Predicting Bitcoin Price using Real Time Price and Tweets from Twitter

Yassaman Davilu

MARCH 24, 2019

Table of Contents

LIST OF FIGURES

Data Scraping and Basic Exploration

The main objective of the project is to predict the bitcoin prices using tweets from twitter and real time Open, High, Low and Close (OHLC) data. We will be looking forward to build a more robust framework and hence predict the prices of bitcoin using twitter stream and OHLC data coming from Crypto trading Platform.

Firstly, we imported the required libraries which will used to load and transform data, building features, conducting statistical analysis and model building and evaluation. Twitter provides two APIs which are streaming API and rest API, streaming API is used to fetch the tweets continuously from Twitter, but has a limit of 3200 tweets per minute and only provide one week older tweets only. Hence, in order to conduct the analysis and incorporate tweets as a feature, we needed a method to fetch tweets which of at least 3 months, in order to make the research informative and meaningful.

We use Get Old Tweets scraper, which is publicly available as open source on GitHub and works well with Python 2.7 as well as Python 3.6 which takes input query, number of maximum tweets and the date range within which we want to get tweets in. Once the tweets were obtained, we used utility function to clean the text in a tweet by removing links and special characters using regex and also classified the sentiment and polarity of the tweets using Text Blob API. After assigning the

tweets, the sentiments, we looked into the percentage of tweets as per the each class of the sentiment as shown below.

	Date	Tweet	SA	positive	neutral	negative
0	2019-01-01	What I said With ILP trillions transactions pe...	0.042857	True	False	False
1	2019-01-02	Bitcoin The Terrifying Rise Financial Blacklis...	-0.500000	False	False	True
2	2019-01-03	Happy birthday Bitcoin This bumblebee software...	0.525000	True	False	False
3	2019-01-04	FF Witty_Crypto kissmybtc Juni0rLancaster ltc_...	0.450000	True	False	False
4	2019-01-05	If bitcoin bank pocket RavenCoin 401kinvestmen...	0.000000	False	True	False

Figure 1 Tweets and Extracted Sentiments Data frame

- *Percentage of positive tweets: 56.63%*
- *Percentage of neutral tweets: 24.1%*
- *Percentage of negative tweets: 19.27%*

We stored last three months tweets and which was enough for making this research. Next, in order to get the bitcoin OHLC data, we used CoinMarketCap.com and fetch the Bitcoin OHLC prices from last three months.

	Date	Open	High	Low	Close	Volume	Market Cap
0	2019-03-24	4035.16	4040.70	4006.19	4022.17	9144851065	70823042992
1	2019-03-23	4022.71	4049.88	4015.96	4035.83	9578850549	71056017910
2	2019-03-22	4028.51	4053.91	4021.54	4023.97	9252935969	70840048102
3	2019-03-21	4083.95	4097.36	4005.15	4029.33	10831212662	70926226882
4	2019-03-20	4070.79	4089.46	4031.11	4087.48	10175916388	71942444088

Figure 2 Bitcoin OHLC Data frame

Exploratory Data Visualization

After creating the bitcoin prices data and the tweets data, we did some exploratory data visualization o make understanding of the data and understand the patterns of different prices such as Open, High, Low and Close, Market Cap as shown in the figures below.

