



# Predicting Bitcoin Price using Time Series and Tweets

Capstone Final Project Report

Shashindra Silva  
Shashindra3@gmail.com

## Contents

List of Figures .....	2
Acronyms and Abbreviations.....	2
1. Abstract.....	3
2. Introduction .....	3
3. Literature survey.....	3
4. Data Acquisition.....	4
5. Data Preprocessing.....	5
6. Sentiment Analysis .....	5
7. Time series Analysis.....	7
8. Time series analysis with average sentiment.....	11
9. Deployment .....	12
Final model.....	12
Deployment.....	13
Deployment on EC2 .....	16
Results.....	17
10. Conclusion.....	18
11. References .....	18

## List of Figures

Figure 1: Preprocessing Tweets .....	5
Figure 2: Compare sentiment values from different methods.....	7
Figure 3: Training and testing data .....	8
Figure 4: Auto ARIMA Results .....	9
Figure 5: LSTM model .....	10
Figure 6: Predictions from FB prophet on the BTC data .....	11
Figure 7: MSE and SMAPE for various models.....	11
Figure 8: Best feature combination .....	12
Figure 9: LSTM Model .....	13
Figure 10: Sample file to access the API .....	14
Figure 11: Flask app to predict the value .....	15
Figure 12: AWS EC2 Instance .....	16
Figure 13: Files on the AWS EC2 .....	16
Figure 14: Invoking the API through local machine .....	17
Figure 15: Model for the final application .....	18

## Acronyms and Abbreviations

ARIMA	Autoregressive Integrated Moving Average
VADER	Valence Aware Dictionary and Sentiment Reasoner
NLP	Natural Language Processing
LSTM	Long Short Term Memory
BTC	Bitcoin
MSE	Mean Square Error
SMAPE	Symmetric Mean Absolute Percentage Error
AWS	Amazon Web Services

## 1. Abstract

In this project, time series methods and NLP methods were used to predict the future price of bitcoin. Several time series methods such as ARIMA , fb prophet, and LSTM models were used initially only using time series data of the bitcoin price. Then sentiment analysis was done of the tweets mentioning bitcoin and that value was used as an exogenous variable for the prediction. Several other features such as number of tweets, average number of likes, etc. were also tested. However, using the average sentiment value obtained from VADER, average number of replies to tweets, and the number of tweets in combination with previous days bitcoin price gave the best accuracy in terms of mean absolute percentage error. In conclusion using a simple sentiment analysis on the tweets mentioning bitcoins will improve the accuracy of the bitcoin price prediction.

## 2. Introduction

Bitcoin is a form of decentralized digital currency that was introduced in 2009 by Satoshi Nakamoto [1]. It is one of the earliest and possibly the most popular cryptocurrencies [2] in the world. While the initial price of a BTC was less than a single dollar in year 2010, it reached over 80000 USD in year 2021. However, more recently the price of BTC has dropped back to 20000 USD. The concept of BTC and other crypto currencies has generated mixed revies among people. Some people claim that whole BTC and crypto schemes is a scam, and some people believe that it is the future of finance. These mixed sentiments on BTC often results in heated discussion on social media platforms such as Twitter and Reddit.

Also, unlike traditional stock prices the buying and selling of BTC occurs throughout the whole day. Thus, the data for BTC transaction are available for whole 24 hours. The objective of this project is to utilize these data with the social media interactions that occurs in Twitter to predict the future price of BTC.

## 3. Literature survey

The work in [3] presents one of the earliest works on predicting BTC price based on Twitter data. Apart from BTC price it also looks at other crypto currencies such as Ethereum. Also, it looks at google trend data when predicting the price. However, the prediction of the BTC price is focussed on the direction of the price (increasing or decreasing) instead of looking at absolute value. One of the findings of the paper was that the sentiment about BTC is always positive regardless of the increase or the decrease of the price. Also, the authors have found out that the tweet volume has a more correlation towards the BTC price than the sentiment values.

Sentiment analysis of tweets is based on VADER model. When it comes to the machine learning models, the authors have used a linear model to predict the BTC price.

Authors of [4] also look at forecasting BTC price based on the tweet data. Unlike the previous work, here the focus is on the tweet sentiment values and historical BTC prices. Here authors have obtained an accuracy score of 63% for the predictions. Similar to [3], this work focuses on the direction of the BTC price. The average scores of sentiment analysis is used to predict the direction of price for BTC by using the Random Forest Regression binary classification model. The use of data related to only a selected 60 days is one of the major drawbacks of this paper.

The citation [5] takes a different approach to predict the BTC price. Instead of using traditional machine learning models, this paper uses reinforcement learning techniques to predict the BTC price. Here authors have used four main attributes: number of followers of the poster, number of comments on tweets, number of likes, and number of retweets. It was found out that the tweets with the most user-related attributes had the greatest effect on the BTC price. Apart from the numerical accuracies CPU computation powers and times have been analyzed for the proposed method. Another recent work in [6] tries to predict the BTC price and the price difference using the tweet sentiment and volume.

## 4. Data Acquisition

Acquisition of data is an important part of an end-to-end machine learning project. Having reliable data will highly impact the accuracy of the system. For this project two separate information values are required.

### 1. BTC daily price

Obtaining the BTC price is easy as it is available as a time series from many financial service websites and APIs such as yahoo finance. However for this project, data available at [7] is used. This includes data starting from 17-Sep-2014 and up to 09-July-2021. The main reason for this selection was that this data set contained aggregated data of BTC price for each day. For the analysis we used the “Open” price of the BTC data.

