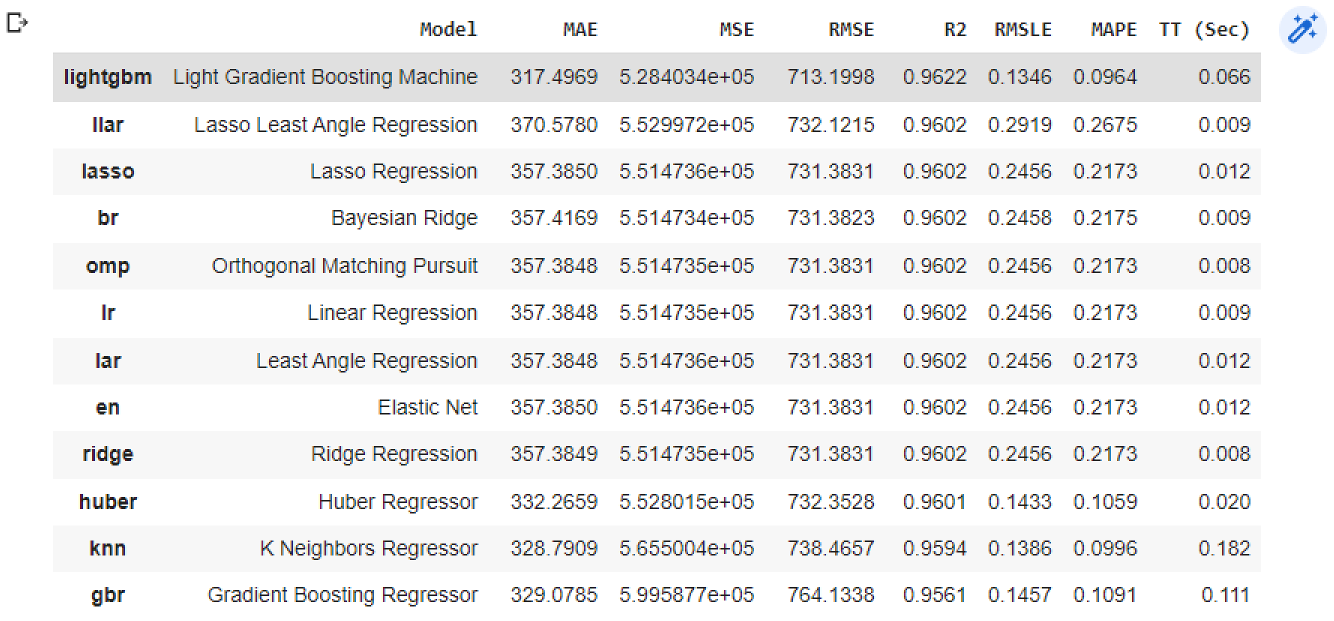
```
regression_setup = setup(data = train_data, target = 'New_Price' , session_id =123 , use_gpu = True)# setup initialization
```

	Description	Value
0	session_id	123
1	Target	New_Price
2	Original Data	(2384, 2)
3	Missing Values	False
4	Numeric Features	1
5	Categorical Features	0
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(1668, 1)
10	Transformed Test Set	(716, 1)

Fig44 : setup initialization

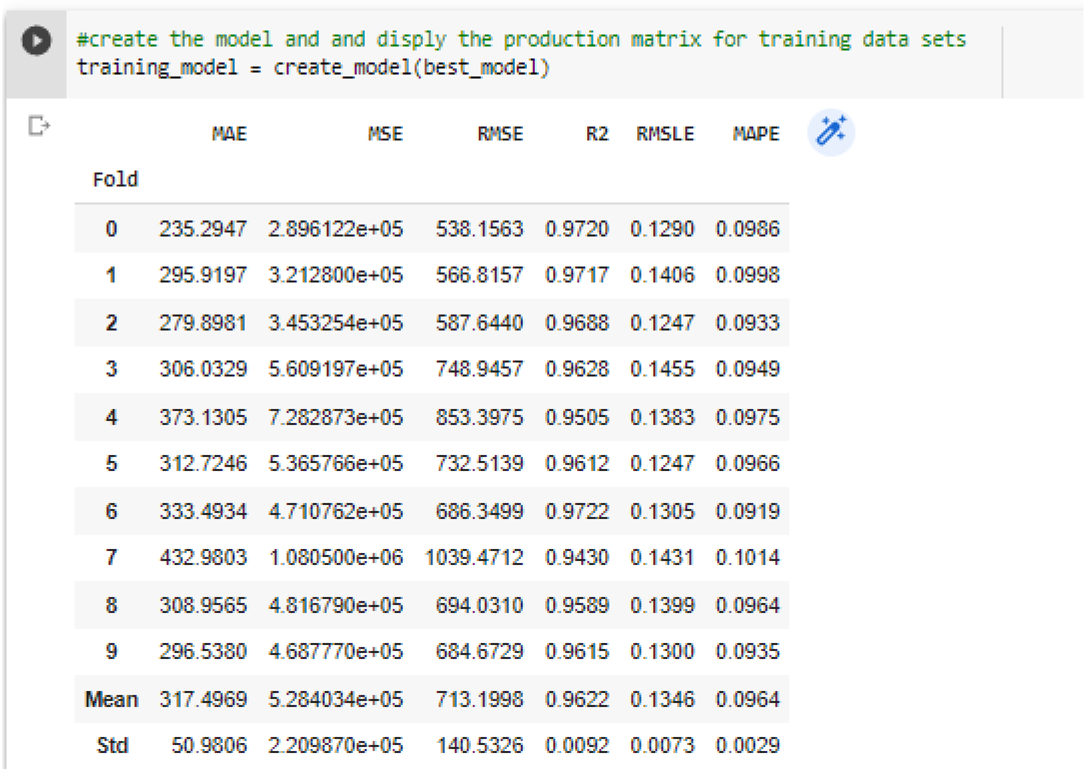


```
#Train all the model ad sort it by R -squire matrix(r2) and store the model.
best_model = compare_models(sort = 'r2')
```