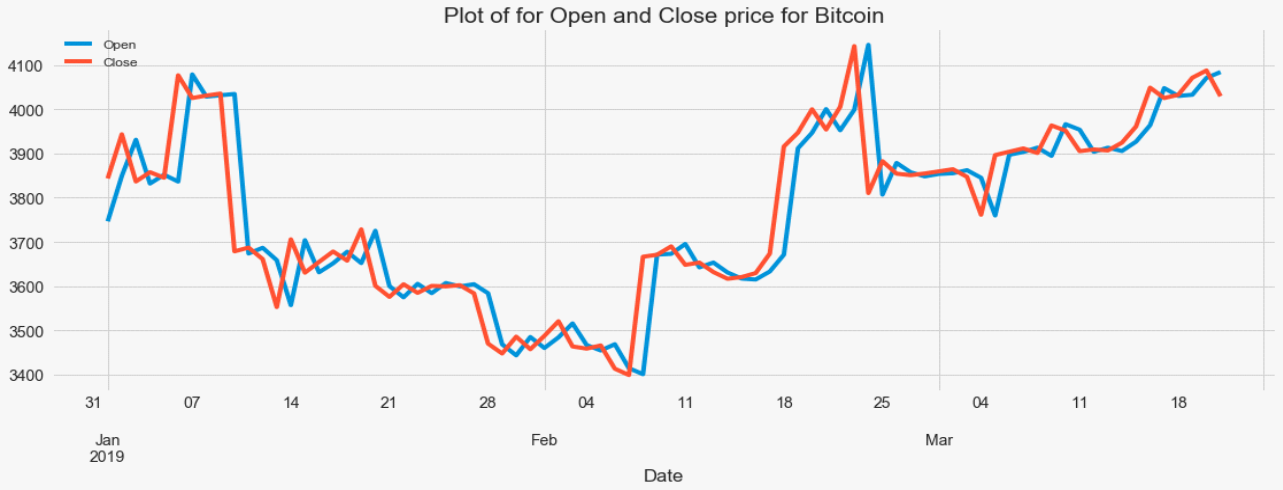

Figure 3 Plot of for Open and Close price for Bitcoin

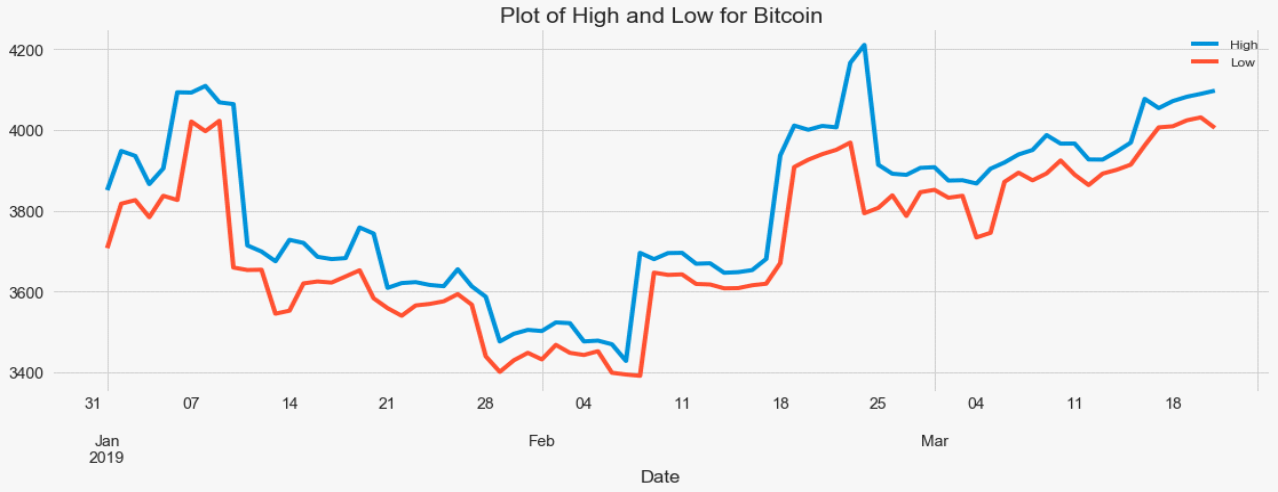

Figure 4 Plot of High and Low for Bitcoin

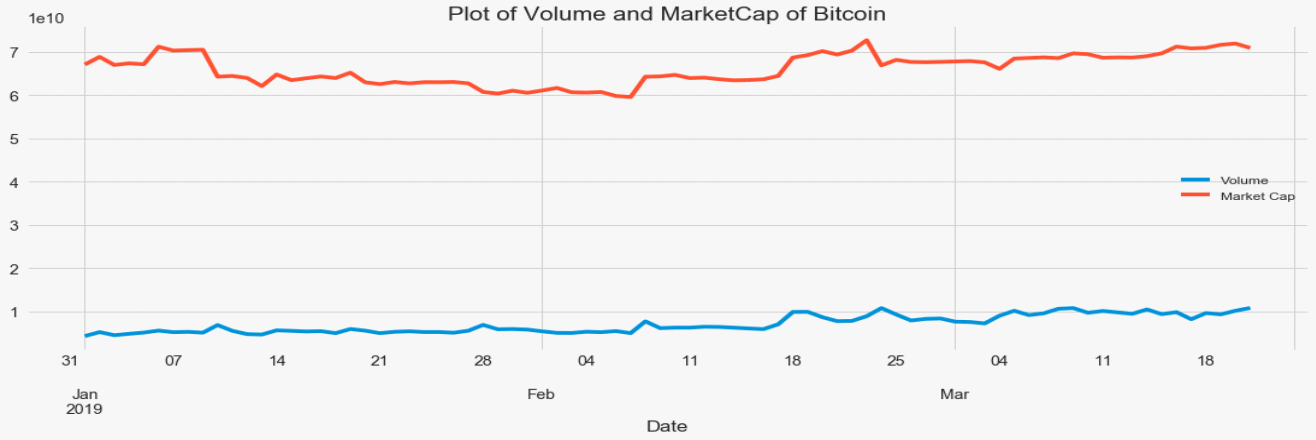

Figure 5 Plot of Volume and Market Cap of Bitcoin