### 2. Tweets mentioning bitcoin and its price

Next important part of data is the set of tweets mentioning BTC. For this data we utilized data available at [8]. This contained tweets scrapped from twitter from 2016-Jan-01 to 2019-Mar-29 mentioning words “Bitcoin” or “BTC”. The authors in [8] have used Twint [9] and Tweepy [10]. This contained 16 million tweets/

Above mentioned data will be used to train and test the model. However, to use the system with new data, updated tweet data is required. To obtain new tweet data Twint [9] is used.

Code to setup Twint and how to use it to obtain tweets mentioning BTC is demonstrated in [11] and [12].

## 5. Data Preprocessing

The next part of the project is to analysis and clean the available data. When it comes to time series data of BTC prices, the data set was well processed and cleaned. There were no missing values nor any abnormal values in the data set. Thus, data cleaning was needed on the time series BTC prices.

However, the BTC tweet data set needed several preprocessing steps. Although [8] mentioned that the tweets are in English, quick glance through the data set verified that there were some tweets in other languages as well. Also, for sentiment analysis other preprocessing steps were required as well.

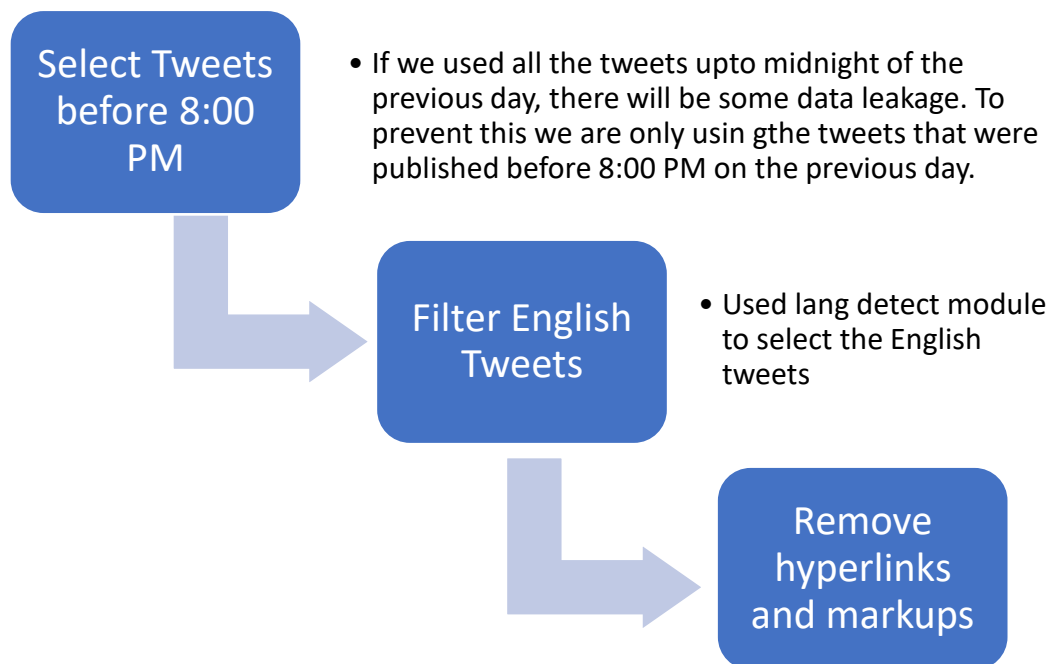


Figure 1: Preprocessing Tweets

Removing null values, non English tweets, and tweets after 8:00 PM is done in [13] and in [14]. After completing all the above processing on the original 16 million tweet data set resulted in a new preprocessed dataset with around 10 million tweets.

## 6. Sentiment Analysis

Next step of the project is to predict the sentiment values of each tweet. There are several libraries that will analyse the sentiment values. In this project several of these libraries were

used and for the final project used the VADER model. More details about these methods are given below.

- TextBlob

TextBlob is one of the simplest text processing libraries available for python [15]. When used for sentiment analysis it returns two results polarity and the subjectivity. Main advantage of textblob is the quick runtime. However, it does not capture the relationship between multiple words when analysing the sentiment values. Textblob provide a rule based sentiment analysing.

- Flair models

Flair models [16] also provide the sentiment analysis results for a given text. However, to call the sentiment analysis methods, flair models expect single sentences. So if the tweet consists of multiple sentences, it has to be split into multiple sentences. At the end, the average value of each sentence is taken as the sentiment of the tweet. Unlike textblob, flair used machine learning to detect the sentiment. It provides pretrained models that will be able to predict the sentiment of new data.

- VADER sentiment analysis

VADER of valence aware dictionary and sentiment reasoner is another rule-based sentiment analyser that is available in python [17]. One advantage of this analyser is that this is optimized for social media text.

- Hugging Face analysis

Hugging face is a state-of-the-art transformer based NLP library [18]. Out of the all the model that were tried in this project hugging face is the most advanced model. However, due to its complexity, there were some errors while processing over 10 million tweets.

The sentiment values obtained through different methods is compared in [19]. As shown in the following scatter plot most of the time the results obtained from variuos libraries matches with each other.