	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lightgbm	Light Gradient Boosting Machine	317.4969	5.284034e+05	713.1998	0.9622	0.1346	0.0964	0.066
llar	Lasso Least Angle Regression	370.5780	5.529972e+05	732.1215	0.9602	0.2919	0.2675	0.009
lasso	Lasso Regression	357.3850	5.514736e+05	731.3831	0.9602	0.2456	0.2173	0.012
br	Bayesian Ridge	357.4169	5.514734e+05	731.3823	0.9602	0.2458	0.2175	0.009
omp	Orthogonal Matching Pursuit	357.3848	5.514735e+05	731.3831	0.9602	0.2456	0.2173	0.008
lr	Linear Regression	357.3848	5.514735e+05	731.3831	0.9602	0.2456	0.2173	0.009
lar	Least Angle Regression	357.3848	5.514736e+05	731.3831	0.9602	0.2456	0.2173	0.012
en	Elastic Net	357.3850	5.514736e+05	731.3831	0.9602	0.2456	0.2173	0.012
ridge	Ridge Regression	357.3849	5.514735e+05	731.3831	0.9602	0.2456	0.2173	0.008
huber	Huber Regressor	332.2659	5.528015e+05	732.3528	0.9601	0.1433	0.1059	0.020
knn	K Neighbors Regressor	328.7909	5.655004e+05	738.4657	0.9594	0.1386	0.0996	0.182
gbr	Gradient Boosting Regressor	329.0785	5.995877e+05	764.1338	0.9561	0.1457	0.1091	0.111

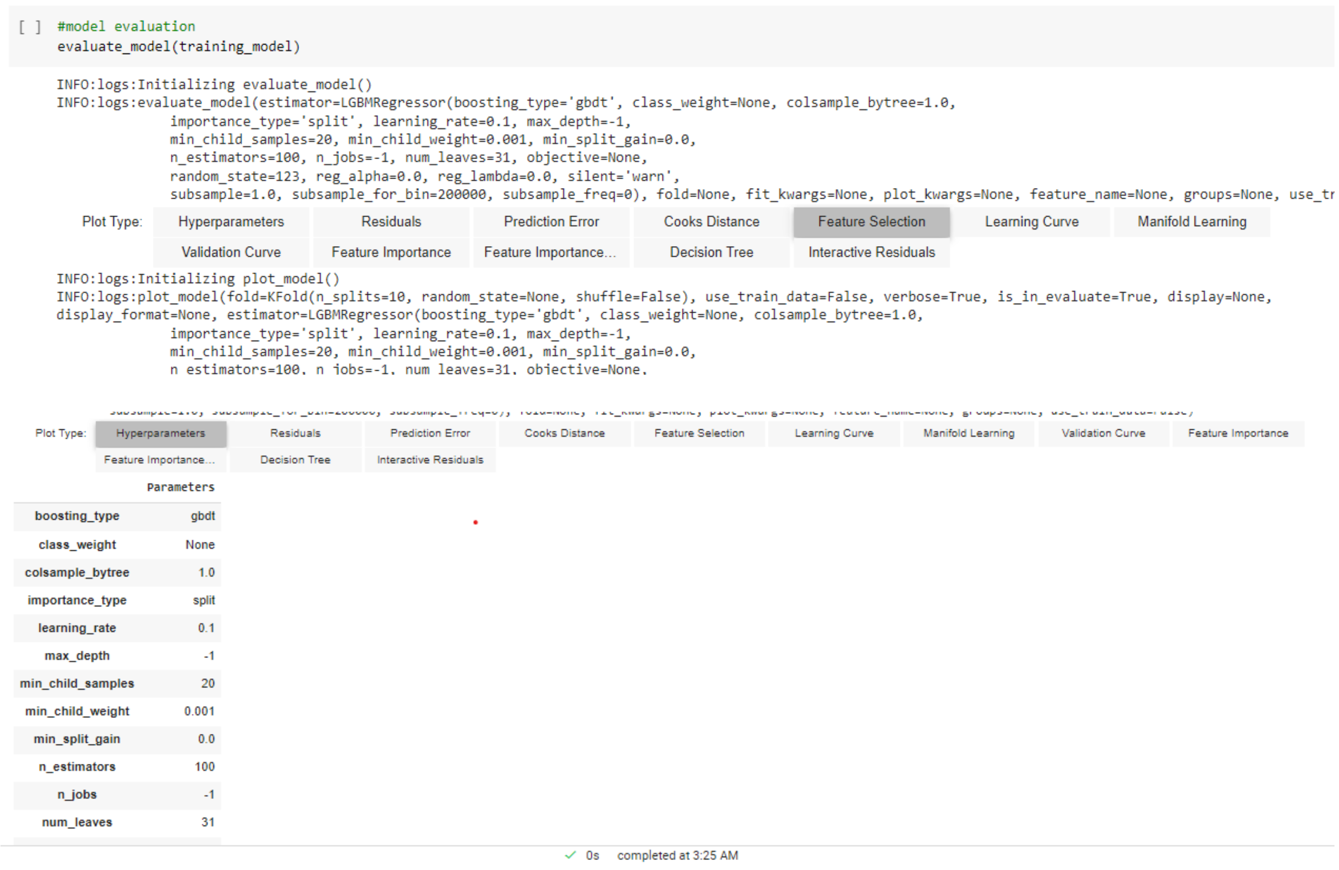
Fig 45 : train all the model and best outcome



```
#create the model and and disply the production matrix for training data sets
training_model = create_model(best_model)
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	235.2947	2.896122e+05	538.1563	0.9720	0.1290	0.0986
1	295.9197	3.212800e+05	566.8157	0.9717	0.1406	0.0998
2	279.8981	3.453254e+05	587.6440	0.9688	0.1247	0.0933
3	306.0329	5.609197e+05	748.9457	0.9628	0.1455	0.0949
4	373.1305	7.282873e+05	853.3975	0.9505	0.1383	0.0975
5	312.7246	5.365766e+05	732.5139	0.9612	0.1247	0.0966
6	333.4934	4.710762e+05	686.3499	0.9722	0.1305	0.0919
7	432.9803	1.080500e+06	1039.4712	0.9430	0.1431	0.1014
8	308.9565	4.816790e+05	694.0310	0.9589	0.1399	0.0964
9	296.5380	4.687770e+05	684.6729	0.9615	0.1300	0.0935
Mean	317.4969	5.284034e+05	713.1998	0.9622	0.1346	0.0964
Std	50.9806	2.209870e+05	140.5326	0.0092	0.0073	0.0029

Fig46; Production matrix after training

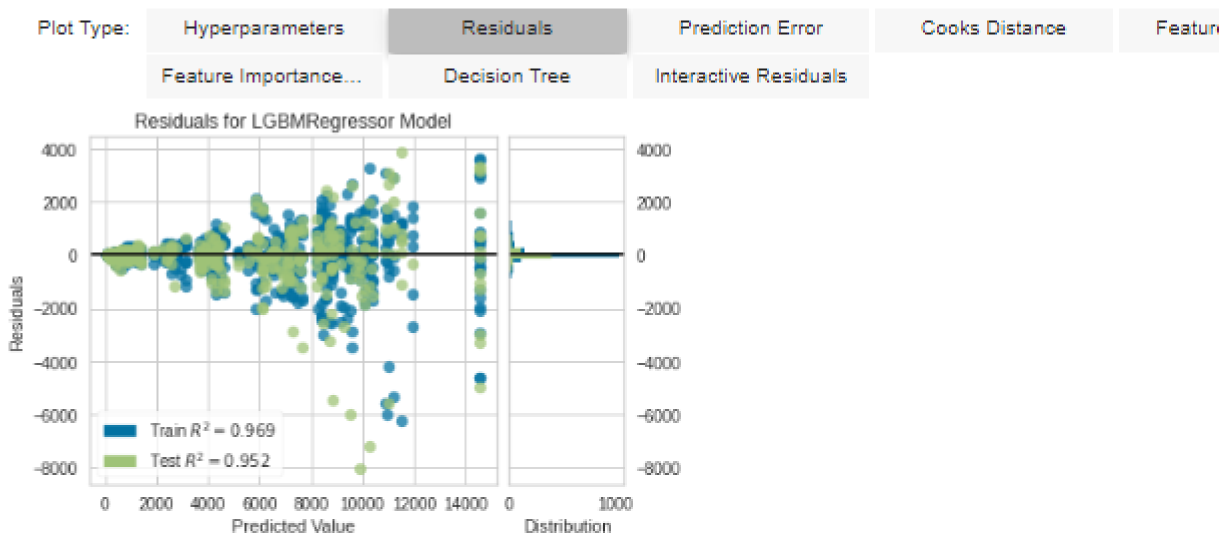