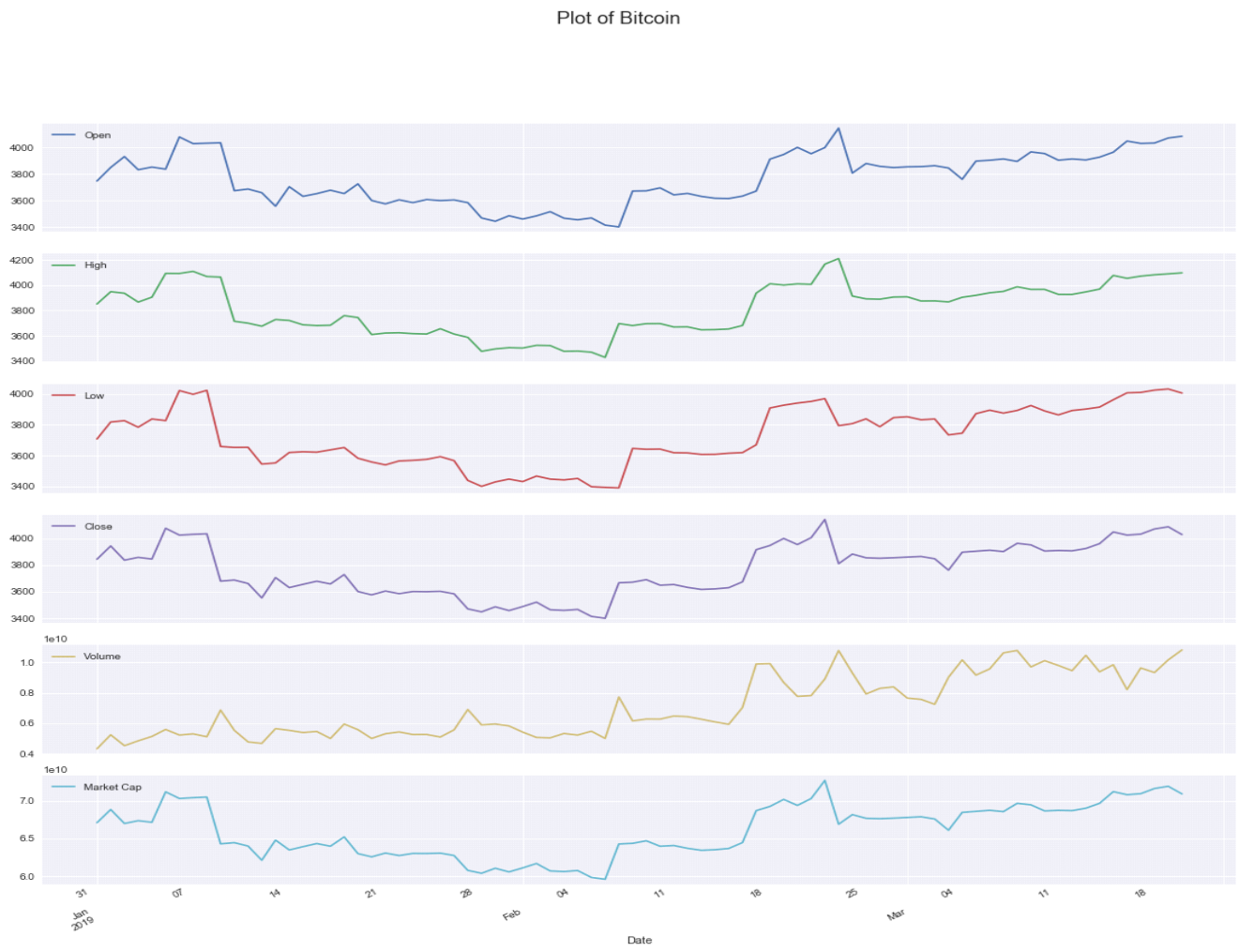

Figure 6 OHLC Combined Plot

Statistical Analysis

The next step was to conduct statistical analysis to look into the trends and patterns in the time series data which can help us further to make decisions in model building and evaluation.