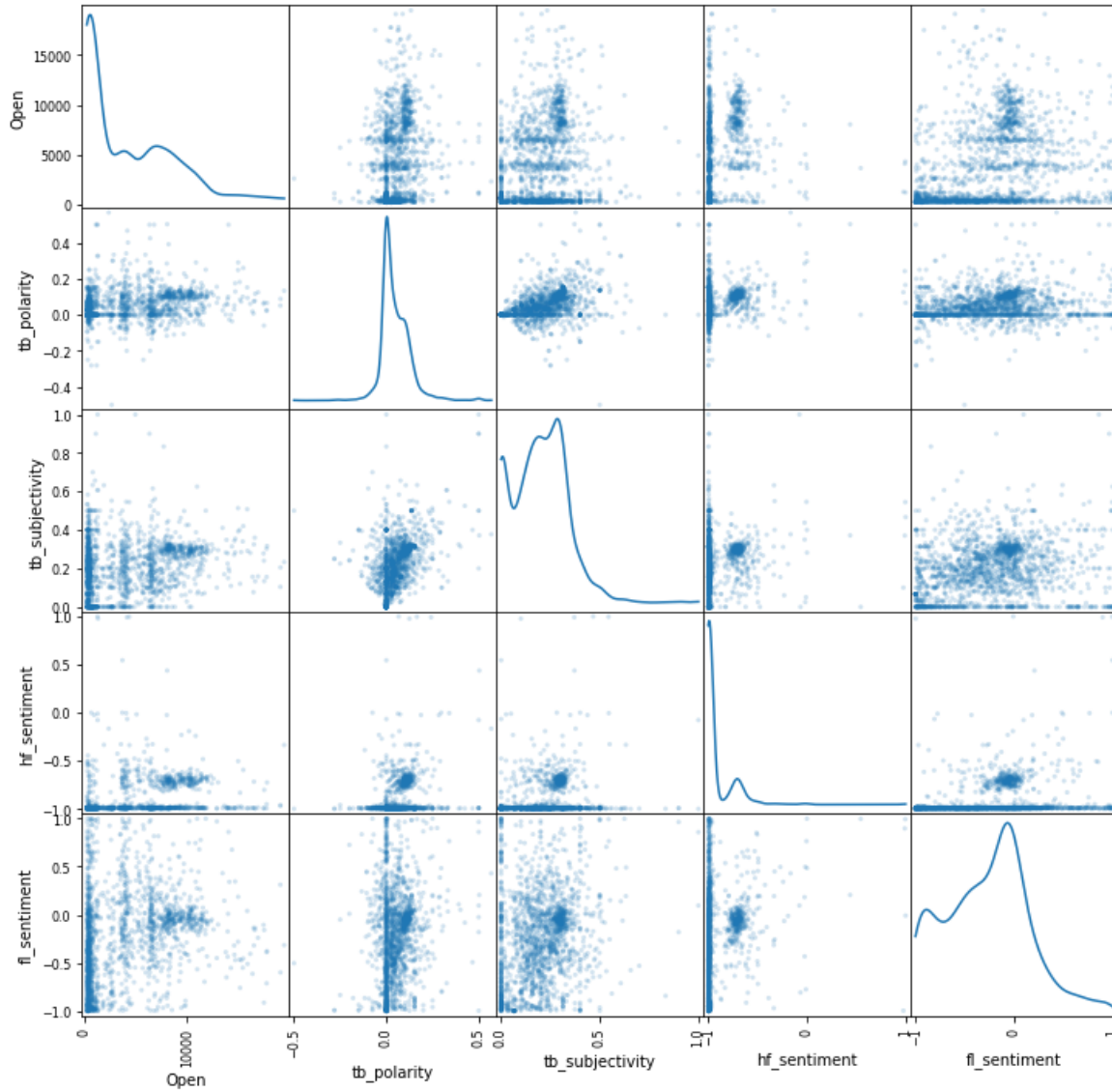


Figure 2: Compare sentiment values from different methods

## 7. Time series Analysis

This section presents the time series analysis that were performed on time series data. Several time series approaches have been tested in [20]. The full BTC price data set is divided in to test and training data sets in 80/20 split.



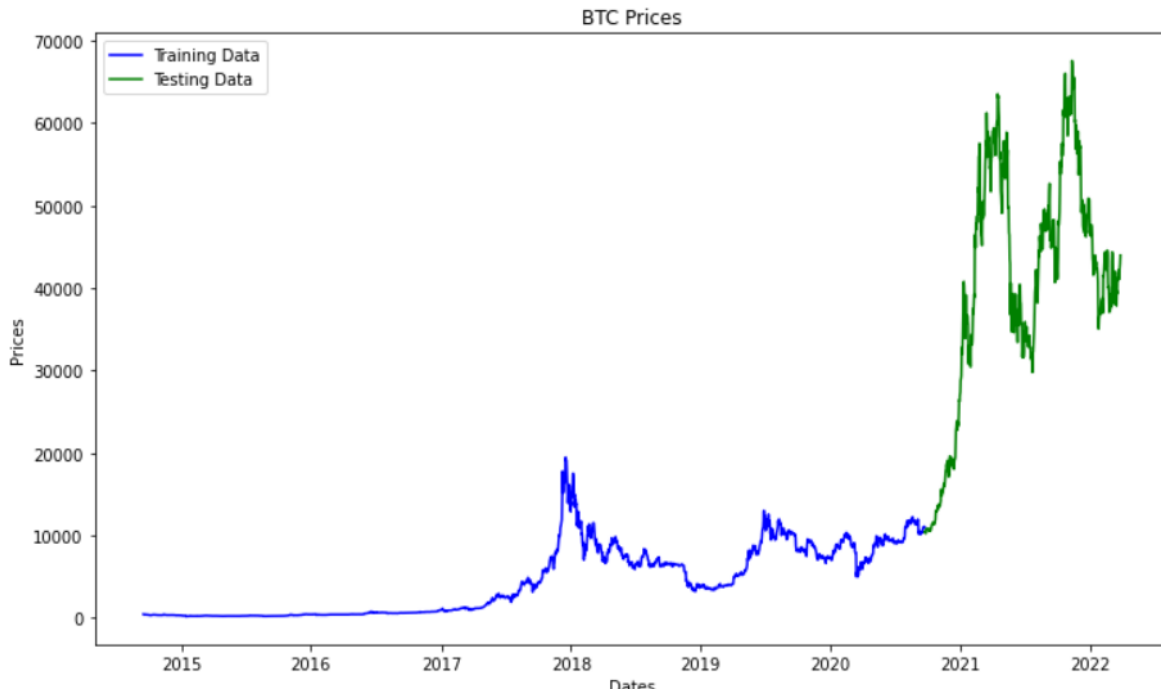


Figure 3: Training and testing data

For time series analysis following for models have been used.