```
[ ]  #model evaluation
     evaluate_model(training_model)

     INFO:logs:Initializing evaluate_model()
     INFO:logs:evaluate_model(estimator=LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
                   importance_type='split', learning_rate=0.1, max_depth=-1,
                   min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
                   n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
                   random_state=123, reg_alpha=0.0, reg_lambda=0.0, silent='warn',
                   subsample=1.0, subsample_for_bin=200000, subsample_freq=0), fold=None, fit_kwargs=None, plot_kwargs=None, feature_name=None, groups=None, use_tr
```

```
     INFO:logs:Initializing plot_model()
     INFO:logs:plot_model(fold=KFold(n_splits=10, random_state=None, shuffle=False), use_train_data=False, verbose=True, is_in_evaluate=True, display=None,
     display_format=None, estimator=LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
                   importance_type='split', learning_rate=0.1, max_depth=-1,
                   min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
                   n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,

     subsample=1.0, subsample_for_bin=200000, subsample_freq=0), fold=None, fit_kwargs=None, plot_kwargs=None, feature_name=None, groups=None, use_train_data=False)
```

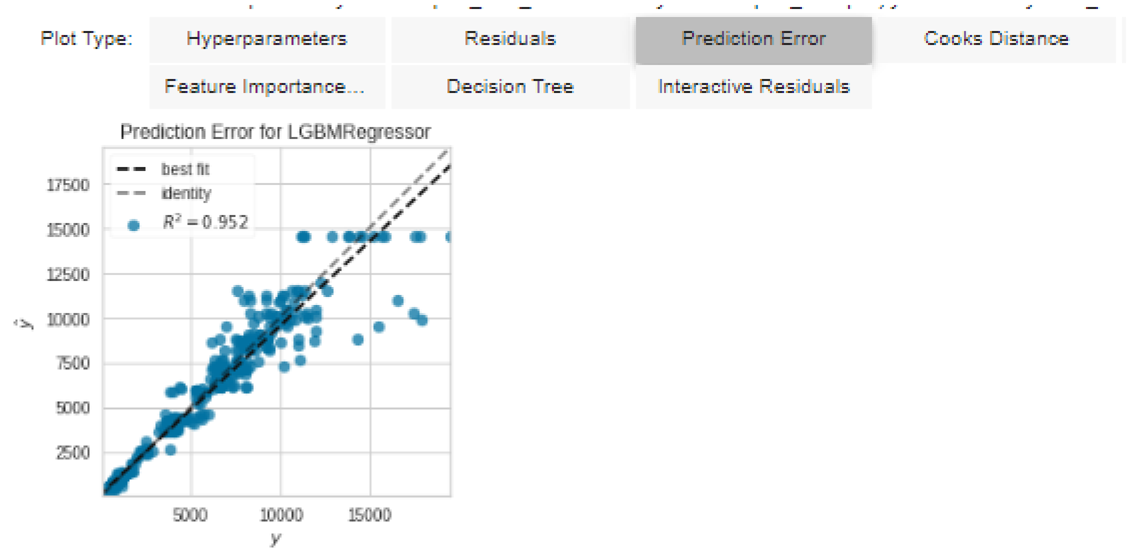
	Parameters
boosting_type	gbdt
class_weight	None
colsample_bytree	1.0
importance_type	split
learning_rate	0.1
max_depth	-1
min_child_samples	20
min_child_weight	0.001
min_split_gain	0.0
n_estimators	100
n_jobs	-1
num_leaves	31

Fig 47: Model evaluation



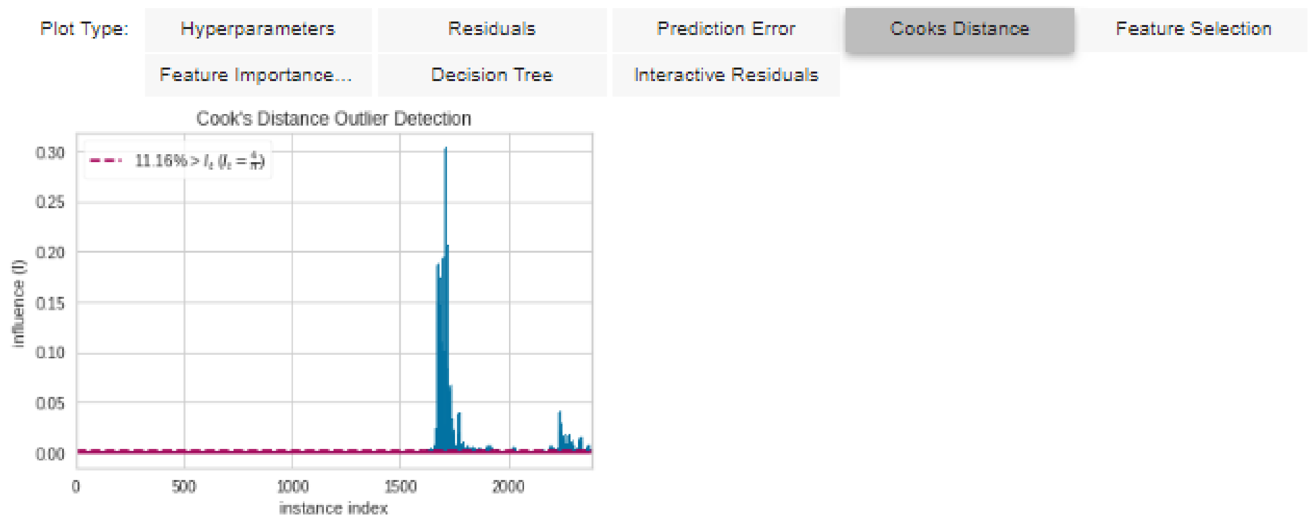
```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed......................................
```

Fig 48: Plotting Residual



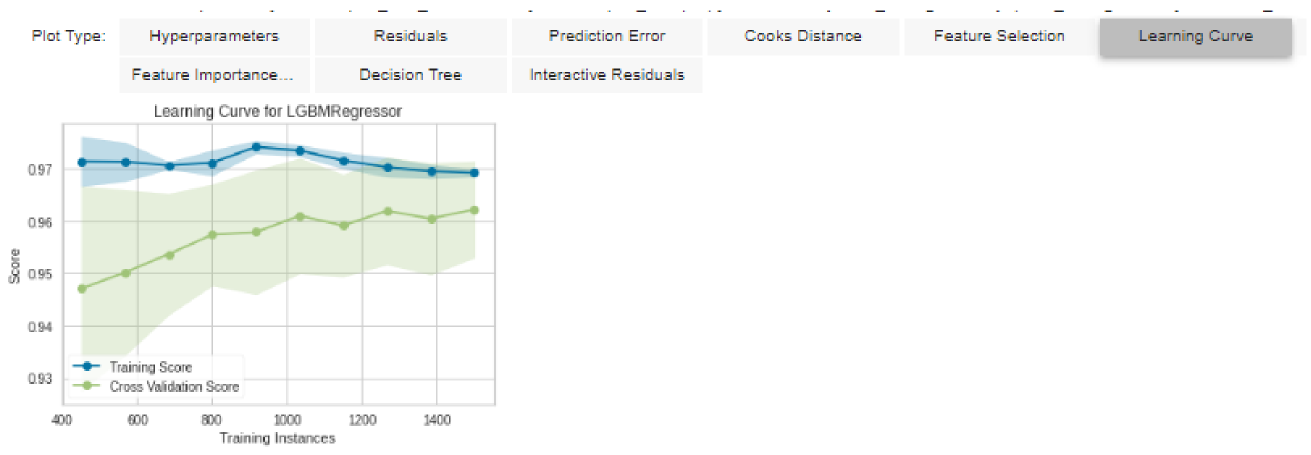
```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed......................................
```

Fig 49: Plotting prediction error



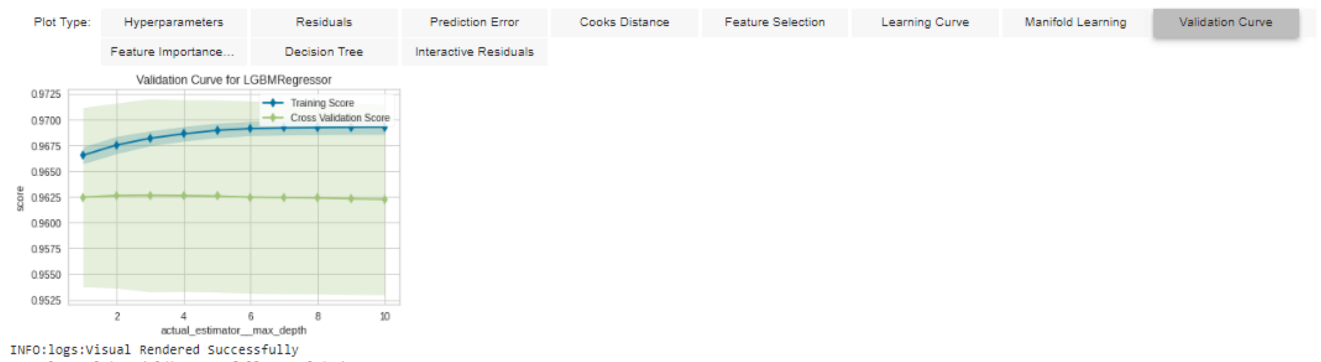
```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed......................................
```

Fig 50: Plotting Cooks distance