We checked if the time series of the Close Price was stationary. There are

many methods to check whether a time series (direct observations, residuals, otherwise) is stationary or non-stationary.

Visual Inspection: By looking at the plots review the time series plot of your data and visually check if there are any obvious trends or seasonality.

Summary Statistics: Review the summary statistics for your data for seasons or random partitions and check for obvious or significant differences.

Statistical Tests: It is to check if the expectations of stationarity are met or have been violated.

1. Trend which corresponds to Long term analysis of the series.
2. Seasonality which is long-term but responsible for periodicity
3. Residual explaining Noise

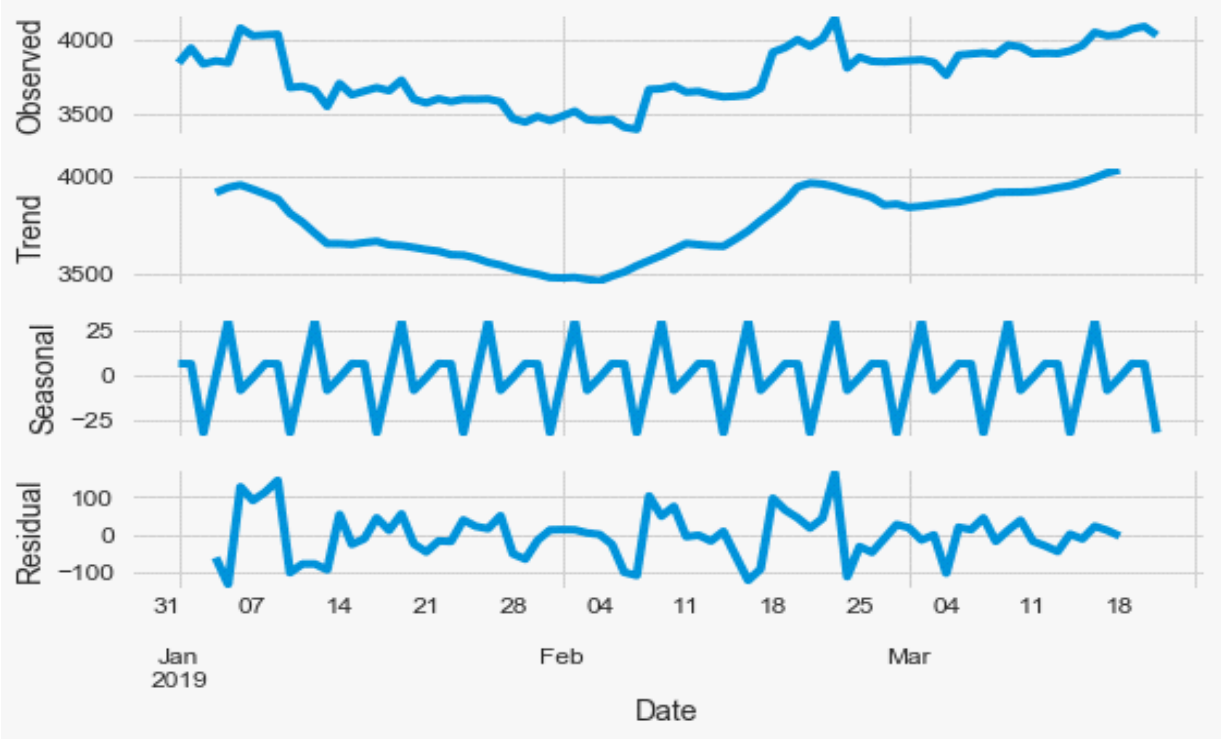

Figure 7 Residual, Seasonality and Trend

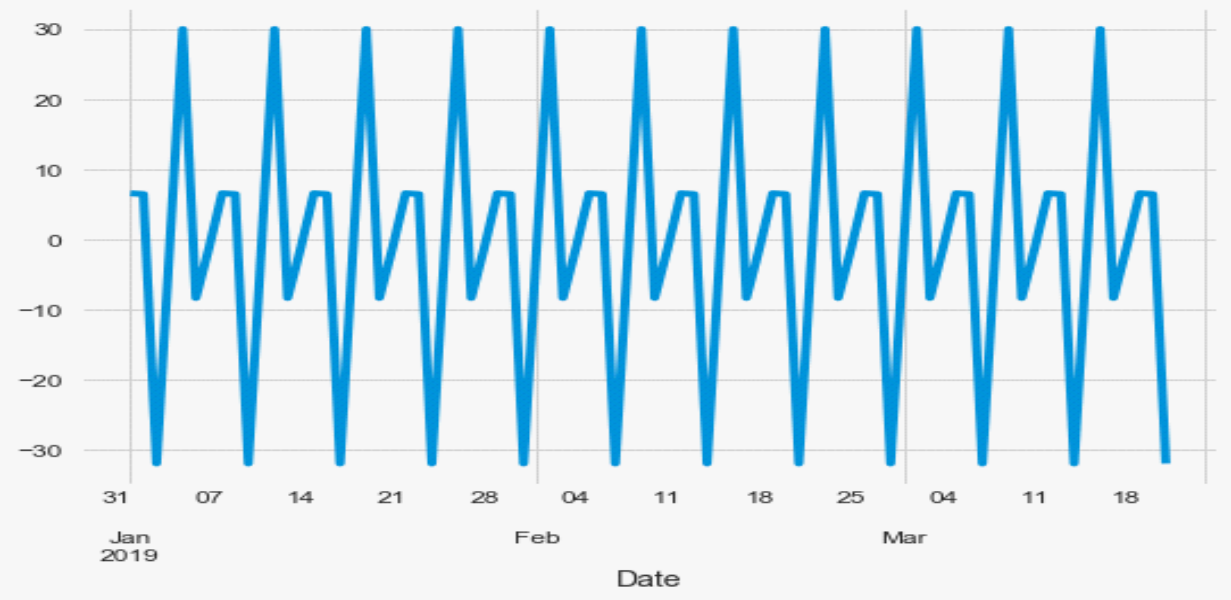

Figure 8 Seasonality of Close Price

The above graph shows the seasonality of the Close Price, which explains the

repetitive oaten in the time series data.

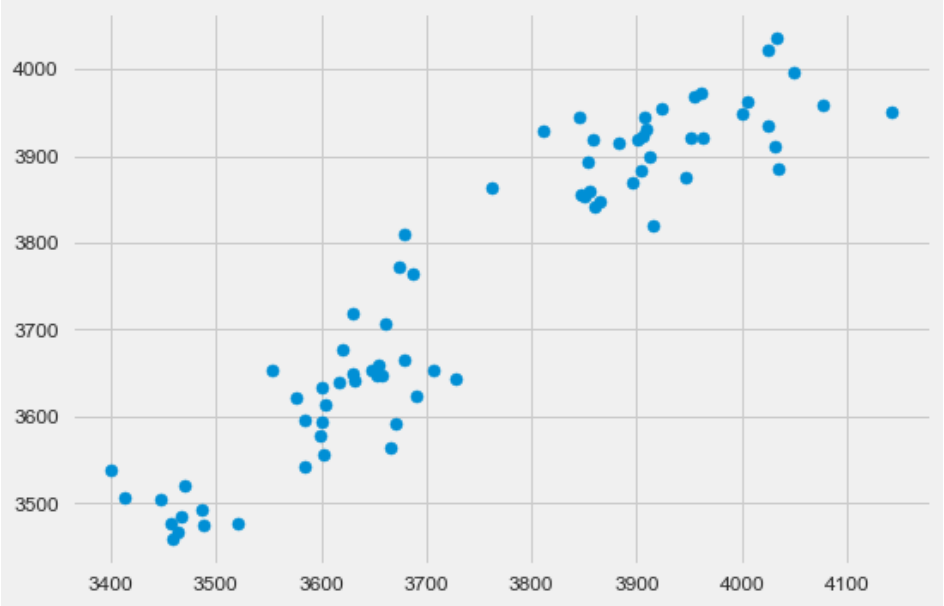

Figure 9 Decomposing Trend for original series data

The magnitude of the seasonal change increases over time as the data values increase i.e. the magnitude of the peaks and drops increases as they vary consistently or seasonally over time.