- ARIMA Model

ARIMA stands for auto regressive and integrated moving average models. For ARIMA models three parameters are required namely  $p$ ,  $d$ , and  $q$ . For the given data we found out that  $p=5$ ,  $d=1$ , and  $q=0$  give the best results.

- Auto ARIMA

Auto ARIMA models will try to find the optimum  $p$ ,  $q$ , and  $d$  values. Thus, extensive search methods are not required to obtain these values. Since auto ARIMA can find the best model by itself, the results obtained through auto ARIMA outperformed ARIMMA models.

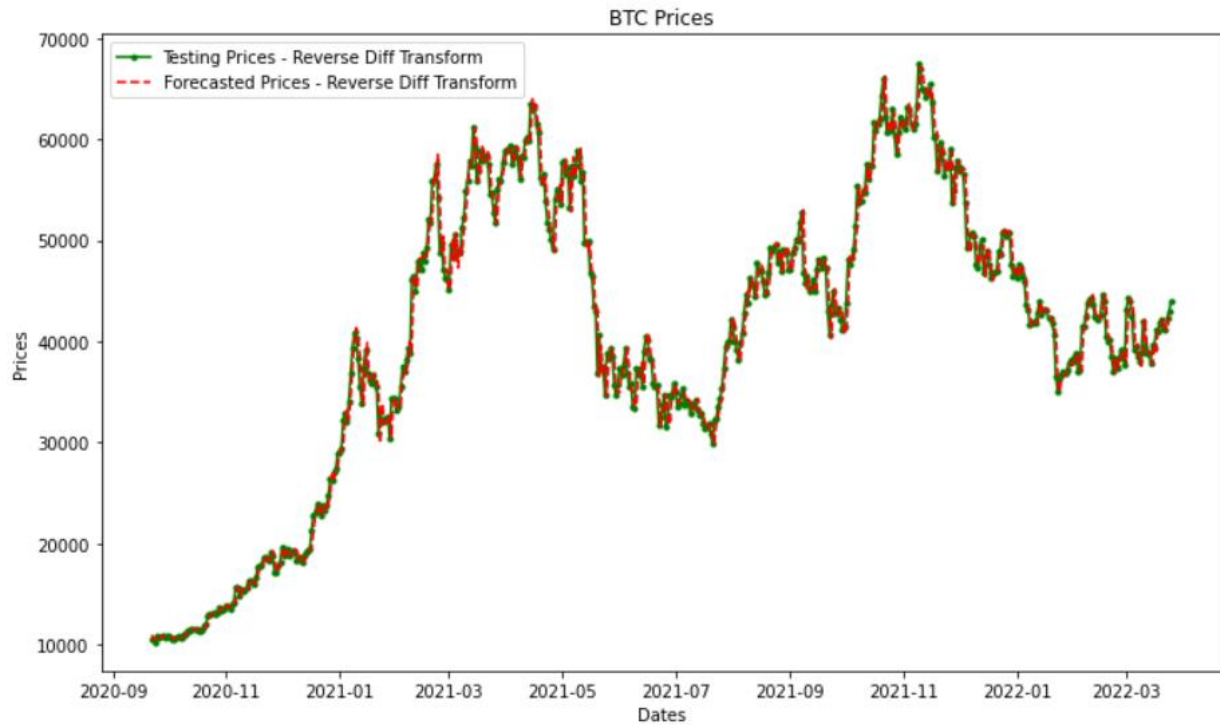


Figure 4: Auto ARIMA Results

- LSTM

LSTM is a deep learning model that can be used to predict time series data. For the LSTM model, the price of the last 60 days is used as the input and the price of BTC is predicted.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 64)	16896
dropout (Dropout)	(None, 60, 64)	0
lstm_1 (LSTM)	(None, 60, 64)	33024
dropout_1 (Dropout)	(None, 60, 64)	0
lstm_2 (LSTM)	(None, 64)	33024
dropout_2 (Dropout)	(None, 64)	0
dense (Dense)	(None, 1)	65

=====  
Total params: 83,009  
Trainable params: 83,009  
Non-trainable params: 0

Figure 5: LSTM model

- FB Prophet

Facebook prophet is a time series prediction library designed for R and python. Prophet has extensive set of libraries and methods to assist with time series predictions. Also it returns the confidentiality values for its predicted values. It supports various time series models such as ARIMA and also have the capability of using seasonality and other data such as weekends, holidays etc.

