

**Predictive Analytics**

***Bitcoin Price Prediction using Machine Learning***

***Team A***

*Gauravi Bhalchandra Patil* [*gbp5152@psu.edu*](mailto:gbp5152@psu.edu)

*Vineeta Peddinti* [*vkp5111@psu.edu*](mailto:vkp5111@psu.edu)

*Nishanth Kadapakonda* [*npk5242@psu.edu*](mailto:npk5242@psu.edu)

*Harsh Anand* [*hpa5116@psu.edu*](mailto:hpa5116@psu.edu) *(Team Lead)*

**School of Graduate Professional Studies**

Data Analytics

IE-575 – Predictive Analytics

Spring, 2019

# Document Control

## Work carried out by

|  |  |  |
| --- | --- | --- |
| **Name** | **Email Address** | **Task description** |
| Gauravi Bhalchandra Patil | [gbp5152@psu.edu](mailto:gbp5152@psu.edu) | Data understand and preprocessing for Python, Modeling LSTM and GRN on Python Google Collaboration |
| Vineeta Peddinti | [vkp5111@psu.edu](mailto:vkp5111@psu.edu) | Data understand and preprocessing for Python, Modeling ARIMA and SARIMAX in R |
| Nishanth Kadapakonda | [npk5242@psu.edu](mailto:npk5242@psu.edu) | Data understand and preprocessing for Python, Modeling ARIMA and SARIMAX in R |
| Harsh Anand | [hpa5116@psu.edu](mailto:hpa5116@psu.edu) | Data understanding, Modeling LSTM and GRN on Google Collaboration on Python Google Collaboration |

## Revision Sheet

|  |  |
| --- | --- |
| **Date** | **Revision Description** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Contents

[Document Control 2](#_Toc34055643)

[Work carried out by 2](#_Toc34055644)

[Revision Sheet 2](#_Toc34055645)

[1. Introduction 4](#_Toc34055646)

[2. Problem Statement 4](#_Toc34055647)

[2.1 Importance 4](#_Toc34055648)

[3. Data 5](#_Toc34055649)

[3.1 Data Description 5](#_Toc34055650)

[3.2 Data Preprocessing 7](#_Toc34055651)

[4. Methodology 8](#_Toc34055652)

[4.1 ARIMA and SARIMA Model 11](#_Toc34055653)

[4.2 LSTM 16](#_Toc34055654)

[4.3 GRU 16](#_Toc34055655)

[5. Results 17](#_Toc34055656)

[5.1 Discussion of Results 17](#_Toc34055657)

[6. References 17](#_Toc34055658)

# Introduction

Bitcoin is a cryptocurrency that isn’t managed by way of a financial institution or company however transactions are recorded inside the blockchain this is public and contains records of every transaction that takes place. Bitcoin was first invented in 2009 via an anonymous founder called Satoshi Nakamoto but it became extremely popular in 2017. Bitcoins are moved in blocks every 10 minutes on a decentralized ledger that connects blocks into a coherent chain dating lower back to the first genesis block. It turned into originally described as peer-to-peer digital cash, but the era has evolved to emphasize being an agreement layer rather than a charging network. Some experts call bitcoin "the currency of the future" or even direct it as an illustration of the social revolution. It has remained the biggest cryptocurrency utilizing marketplace cap.

The bitcoin price is volatile and has increased several times during the 2017 year. The price of bitcoin does not depend on the business events or intervening government authorities, unlike the stock market, therefore, it is challenging for existing or potential investors and for government structures to accurately predict the bitcoin price. To understand the dynamic economies of scales it is important to identify daily changes in the bitcoin market to obtain a correlation between the external features and use machine learning techniques to predict the closest possible trend and price. This makes the demand for Bitcoin price prediction mechanism to be high.

# Problem Statement

As the Bitcoin price movement can contribute significantly to the preparation of a consistent economic policy, there could be a wide range of analysis and prediction possibilities that could be carried out on the available open-source bitcoin dataset. Our project is focused to explore the following diaspora in the area of prediction of the bitcoin price value.

1. Understand the price value trend, seasonality of the bitcoin price movement
2. Explore different deep artificial intelligence techniques such as Deep learning, and time series analysis techniques to predict the price value for the future
3. Compare different models and propose the best model for further exploration and analysis on the real-time dataset.

