Pattern Recognition & Machine Learning (CSL 2050)

Bitcoin Price Prediction

Image source -
https://www.google.com/search?q=bitcoin+price+prediction&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj4htSXsLf3AhU9UGwGHXKhBqEQ_AUoBHoECAEQBg#imgrc=VMhI5tl5xUoBHM

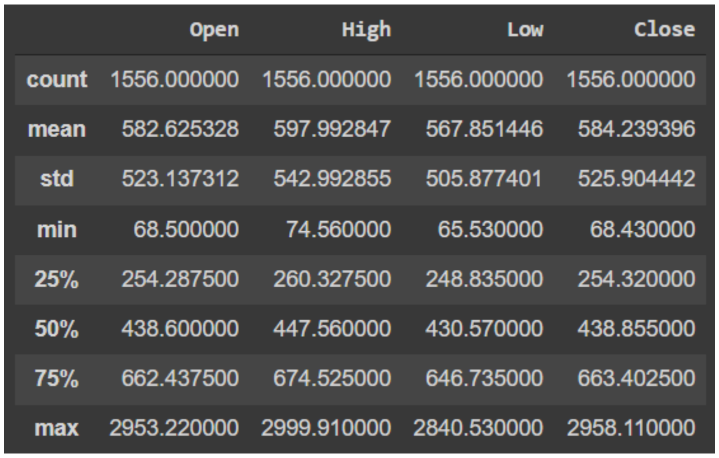
BONUS PROJECT

Riyanshu Jain
B20AI060

Preprocessing

Analysis of dataset -

The dataset contains 1556 rows and 7 columns, where the description of columns is as follows –



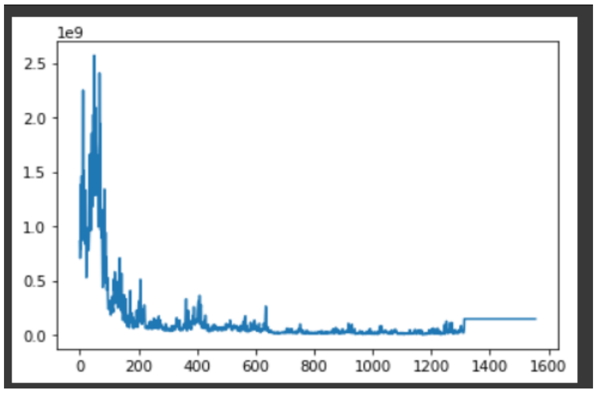
	Open	High	Low	Close
count	1556.000000	1556.000000	1556.000000	1556.000000
mean	582.625328	597.992847	567.851446	584.239396
std	523.137312	542.992855	505.877401	525.904442
min	68.500000	74.560000	65.530000	68.430000
25%	254.287500	260.327500	248.835000	254.320000
50%	438.600000	447.560000	430.570000	438.855000
75%	662.437500	674.525000	646.735000	663.402500
max	2953.220000	2999.910000	2840.530000	2958.110000

Apart from these, rest of the columns are –

- Date : Values ranging from April 28, 2013 to July 31, 2017
- Volume : Integer values in the range of 10^8
- Market Cap : Integer values in the range of 10^9

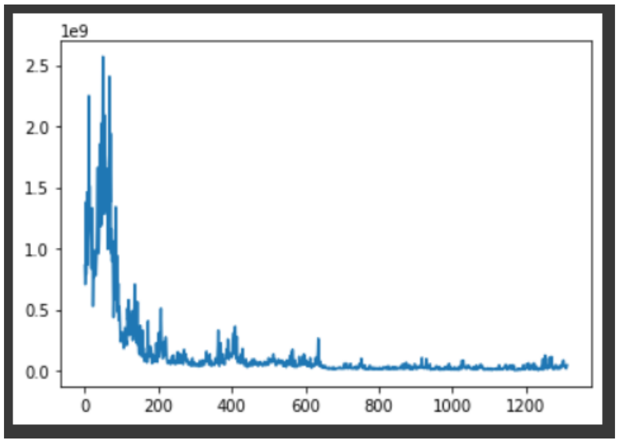
From here we can clearly see that the scale of all the variables are not the same and we will definitely need to rescale the values to get good results from any model.

There are also many entries in 'Volume' column with their values as '-', which is not acceptable for an integer column, hence we filled the null entries with the mean of column. The plot after replacing with the mean is as follows –



As we can see that in the end there is a clear constant line which is clearly not fitting with rest of the curve, hence we can't replace the null entries with the mean.