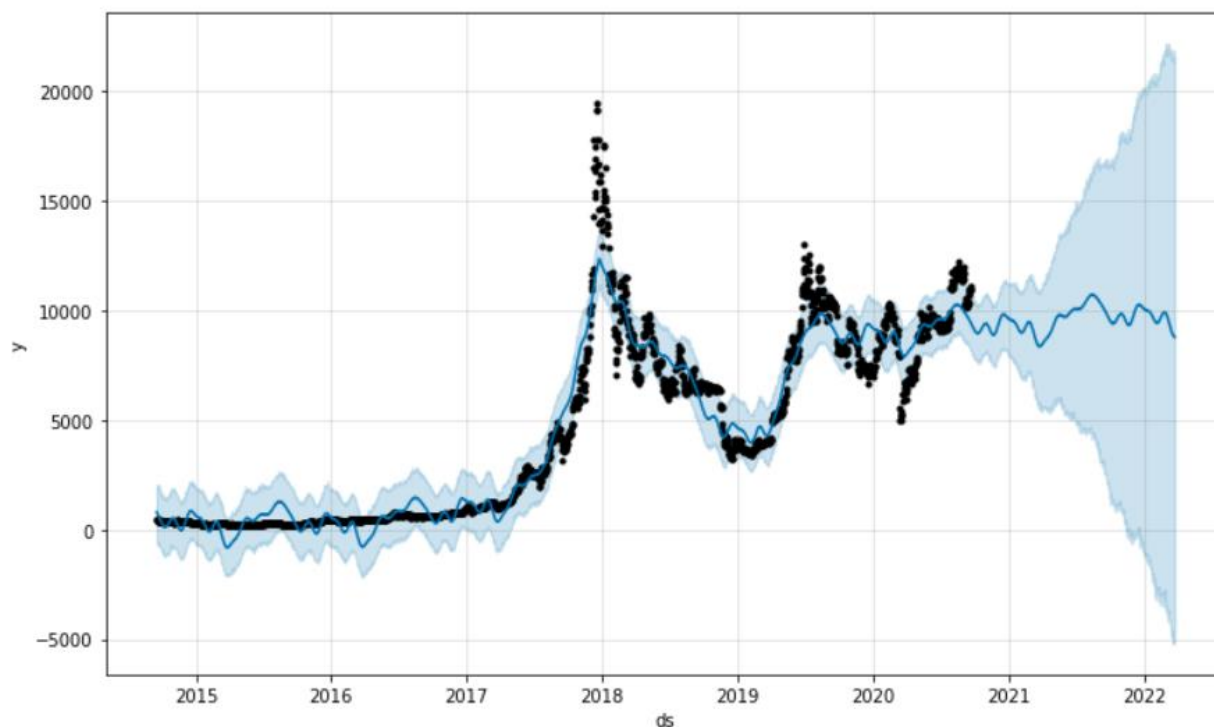


Figure 6: Predictions from FB prophet on the BTC data

Each of the above model was accessed with MSE and the SMAPE value for the predictions. Below is the summarized results for the time series analysis for various models. As seen below auto ARIMA model provided the best results. Also, it should be noted that the FB prophet had the lowest accuracy. This is because there was no retraining of the model after each new data point. Thus, the comparison of fb prophet with other models is not entirely fair.

	Method	MSE	SMAPE
0	ARIMA (5,1,0)	7.583865e+06	5.002120
1	Auto ARIMA	2.932842e+06	2.914987
2	LSTM	2.411039e+07	8.727481
3	Prophet	1.176769e+09	NaN

Figure 7: MSE and SMAPE for various models

## 8. Time series analysis with average sentiment

Next step of the project is to use both time series and sentiment values for the prediction of the BTC price. For this task, the average sentiment values were obtained for each day [21]. This is

documented in [22]. Apart from the BTC time series following seven features were also explored.

- Lagged number of tweets
- Lagged average VADER sentiment
- Lagged average textblob polarity
- Lagged average textblob subjectivity
- Lagged average number of likes
- Lagged average number of replies
- Lagged average number of retweets

Then an extensive search was done on the possibility of all the combinations of above features.

## 9. Deployment

### Final model

Through extensive analysis of various models and use of various feature that were available, it was discovered that using the lagged number of tweets, lagged average sentiment value of bitcoin tweets obtained through VADER analysis, and the lagged average number of replies to tweets are the best exogenous variables to combine with the time series of bitcoin price. A simple LSTM model with the time series of the bitcoin for the last 60 days plus the above-mentioned features provided the best MSE and the mean absolute percentage error.

Features	MSE	SMAPE
lag_num_tweets.lag_likes.lag_retweets	213409.443074	4.486820
lag_num_tweets.lag_vd_sentiment.lag_tb_polarit...	241400.554407	4.690073
lag_num_tweets.lag_vd_sentiment.lag_replies	228524.703160	4.709140
lag_num_tweets.lag_vd_sentiment.lag_tb_polarit...	241110.839878	4.725779
lag_likes	257267.145546	4.729280

Figure 8: Best feature combination

Model: "sequential\_277"

Layer (type)	Output Shape	Param #
lstm_554 (LSTM)	(None, 63, 64)	16896
dropout_554 (Dropout)	(None, 63, 64)	0
lstm_555 (LSTM)	(None, 64)	33024
dropout_555 (Dropout)	(None, 64)	0
dense_277 (Dense)	(None, 1)	65

=====  
Total params: 49,985  
Trainable params: 49,985  
Non-trainable params: 0