```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed......................................
```

Fig 51: Leaning curve



```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed......................................
```

Fig 52: Validation Curve

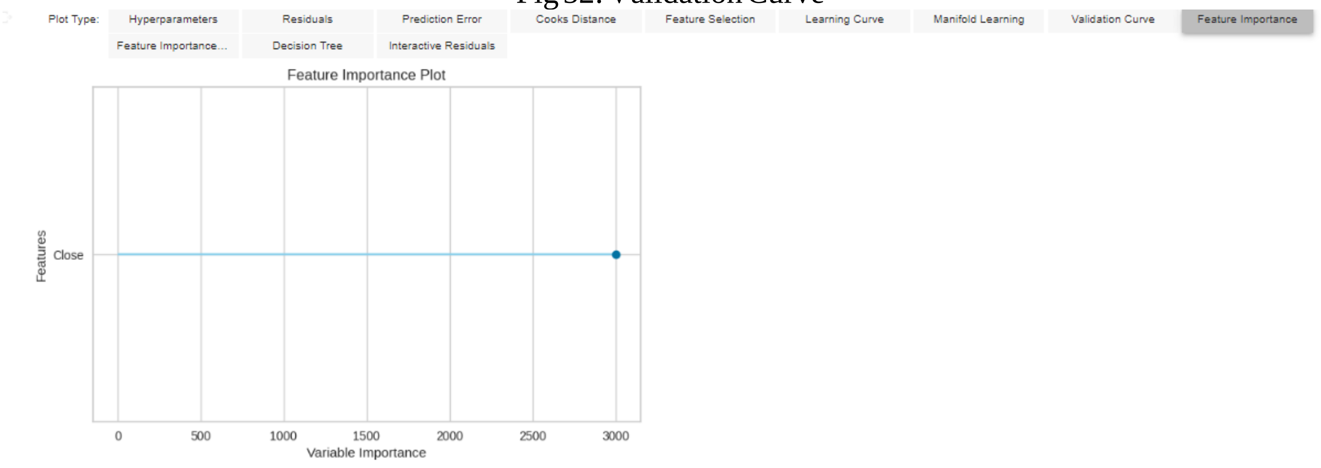
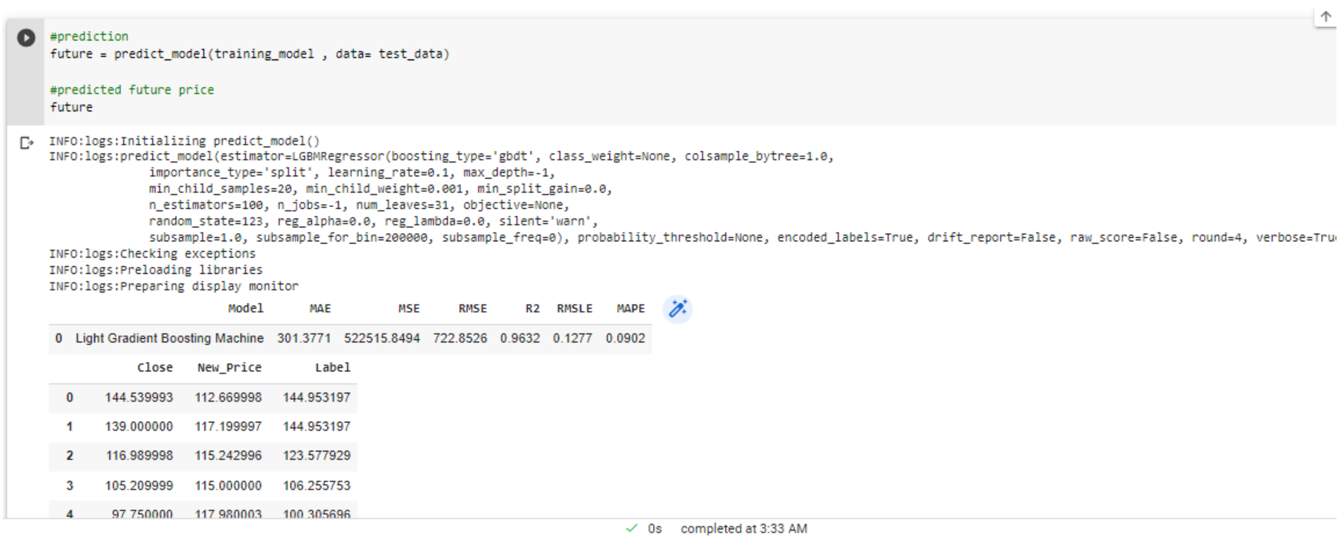


Fig53: Feature importance



```
#prediction
future = predict_model(training_model , data= test_data)