So, next I tried to fit the curve with the use of np.polyfit function and the plot obtained is as follows –

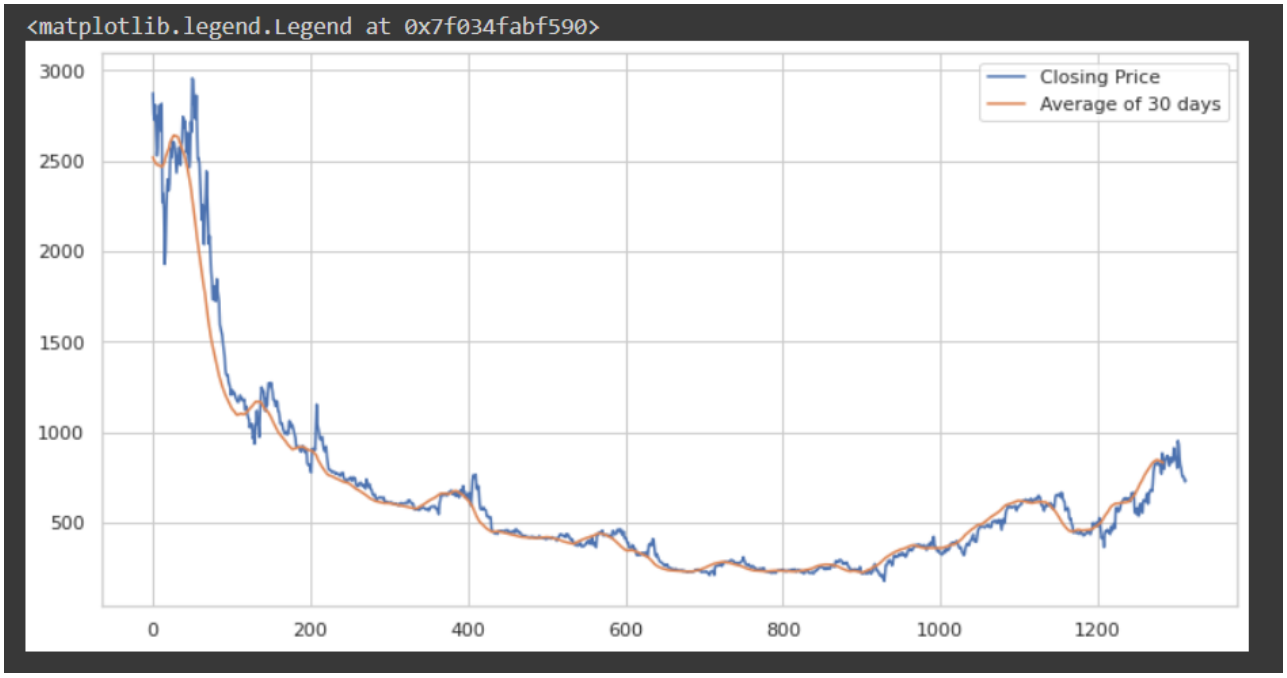


- Here, we can see that the curve fitting is somehow justified with the previous values and can be used for training purposes.

But, one more point arises that the missing data in the column is of year 2013, which will not play a major role in predicting the prices for the years, later than 2014. Hence, at the end, I decided to drop the rows with missing data.