Figure 9: LSTM Model

## Deployment

For the deployment of the project, the LSTM model was saved as a .h5 file and the scalers for time series, lagged number of tweets, lagged average sentiment value, and lagged average replies were saved as pickle file. A Flask app was designed to output the predicted value when an appropriate API call is instantiated. The Flask server and the possible API call is shown in the following figures.

```

1 import requests
2 #url = 'http://localhost:5000/api'
3 url='http://ec2-107-21-156-198.compute-1.amazonaws.com:5000/api'
4 r = requests.post(url,json={'last_60_days':[6386.129883, 6413.629883, 6411.759766, 6373.189941, 6351.240234,
5      5736.149902, 5645.319824, 5578.580078, 5559.740234, 5620.779785,
6      4863.930176, 4465.540039, 4611.569824, 4360.700195, 4347.689941,
7      3880.780029, 4015.070068, 3765.949951, 3822.469971, 4269.004395,
8      4289.088867, 4024.464355, 4200.733398, 4147.32373 , 3886.294922,
9      3958.894775, 3754.074463, 3512.590332, 3421.9104 , 3473.227539,
10     3612.046387, 3497.554688, 3421.458252, 3487.879395, 3311.751953,
11     3243.997559, 3236.274658, 3253.123047, 3544.761475, 3706.824951,
12     3742.195068, 4133.703613, 3898.08374 , 4020.994629, 4000.331787,
13     4081.030518, 3819.666748, 3854.688477, 3653.131836, 3932.491699,
14     3822.384766, 3866.839111, 3746.713379, 3849.216309, 3931.048584,
15     3832.040039, 3851.973877, 3836.519043, 4078.584961,
16     4028.472168], 'num_tweets':599, 'vd_sentiment':.085170450751252, 'replies':0.9499165275459098})
17 print(r.json())

```

Figure 10: Sample file to access the API

```

1  # Import Libraries
2  import numpy as np
3  import pandas as pd
4  from flask import Flask, request, jsonify
5  from tensorflow.keras.models import load_model
6  import pickle
7
8  app = Flask(__name__)
9  # Load the model
10 save_path = 'model.h5'
11 ## Load tensorflow model
12 model = load_model(save_path)
13 #Load scalers for other features
14 main_scaler=pickle.load(open('mainScaler.pkl','rb'))
15 lag_num_tweets_scaler=pickle.load(open('lag_num_tweets_scaler.pkl','rb'))
16 lag_vd_sentiment_scaler=pickle.load(open('lag_vd_sentiment_scaler.pkl','rb'))
17 lag_replies_scaler=pickle.load(open('lag_replies_scaler.pkl','rb'))
18
19
20 @app.route('/')
21 def home_endpoint():
22     return 'Welcome to BTC price prediction. Please call api method with the required features!'
23
24 @app.route('/api',methods=['POST'])
25 def predict():
26     # Get the data from the POST request.
27     data = request.get_json(force=True)
28     # Extract indivsual features
29     X_test = np.array(data['last_60_days'])
30     num_tweets = np.array(data['num_tweets'])
31     vd_sentiment = np.array(data['vd_sentiment'])
32     replies = np.array(data['replies'])
33
34     #Do scaling
35     X_test=main_scaler.transform(X_test.reshape(-1, 1))
36     X_test = np.reshape(X_test, (1,60,1))
37     num_tweets=lag_num_tweets_scaler.transform(num_tweets.reshape(-1, 1))
38     vd_sentiment=lag_vd_sentiment_scaler.transform(vd_sentiment.reshape(-1, 1))
39     replies=lag_replies_scaler.transform(replies.reshape(-1, 1))
40     X_test=np.append(X_test,[num_tweets,vd_sentiment,replies])
41     X_test = np.reshape(X_test, (1,63,1))
42
43     # Make prediction using model Loaded from disk as per the data.
44     prediction = model.predict(X_test)
45     #Inverse transform
46     ans=main_scaler.inverse_transform(prediction[0][0].reshape(1, -1))
47     return jsonify(f'Predicted BTC price is {str(ans[0][0])} USD.')
48
49 if __name__ == '__main__':
50     app.run(port=5000, debug=False)

```

Figure 11: Flask app to predict the value



## Deployment on EC2

For deployment AWS EC2 servers were selected due to its freely available resources, well formulated documentation, and technical support articles online, and the suitability for this type of a project.

Since this project utilized TensorFlow, when selecting a machine instance, “Deep Learning AMI GPU TensorFlow 2.10.0 (Ubuntu 20.04)” instance was selected. This instance supported TensorFlow with GPU optimization.

Instance summary for i-0832df332e9d29cd7 (webApp2) <a href="#">Info</a>		
Updated less than a minute ago		
Instance ID i-0832df332e9d29cd7 (webApp2)	Public IPv4 address 107.21.156.198   <a href="#">open address</a>	Private IPv4 addresses 172.31.18.37
IPv6 address -	Instance state <span>Running</span>	Public IPv4 DNS ec2-107-21-156-198.compute-1.amazonaws.com   <a href="#">open address</a>
Hostname type IP name: ip-172-31-18-37.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-18-37.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address 107.21.156.198 [Public IP]	VPC ID vpc-0902a9e352c0f133b	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a> <a href="#">Learn more</a>
IAM Role -	Subnet ID subnet-04213891e61f7b7af	Auto Scaling Group name -