1. If the magnitude of the seasonal changes is constant, then the seasonal changes are additive.

2. If the magnitude of the seasonal changes is greater when the data values are greater, then the seasonal changes are multiplicative. The extra variability can make multiplicative seasonal changes harder to forecast accurately

Based on the above rules and observations, the Bitcoin series data is

Multiplicative as the Seasonal plot for no model applied and multiplicative is the same. Next, we conducts stationarity test which depicts the p-value and t-test statistics or Dickey Fuller test which are shown in the figures below.

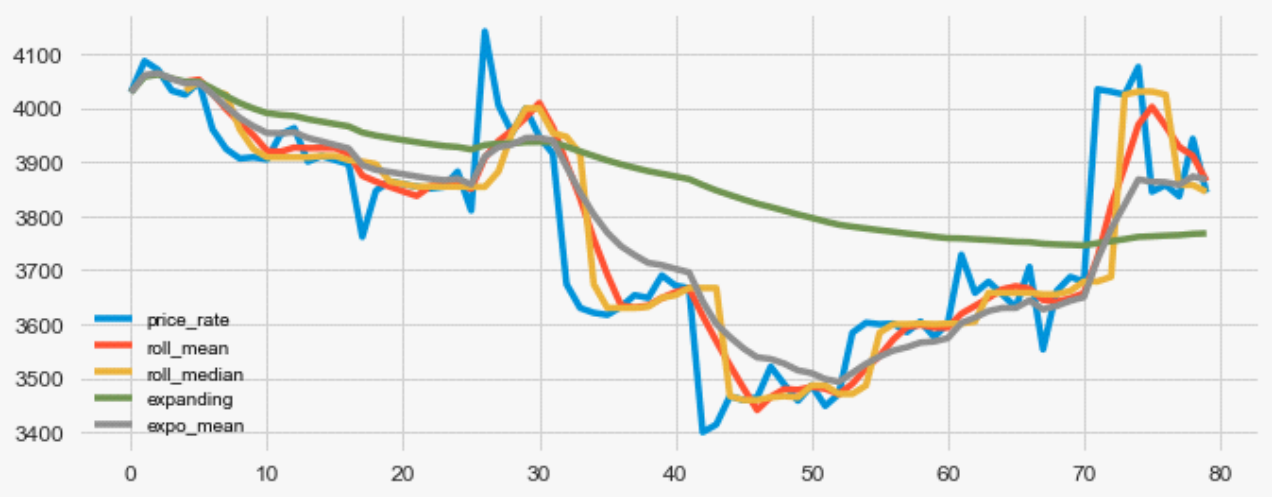

Figure 10 Stationarity test which depicts the p-value and t-test statistics or Dickey Fuller test

Based on the reference:

a. p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.
b. p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

From the above Stationary test, the Time series is not stationary since the p-val = 0.18

Feature Engineering and Machine Learning Modelling

After conducting the statistical analysis, the next step was to perform feature engineering on the tweets data frame and merge the prices data. For tweets, we cleaned the text and created TFIDF features which are best features to create for textual data. After, creating the textual features, we merged the price and the tweets data as a final data frame.

Support Vector Machine

Firstly, we build a Support Vector Machine Regression Model with radial kernel and found that, the Mean Absolute error and the Mean Squared error were close to 349.11.

	True Close Price	Predicted Close Price
0	4025.23	3698.12
1	4032.51	3698.12
2	4071.19	3698.12
3	4087.48	3698.12
4	4029.33	3698.12
5	4023.97	3698.12
6	4035.83	3698.12
7	4022.17	3698.12

Figure 11 Support Vector Machine - Model Evaluation

We can change the algorithm from Support vector Machine to tree based learning algorithms, which can learn the data more accurately and does not over fit.

Random Forest

Then, we trained a Random Forest Repressor with 100 number of trees and two as a maximum depth, then we calculated the Root mean Square Error and Mean Absolute Error which were found to be 27.85 and 36.63 respectively as shown in the figure below.

	True Close Price	Predicted Close Price
0	4025.23	4009.627362
1	4032.51	4009.627362
2	4071.19	4009.627362
3	4087.48	4009.627362
4	4029.33	4009.627362
5	4023.97	4009.627362
6	4035.83	4009.627362
7	4022.17	4009.627362

Figure 12 Random Forest - Model Evaluation

XG Boost (XGB) Regressor

Next, we trained XG Boost (XGB) Regressor with optimum parameters, to check how the accuracy of the model is increased and we found the RMSE and MAE is dropped significantly as shown in the figure below to 19 and 25.