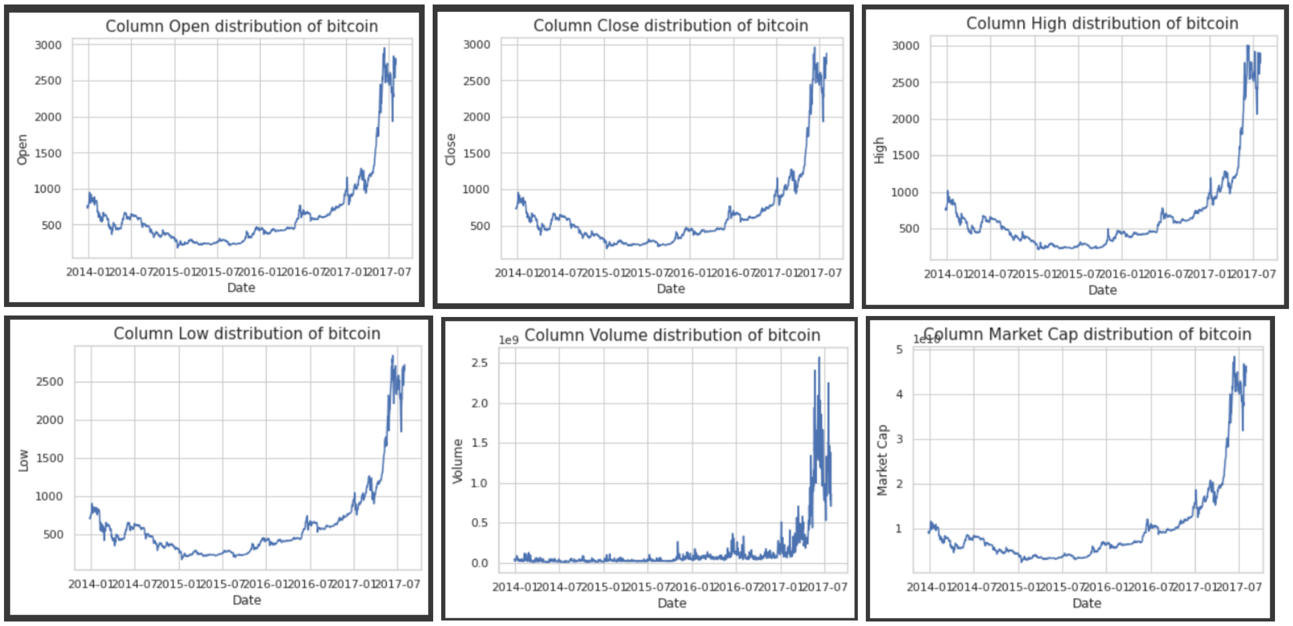
Data Visualization

- Plot of the Closing Price along with the rolling average of 30 days –



```
<matplotlib.legend.Legend at 0x7f034fabf590>
```

- Plots of the each column with date on x-axis, and column values on y-axis –

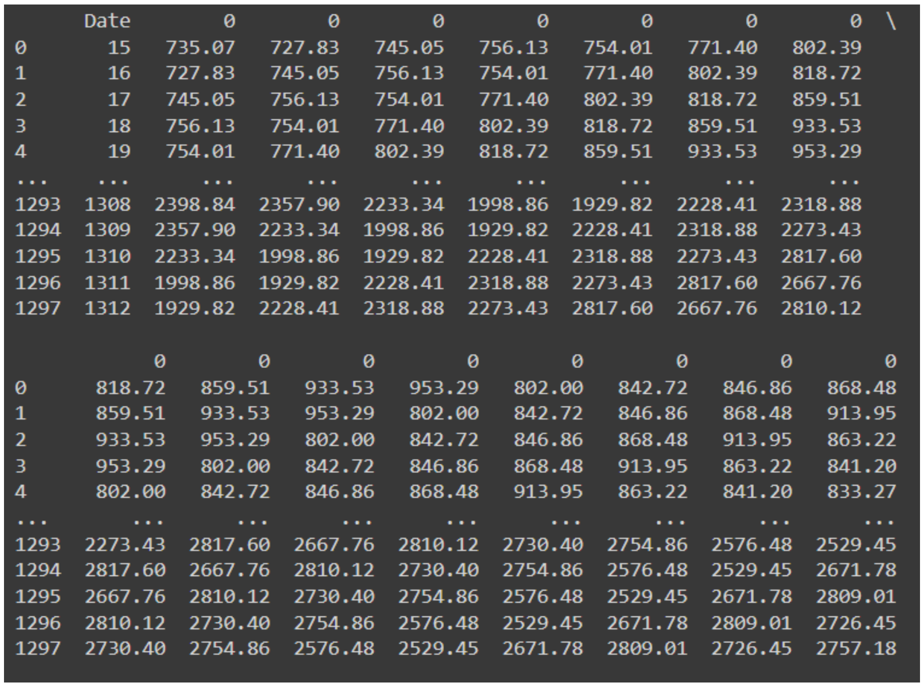


From here, we can see that there is a similarity in the curve of each column other than the Volume, so we can predict that the while forecasting the Volume column's values, they will be greatly varied in comaprison to rest of the columns.