Figure 12: AWS EC2 Instance









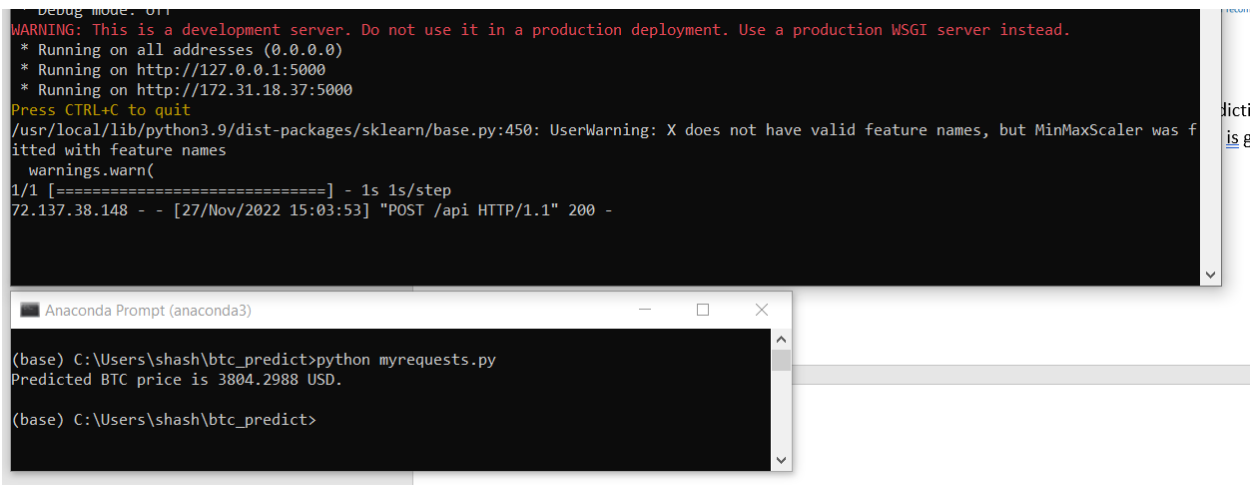
btc_predic				
Download Edit Properties New				
/home/ubuntu/btc_predict/				
Name	Size	Changed	Rights	Owner
		2022-11-26 10:14:32 PM	rw-r--r--	ubuntu
 lag_num_tweets_scaler.pkl	1 KB	2022-11-26 7:01:31 PM	rw-rw-r--	ubuntu
 lag_replies_scaler.pkl	1 KB	2022-11-26 7:01:43 PM	rw-rw-r--	ubuntu
 lag_vd_sentiment_scaler.pkl	1 KB	2022-11-26 7:01:51 PM	rw-rw-r--	ubuntu
 mainScaler.pkl	1 KB	2022-11-26 7:01:28 PM	rw-rw-r--	ubuntu
 mlApp.py	2 KB	2022-11-26 10:18:21 PM	rw-rw-r--	ubuntu
 model.h5	434 KB	2022-11-26 7:02:00 PM	rw-rw-r--	ubuntu
 myrequests.py	2 KB	2022-11-26 8:10:41 PM	rw-rw-r--	ubuntu

Figure 13: Files on the AWS EC2

After installing the required packages and doing all the imports, the bitcoin price prediction app was deployed on the above AWS instance. Some screenshots of the usage of this app is given below.



```
Debug mode: ON
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.31.18.37:5000
Press CTRL+C to quit
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
1/1 [=====] - 1s 1s/step
72.137.38.148 - - [27/Nov/2022 15:03:53] "POST /api HTTP/1.1" 200 -

Anaconda Prompt (anaconda3)
(base) C:\Users\shash\btc_predict>python myrequests.py
Predicted BTC price is 3804.2988 USD.

(base) C:\Users\shash\btc_predict>
```

Figure 14: Invoking the API through local machine

## Results

The model of the application development is given in the following diagram.