	True Close Price	Predicted Close Price
0	4025.23	4029.117920
1	4032.51	4070.323242
2	4071.19	4080.256836
3	4087.48	4080.256836
4	4029.33	4070.323242
5	4023.97	4028.976074
6	4035.83	4079.733887
7	4022.17	4029.405029

Figure 13 XG Boost - Model Evaluation

Long Short Term Memory (LSTM)

The next part of the research is to build the model using LSTM, we also scaled the features and used principal component analysis to reduce the dimension of the features to avoid overfitting and hence to make sure the model understand the features in best manner. Next, we have to reshape the data. As LSTM model, which we are going to build in next step only accepts input with 3-D shape. We are using numpy to do that.

Now, we had data and it is reshaped as well as well normalized, so we can feed it to LSTM model for training and future predictions. To create model we are using Keras framework as it is most easy to use and very much user friendly. The following is the architecture of the LSTM model, we created.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 128)               71168
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 16)                1040
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 80,481
Trainable params: 80,481
Non-trainable params: 0
```

Figure 14 LSTM Model Structure

The results obtained from LSTM are shown below.

	True Close Price	Predicted Close Price
0	0.842139	0.803344
1	0.851937	0.826184
2	0.903992	0.850370
3	0.925914	0.875366
4	0.847657	0.790542
5	0.840444	0.805765
6	0.856405	0.837596
7	0.838021	0.803454

Figure 15 LSTM - Model Structure

It was found that the LSTM model did not performed better than Xgboost, and the unscaled features are also distributed more in a feature range. It can be concluded that, if we tune the LSTM model more, we can achieve better model and hence,

Xgboost model performed the best. It is also to be noted that, by the increasing the size of the data, we can further enhance the model performance and incorporating more tweets, we can assure yourself to get better predictions as a future scope of the research.

Tuning Extreme Boosting Algorithm

Extreme Boosting is a tree boosting and a highly effective and widely used machine learning method, which is used widely by data scientists and researchers to achieve state-of-the-art results on many machine learning challenges. The main advantage of using Xgboost is it does not over-fit and is highly efficient and reliable as compared to neural networks. Xgboost learn creating trees and optimizes multiple trees, creating a strong learner from weak learner.

Xgboost is invariant to scaling of inputs, so we do not need to do careful about features normalization as it understand using decision trees. Xgboost also learn higher order interaction between features and can be scaled.

Xgboost works on the concept of boosting. This is an ensemble decision tree method which creates a strong classifier (model) based on weak classifiers. In this context, weak and strong refer to a measure of how correlated are the learners to the actual target variable. By adding models on top of each other iteratively, the errors of the previous model are corrected by the next predictor, until the training data is

accurately predicted or reproduced by the model.

Now, gradient boosting also comprises an ensemble method that sequentially adds predictors and corrects previous models. However, instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction. So, in the end, we are updating our model using gradient descent and hence the name, gradient boosting. This is supported for both regression and classification problems. Xgboost specifically, implements this algorithm for decision tree boosting with an additional custom regularization term in the objective function. So, firstly, we implemented Decision tree algorithm based on a single tree and then we created Xgboost algorithm.

We also found that LSTM model did not perform well and over fits and it can be concluded that for this challenge, it is obvious that tree based models and especially Extreme Boosted Trees is very superior to Neural Network models as we are combining all the features together, which include bitcoin prices as well as textual features.

The final step was to tune the model Xgboost model, so, we tuned Xgboost parameters using hyper parameter optimization. Xgboost is also a based on an ensemble of decision trees, but different from random forest. The trees are not

averaged, but added.

I initially fixed the number of trees to 500 which was determined that with a quick experiment, then we can find good values for the other parameters.

The most important parameters are:

1. **Number of trees** (n_estimators)

2. **Learning rate** - later trees have less influence (learning_rate)

3. **Tree complexity** (max_depth)

4. **Gamma** - Make individual trees conservative, reduce overfitting

5. **Column sample per tree** - reduce overfitting

6. **Subsample** - Would randomly sample half of the training data prior to growing trees and this will prevent overfitting.

7. **Reg Alpha** - Regularization term on weights. Increasing this value will make model more conservative. Normalized to number of training examples.

8. **Minimum Child weight** - This simply corresponds to minimum number of instances needed to be in each node.

After tuning the parameter, we obtained the best values of the each parameter using hyper parameter tuning and it was found that the RMSE again dropped to 16,

which is a significant achievement.