#predicted future price
future
```

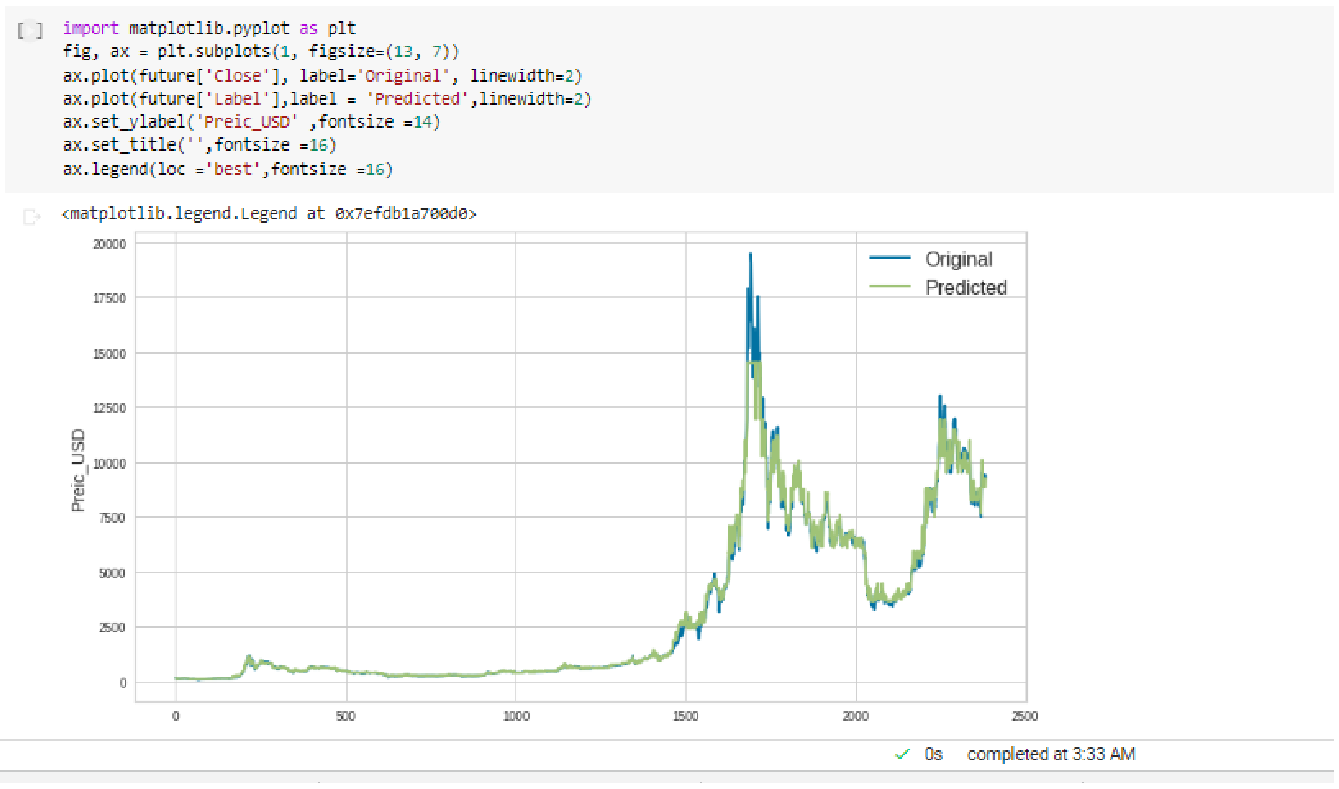
```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.1, max_depth=-1,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
              random_state=123, reg_alpha=0.0, reg_lambda=0.0, silent='warn',
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0), probability_threshold=None, encoded_labels=True, drift_report=False, raw_score=False, round=4, verbose=Tru
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Light Gradient Boosting Machine	301.3771	522515.8494	722.8526	0.9632	0.1277	0.0902

	Close	New_Price	Label
0	144.539993	112.669998	144.953197
1	139.000000	117.199997	144.953197
2	116.989998	115.242996	123.577929
3	105.209999	115.000000	106.255753
4	97.750000	117.980003	100.305696

✓ 0s completed at 3:33 AM

Fig 54: Prediction and predicted result



```
[ ] import matplotlib.pyplot as plt
    fig, ax = plt.subplots(1, figsize=(13, 7))
    ax.plot(future['Close'], label='Original', linewidth=2)
    ax.plot(future['Label'],label = 'Predicted',linewidth=2)
    ax.set_ylabel('Preic_USD' ,fontsize =14)
    ax.set_title('',fontsize =16)
    ax.legend(loc ='best',fontsize =16)
```

```
<matplotlib.legend.Legend at 0x7efdb1a700d0>
```

Fig55 : Visualization of Prediction By Pycret Model

13.Result

13.1 By Pycret Model

1. A special library of python programming called Pycret has been used to make prediction over the same dataset. This model predict the price as well.

Results presented in Fig 57 compare the actual and Pycret-predicted price of BTC.
The graph shows that the predicted and the actual price is approximately the same over the entire interval. This model is considered the best model. The mean absolute percentage error for the prediction model of BTC for Light gradient boosting model is 0.2454%, and the root mean square error (RMSE) is 713.1998 ,R2 value is 0.9622, RMSLE is 0.1346,MAPE is 0.0964, MAE is 317.4969 which is for Light gradient Bosting model can seen in Fig 45 .The mean of R2 is 0.9622,standard deviation of R2 is 0.0092, similarly Mean of MAE is 317.4969 and its respective standard deviation is 50.9806 can seen in Fig 46 Statistical analysis of the data indicates that the predicted price closing price has overlapped graph , means prediction seems accurate.Figure 57 illustrates the comparison between the original and the predicted price of BTC using the pycret. The graph shows that the difference between the predicted and the actual price is virtually non-existent along the testing set, with very small differences in the top few peaks of the time series. This model is considered to be the best. The mean absolute percentage error for the prediction model (MAPE) of the light gradient boosting model for BTC is 0.2454%.