Methodology 1

(for predicting the current value based on the previous 15 days of the data)

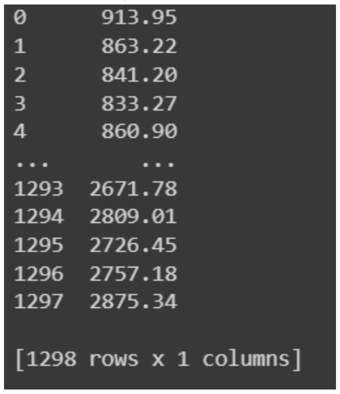
- For this purpose, prepare_data function is created which prepares the data for every column based on the previous 15 days data as the features to predict the label, which is current day's prediction. The resultant train dataset for the column 'Close' looks like –



```
      Date        0       0       0       0       0       0       0  \
0       15   735.07  727.83  745.05  756.13  754.01  771.40  802.39
1       16   727.83  745.05  756.13  754.01  771.40  802.39  818.72
2       17   745.05  756.13  754.01  771.40  802.39  818.72  859.51
3       18   756.13  754.01  771.40  802.39  818.72  859.51  933.53
4       19   754.01  771.40  802.39  818.72  859.51  933.53  953.29
...    ...      ...     ...     ...     ...     ...     ...     ...
1293  1308  2398.84 2357.90 2233.34 1998.86 1929.82 2228.41 2318.88
1294  1309  2357.90 2233.34 1998.86 1929.82 2228.41 2318.88 2273.43
1295  1310  2233.34 1998.86 1929.82 2228.41 2318.88 2273.43 2817.60
1296  1311  1998.86 1929.82 2228.41 2318.88 2273.43 2817.60 2667.76
1297  1312  1929.82 2228.41 2318.88 2273.43 2817.60 2667.76 2810.12

            0       0       0       0       0       0       0       0
0      818.72  859.51  933.53  953.29  802.00  842.72  846.86  868.48
1      859.51  933.53  953.29  802.00  842.72  846.86  868.48  913.95
2      933.53  953.29  802.00  842.72  846.86  868.48  913.95  863.22
3      953.29  802.00  842.72  846.86  868.48  913.95  863.22  841.20
4      802.00  842.72  846.86  868.48  913.95  863.22  841.20  833.27
...       ...     ...     ...     ...     ...     ...     ...     ...
1293  2273.43 2817.60 2667.76 2810.12 2730.40 2754.86 2576.48 2529.45
1294  2817.60 2667.76 2810.12 2730.40 2754.86 2576.48 2529.45 2671.78
1295  2667.76 2810.12 2730.40 2754.86 2576.48 2529.45 2671.78 2809.01
1296  2810.12 2730.40 2754.86 2576.48 2529.45 2671.78 2809.01 2726.45
1297  2730.40 2754.86 2576.48 2529.45 2671.78 2809.01 2726.45 2757.18
```

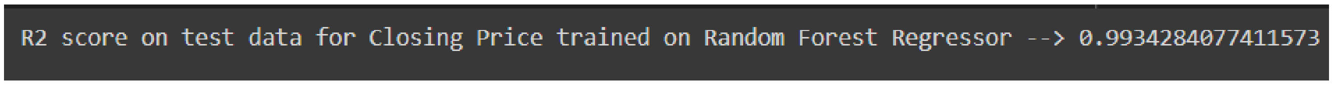
It contains 16 columns, where 15 columns are of previous 15 days data and 16th column is of date and the value of date is expressed in integer format, which is nothing but the number of days from the starting date of the dataset. And the test dataset is as follows containing the actual value for that particular date –



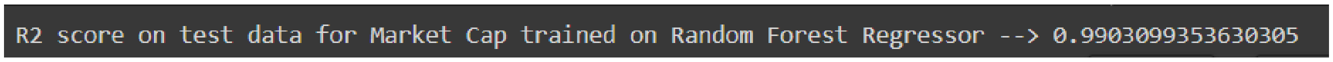
```
0       913.95
1       863.22
2       841.20
3       833.27
4       860.90
...        ...
1293   2671.78
1294   2809.01
1295   2726.45
1296   2757.18
1297   2875.34

[1298 rows x 1 columns]
```

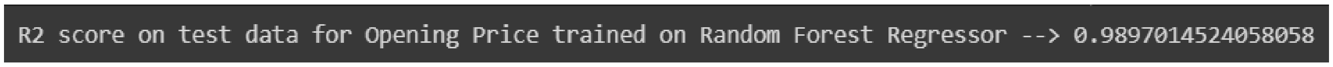
Then the Random Forest Regressor is used for forecasting the values and the results obtained are as follows –