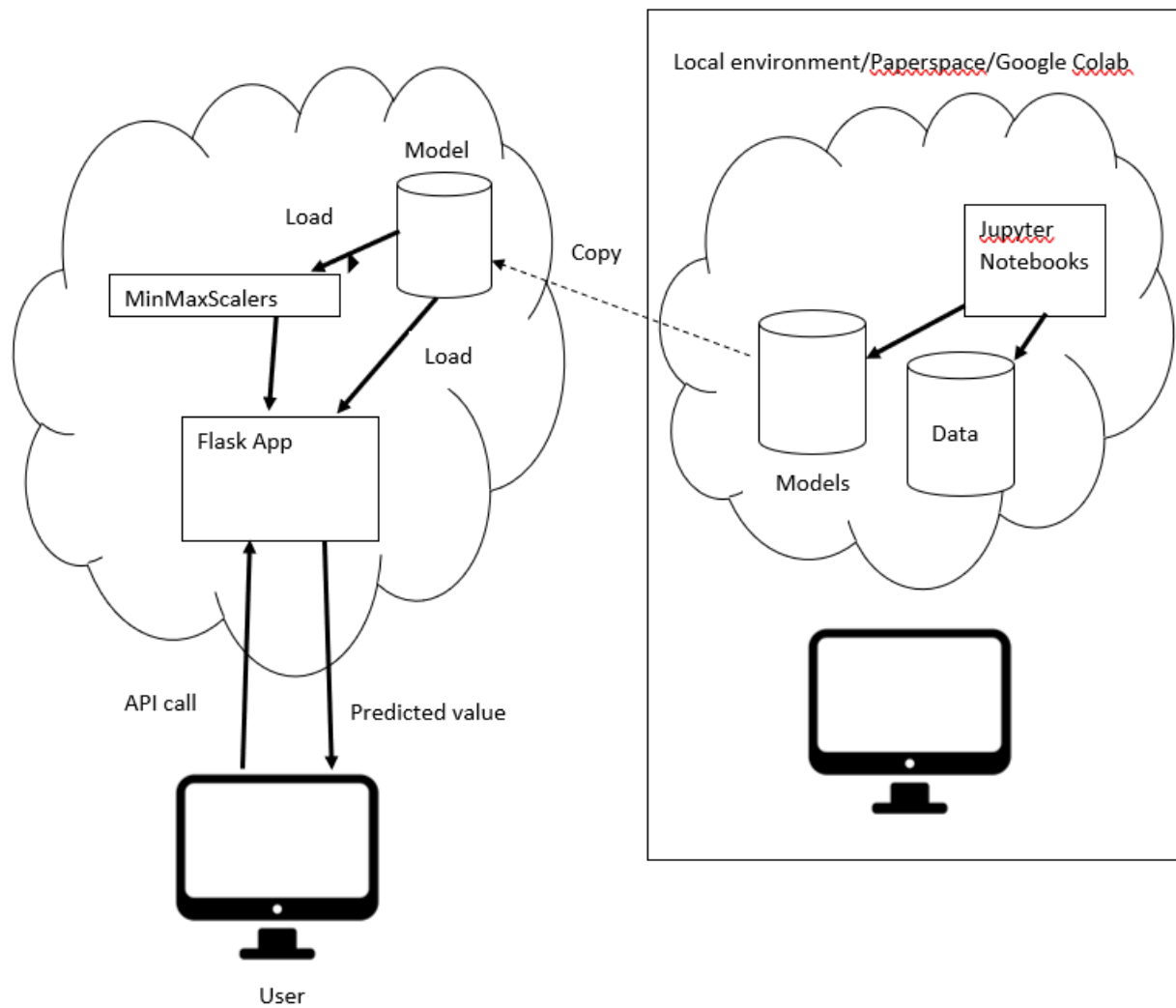


Figure 15: Model for the final application

## 10. Conclusion

## 11. References

[1 [Online]. Available: <https://en.wikipedia.org/wiki/Bitcoin>.

]

[2 [Online]. Available: <https://en.wikipedia.org/wiki/Cryptocurrency>.

]

[3 J. Abraham, D. Higdon, J. Nelson and J. and Ibarra, "Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis," *SMU Data Science Review*, vol. 1, no. 3, 2018.

- [4 O. & J. H. & O. R. & L. J. Sattarov, "Forecasting Bitcoin Price Fluctuation by Twitter  
] Sentiment Analysis," 2020.
- [5 S. Choi and O. J., "Twitter Attribute Classification With Q-Learning on Bitcoin Price  
] Prediction," *IEEE Access*, vol. 10, pp. 96136-96148, 2022.
- [6 J. G. A. & E. J. Critien, "Bitcoin price change and trend prediction through twitter sentiment  
] and data volume," *Financ Innov*, vol. 8, no. 45, 2022.
- [7 [Online]. Available: [https://www.kaggle.com/datasets/varpit94/bitcoin-data-updated-till-  
\] 26jun2021](https://www.kaggle.com/datasets/varpit94/bitcoin-data-updated-till-26jun2021).
- [8 [Online]. Available: [https://www.kaggle.com/datasets/alaix14/bitcoin-tweets-20160101-to-  
\] 20190329](https://www.kaggle.com/datasets/alaix14/bitcoin-tweets-20160101-to-20190329).
- [9 [Online]. Available: <https://github.com/twintproject/twint>.  
]
- [1 [Online]. Available: <https://github.com/tweepy/tweepy>.  
0]
- [1 [Online]. Available:  
1] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/ce8ec887eca6ef0ef843a  
380678bee072f696d83/Data%20Wrangling/Twint\\_setup\\_2.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/ce8ec887eca6ef0ef843a380678bee072f696d83/Data%20Wrangling/Twint_setup_2.ipynb).
- [1 [Online]. Available: <https://medium.com/p/68b162c7d132>.  
2]
- [1 [Online]. Available:  
3] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/e5bae31517884dc1bf0d  
1a7ea37a3a9c3d1a2019/Data%20Wrangling/RemoveNULLsAndNOnEnglishTweets.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/e5bae31517884dc1bf0d1a7ea37a3a9c3d1a2019/Data%20Wrangling/RemoveNULLsAndNOnEnglishTweets.ipynb).
- [1 [Online]. Available:  
4] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/edd9e2f720279df6e95d  
ea7055c2392f0f023d56/Data%20Wrangling/DataProcessing.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/edd9e2f720279df6e95dea7055c2392f0f023d56/Data%20Wrangling/DataProcessing.ipynb).
- [1 [Online]. Available: <https://textblob.readthedocs.io/en/dev/>.  
5]
- [1 [Online]. Available: <https://github.com/flairNLP/flair>.  
6]

[1 C. & G. E. Hutto, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI, 2014.

[1 [Online]. Available: <https://huggingface.co/>.  
8]

[1 [Online]. Available:  
9] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/9cd51129f585e9f2bc5a5ebdc48547a788641377/Data%20Wrangling/EDA\\_with\\_sentiments\\_and\\_price.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/9cd51129f585e9f2bc5a5ebdc48547a788641377/Data%20Wrangling/EDA_with_sentiments_and_price.ipynb).

[2 [Online]. Available:  
0] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/main/code/BTC\\_Time\\_Series\\_Forecasting.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/main/code/BTC_Time_Series_Forecasting.ipynb).

[2 [Online]. Available:  
1] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/89ecd8b3e0e4c9ca47ad0b24facd77e06ebf0279/code/CompleteTweetSentimentAddition.ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/89ecd8b3e0e4c9ca47ad0b24facd77e06ebf0279/code/CompleteTweetSentimentAddition.ipynb).

[2 [Online]. Available:  
2] [https://github.com/shashi3876/Capstone\\_Project\\_UCSD\\_Ext/blob/50fb9af65fc6ae38b30f7c72df51cffee40c3c67/code/ARIMAX\\_with\\_BTC\\_TWEETS\\_Forecasting%20\(1\).ipynb](https://github.com/shashi3876/Capstone_Project_UCSD_Ext/blob/50fb9af65fc6ae38b30f7c72df51cffee40c3c67/code/ARIMAX_with_BTC_TWEETS_Forecasting%20(1).ipynb).