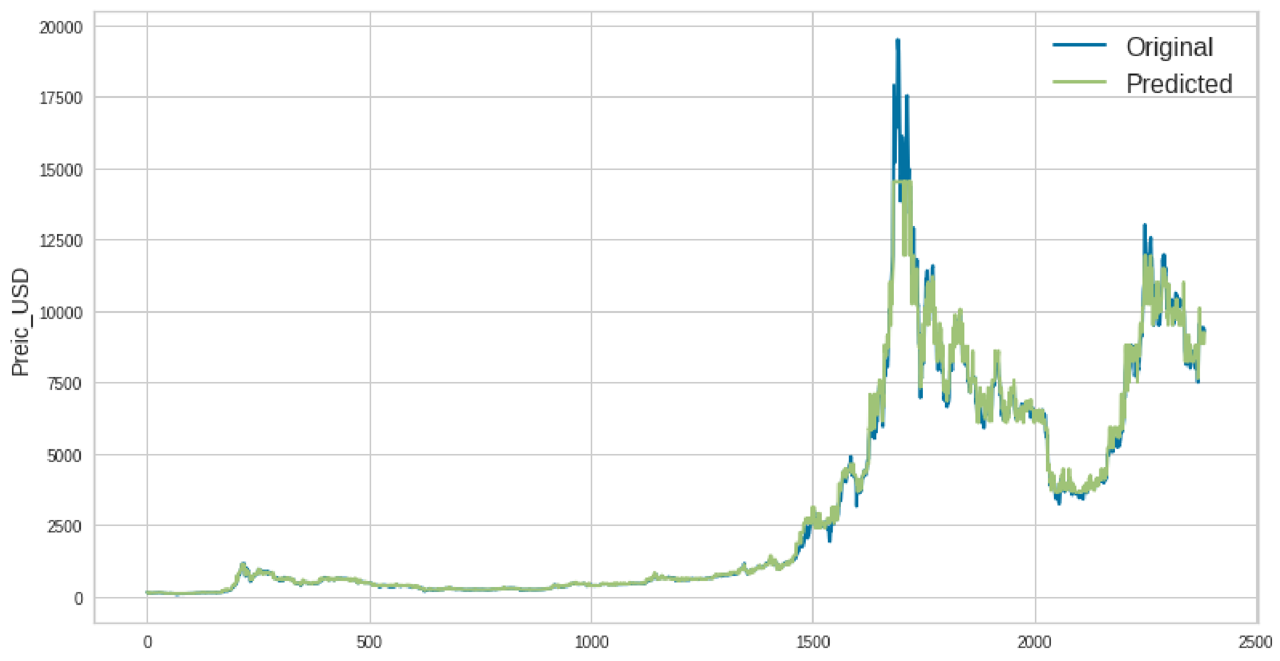


Fig 57: Result Predicted by Pycret Model

13.2 .By LSTM Model

2.A special variation of Recurrent Neural Network (RNN) viz Long short term memory(LSTM) model i s performed on predicted and actual models to know the bitcoin prediction. The Machine learning model predict the price .