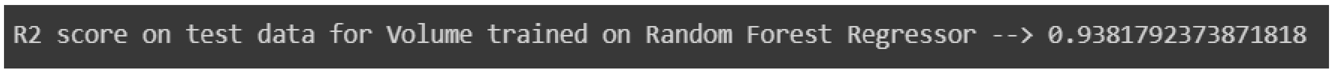
```
R2 score on test data for Closing Price trained on Random Forest Regressor --> 0.9934284077411573
```



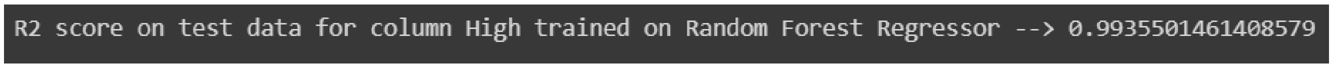
```
R2 score on test data for Market Cap trained on Random Forest Regressor --> 0.9903099353630305
```



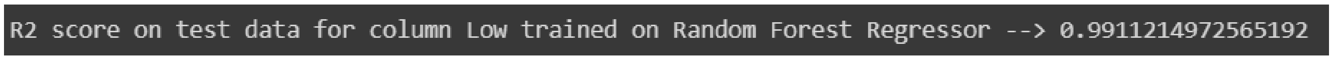
```
R2 score on test data for Opening Price trained on Random Forest Regressor --> 0.9897014524058058
```



```
R2 score on test data for Volume trained on Random Forest Regressor --> 0.9381792373871818
```



```
R2 score on test data for column High trained on Random Forest Regressor --> 0.9935501461408579
```



```
R2 score on test data for column Low trained on Random Forest Regressor --> 0.9911214972565192
```

As we can see from the results, the R2 score is 0.99 for almost all column values but is less for Volume column as we have predicted above from the plot of Volume v/s Date.

Methodology 2

(for predicting the Market Cap value based on rest of the columns)

- In this experiment, top 1000 rows are given as train data and the rest of the 313 rows are used as test data. The reason being that the last entries in the test data contains a wide range of values from 9*(10^9) to 4*(10^10).
- Then the values are rescaled using the standard scaler function of sklearn. Then we applied two models on the train data and predicted the values of test data. The two models chosen are – SVM and Linear regression. The results are as follows –

```
R2 score on SVR model for Market Cap Prediction -->  0.9736249498295703
```

```
R2 score on Linear Regression model for Market Cap Prediction -->  0.982343429008827
```

The comparison between the predicted and the actual values for both the models are saved in the colab file shared, you can have a look at them too. The values came out to be close which is evident from the r2 score also.

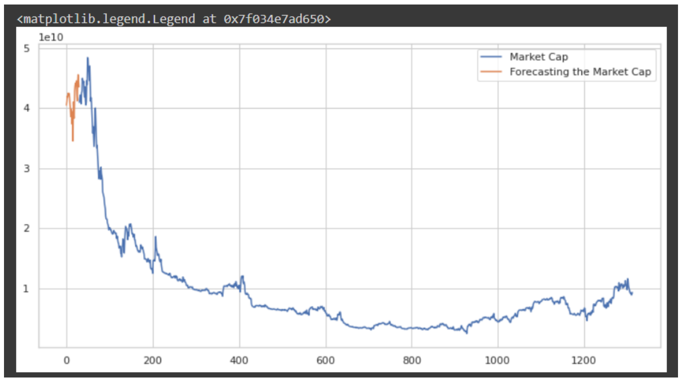
Methodology 3

(for predicting the Market Cap value after a month)

- In this experiment, a new column is created 'Price after a month' which is the label and its data is to be predicted from the current values. The values of the new column are nothing but the values of the Market Cap column shifted by an amount of 30 values. Hence we are predicting the value of the next month using the current date.
- The Random forest regressor is then trained on the data and the results of test data came out to be –

```
R2 score on RF model is : 0.9945894632581953
```

As we shifted the data by 30 days after, we don't have any data for the ending 30 days in the dataset and hence, the data is forecasted with the trained model and is plotted as follows –



```
<matplotlib.legend.Legend at 0x7f034e7ad650>
```

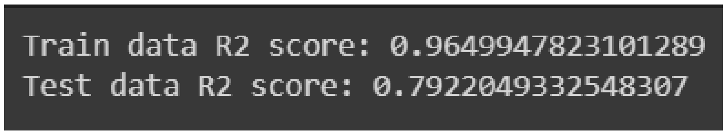
The blue curve represents the available data and the orange curve is the forecasted data for the next 30 days.