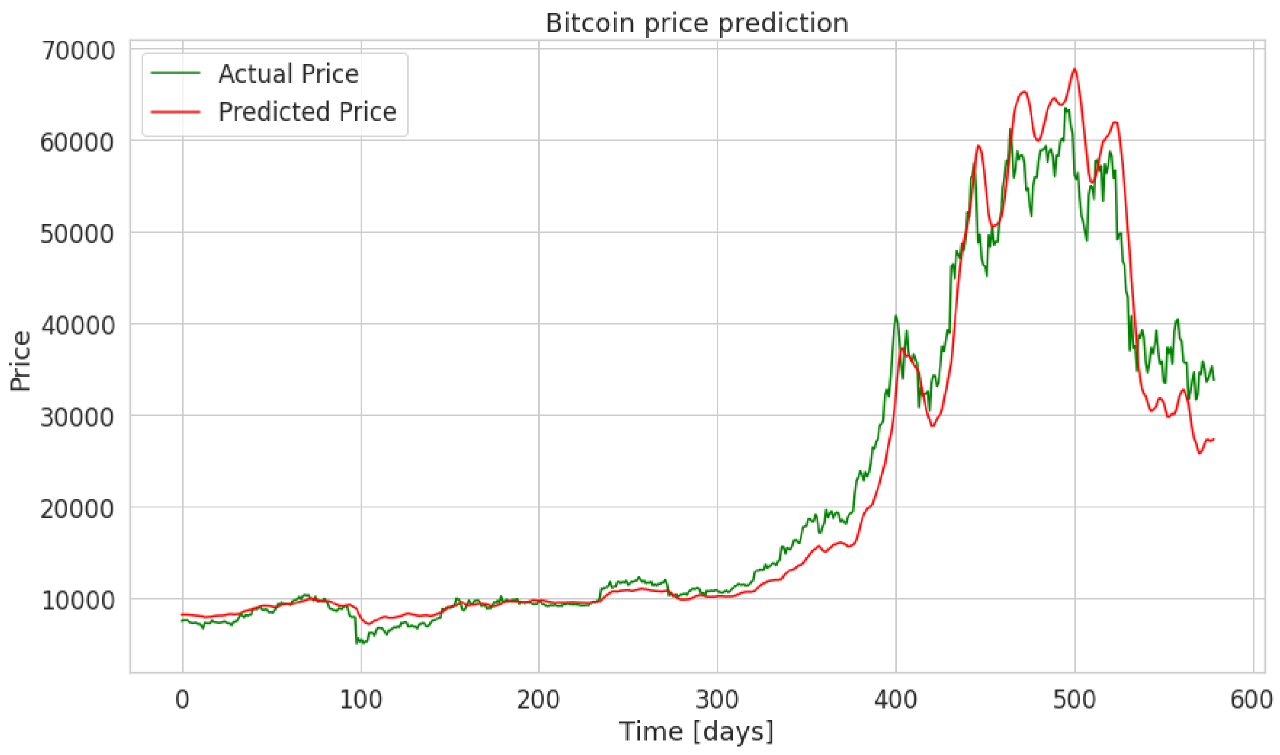


Fig 56: Result predicted by LSTM Model

The model predictions based on the accuracy scores, which are presented in Fig 36 . I find that all tested m odels' predictive accuracy is 84%. Furthermore, the models have a probability of less than 3.08E-
09 for a true accuracy of 84% output screenshot of Fig 36 BTC. Results presented in Fig 56 compare the a ctual and LSTM-
predicted price of BTC. The graph shows that the predicted and the actual price is approximately the sam e over the entire interval. This model considered the second-
best model. The mean absolute percentage error for the prediction model of BTC for LSTM is 1.1234%, a nd the root mean square error is 410.399

14.Conclusion

Since its inception in 2008, Bitcoin has gained widespread popularity among millions of users all around the world. Si nce then, academics has created a huge body of work on bitcoin transactions. Bitcoin is most well studied cryptocurre ncies. It is conducted statistical analysis focusing on the BTC time-
series data in this study between 2013 to 2021.applying various tools of data science and present it visually majorly ap plying LSTM Model based on RNN and the second is employing pythons reputed library Pycret ,future price can be p redicted using the historical data set and the bitcoin is been increasing due to the closeness values. Built a Bidirectiona l LSTM Recurrent Neural Network in TensorFlow 2. The developed model (and preprocessing "pipeline") is specific t o the historical data set for crypto.One interesting direction of future investigation might be analyzing the correlation b etween different cryptocurrencies and how would that affect the performance of our model.

In this project , 2 types of machine learning algorithm are constructed and used for predicting the prices of three types of cryptocurrency—
BTC. The comparison of actual and predicted prices, based on above calculation for the targeted cryptocurrencies (Bi tcoin)can be considered efficient and reliable according to both the model.
However, Pycret represents more accuracy than LSTM and Pycret predict with substantial low differences between the actual and the predicted prices for both BTC The Fig 57 clearly depict that. So it can conclude that,

- The python library Pycret is reliable and acceptable for cryptocurrency prediction
- Light gradient boosting in pycret model can predict cryptocurrency prices comparatively better than LSTM but overall all both algorithms represent excellent predictive results.

In future work, I will investigate other factors that might affect the prices of the cryptocurrency market, and will focus on the effect that social media in general and tweets in particular can have on the price and trading volume of cryptocur rencies by analyzing tweets using natural language processing techniques and sentiment analysis.

References

seaborn.boxplot seaborn 0.11.1 documentation. (n.d.). Seaborn.pydata.org. https://seaborn.pydata.org/generated/seaborn.boxplot.html

Ahmad, H. (2019, December 23). Forecasting Future Prices of Cryptocurrency using Historical Data. Medium. https://towardsdatascience.com/forecasting-future-prices-of-cryptocurrency-using-historical-data-83604e72bc68

Ahmad, H. (2019, December 5). Exploratory Data Analysis of CryptoCurrency Historical Data. Medium. https://medium.com/@hamzaahmad86/exploratory-data-analysis-of-cryptocurrency-historical-data-d8ec719641e7

Roberts, C. (2022, November 29). Python Time Series Analysis Guide. Medium. https://medium.com/@mrconnor/python-time-series-analysis-guide-6903f49f40ac

Sharma, R. (2022, May 12). Is There a Cryptocurrency Price Correlation to Equity Markets? Investopedia. https://www.investopedia.com/news/are-bitcoin-price-and-equity-markets-returns-correlated/

Ahmad, H. (2019, December 23). Forecasting Future Prices of Cryptocurrency using Historical Data. Medium. https://towardsdatascience.com/forecasting-future-prices-of-cryptocurrency-using-historical-data-83604e72bc68

G-Research Crypto Forecasting. (n.d.). Kaggle.com. Retrieved November 30, 2022, from https://www.kaggle.com/c/g-research-crypto-forecasting/

Edwards, J. (2022, July 2). Bitcoin's Price History. Investopedia. https://www.investopedia.com/articles/forex/121815/bitcoins-price-history.asp

Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. Work Pap; 2008. https://bitcoin.org/bitcoin.pdf.

Beck R, Avital M, Rossi M, Thatcher JB. Blockchain technology in business and information systems research. Bus Inf Syst Eng.2017;59:381e384. https://doi.org/10.1007/s12599-017-0505-1.

Gu S, Kelly B, Xiu D. Empirical asset pricing via machine learning. Rev Financ Stud. 2020;33:2223e2273. https://doi.org/10.1093/rfs/hhaa009.

JonaThan Donier Jean-Philippe Bouchaud, Julius Bonart. TRADES, QUOTES AND PRICES. Financial Markets Under the Microscope. Cambridge University Press, 2018.

Lo AW. The adaptive markets hypothesis. In: Adaptive Markets. vol. 30. Princeton University Press; 2019:176e221. https://doi.org/10.2307/j. ctvc7778k.9.

Darryl Shen. Order imbalance-based strategy in high frequency trading. Master's thesis, Linacre College,University of Oxford, 2015.

Valkov, V. (2019, June 29). Cryptocurrency price prediction using LSTMs | TensorFlow for Hackers (Part III). Medium. https://towardsdatascience.com/cryptocurrency-price-prediction-using-lstms-tensorflow-for-hackers-part-iii-264fcdbccd3f