Methodology 4

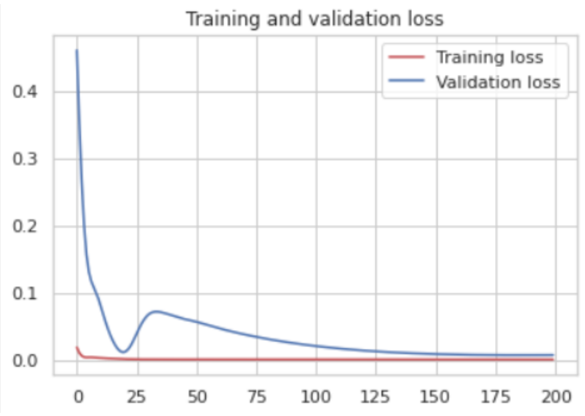
(for predicting the Market Cap value using LSTM)

- Long Short-term Memory model is the most popular model in time-series domain. It is based on recurrent neural networks to deal with temporal data and contains memory state due to which it is capable of learning dependencies on time.
- In this method, the 365 days data (2016-08-02 to 2017-07-31) is used in which starting 70 percent is used for training and the rest 30 percent for testing.
- Now, the model is imported from keras tensorflow library and trained using the time step (or lookback value) as 15, loss function as mean squared error and optimizer as Adaptive Moment Estimation 'adam'.

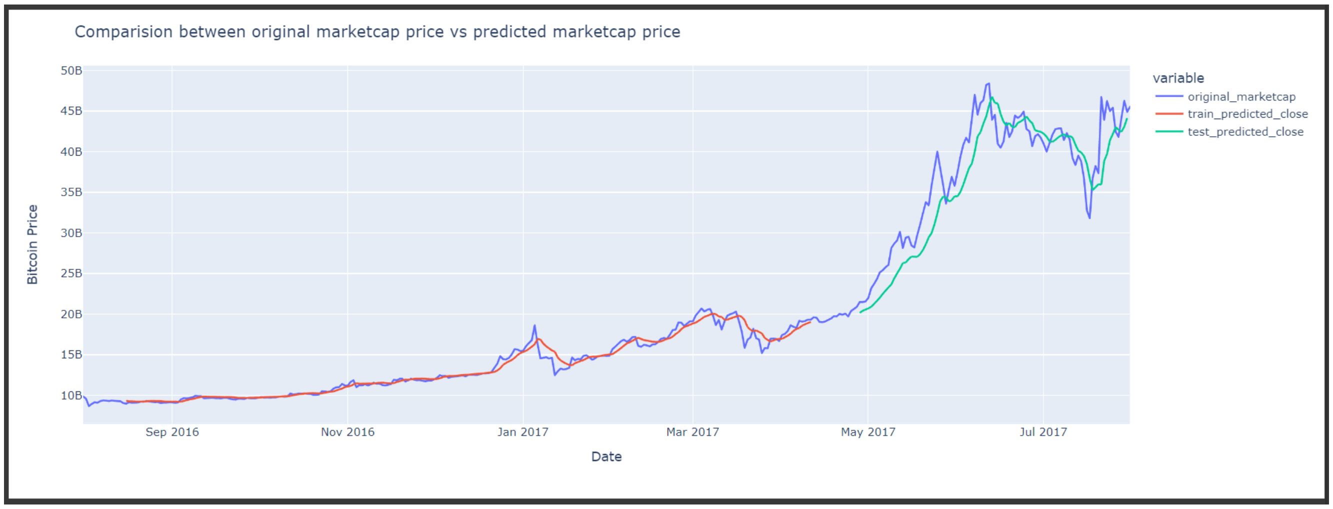
The plot for train and validation loss for 200 epochs and 32 batch size is as follows –



```
Train data R2 score: 0.9649947823101289
Test data R2 score: 0.7922049332548307
```



The comparison plot for original values, train_prediction and test_prediction is as follows –



Now, at the end, comparing all the four methodologies implemented, we can clearly see that by methodology 1, we have got the best results taking r2_score as metric. But the models implemented in all four methods were not the same, nor the train and test data were same in all of them as we have used different ideas in all of the four methodologies.

This is all about the project explaining my ideas, experiments and the results.

The End….