## 2.1 Importance

Even though bitcoin has been trading for almost 10 years, regulation is still in its very early days. Given that bitcoin and other cryptocurrency prices fluctuate in accordance with fast-paced technological developments, as well as economic, security and political factors, the policymakers need to understand the core dynamics to make the right policy before it can affect the global market. Bloomberg predicts that the increasing global uncertainties and a weak dollar will likely push more investors into bitcoin as it becomes recognized as a store of value. The cryptocurrency's fixed supply will further drive price increases throughout the year and will affect the normal dynamic of the market. Based on the prediction result, the economist and the business can take appreciate steps for their benefit.

# Data

The predictive models for predicting the bitcoin price movement is created on the dataset published in Kaggle (<https://www.kaggle.com/mczielinski/bitcoin-historical-data/data>). The dataset includes the historical bitcoin market data at 1-min intervals for select bitcoin exchanges where trading takes place.

There are two datasets available from different exchanges -

1. “coinbaseUSD1-mindata2014-12-01to\_2019-01-09.csv”
2. “bitstampUSD1-mindata2012-01-01to\_2019-08-12.csv”.

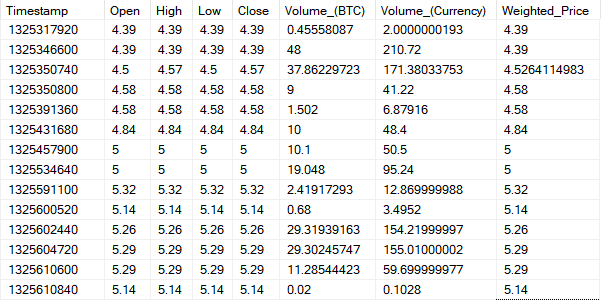
For this project, we have leveraged “bitstampUSD1-mindata2012-01-01to\_2019-08-12.csv”. The bit stamp data is for the time period of Jan 2012 to August 2019, with minute to minute updates of OHLC (Open, High, Low, Close), Volume in BTC and indicated currency, and weighted bitcoin price. Timestamps are in Unix time. Timestamps without any trades or activity have their data fields filled with “NaNs”. If a timestamp is missing, or if there are jumps, this may be because the exchange (or its API) was down, the exchange (or its API) did not exist, or some other unforeseen technical error in data reporting or gathering.

## 3.1 Data Description

The data consists of 8 columns and 3,997,697 rows. The sample data can be seen in the Figure 1. Table 1 shows the data description and the data type of all the columns.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Description** | **Type** |
| Timestamp | Start time of time window (60s window), in Unix time | UNIX Time, Integrer |
| Open | Open price at start time window | Numeric |
| High | High price within time window | Numeric |
| Low | High price within time window | Numeric |
| Close | Close price at end of time window | Numeric |
| Volume\_(BTC) | Amount of BTC transacted in time window | Numeric |
| Volume\_(Currency) | Amount of Currency transacted in time window | Numeric |
| Weighted\_Price | Volume-weighted average price (VWAP) | Numeric |

*Table 1: Data Dictionary*



*Figure 1: Snapshot of the sample data*

Following is the statistics after the initial exploration of the data –

1. Missing values – There are
2. Duplicate records – There are no duplicate timestamps or records in the dataset.

1. Skewness – Skewness check is not required in this dataset because of the point in time value of the data.
2. Normalization required? – As all the data is given at a point interval, normalization of the data is not required in this scenario.
3. Number of yearly records –

Below the number of records for each year that’s present in the data –

|  |  |
| --- | --- |
| **Year** | **#Records** |
| 2011 | 968 |
| 2012 | 527,040 |
| 2013 | 525,600 |
| 2014 | 525,600 |
| 2015 | 519,128 |
| 2016 | 527,040 |
| 2017 | 525,600 |
| 2018 | 525,600 |
| 2019 | 321,121 |

1. Number of yearly records with the open price at the start time window which as not “NaN”s –

|  |  |
| --- | --- |
| **Year** | **# Records** |
| 2011 | 4 |
| 2012 | 26,629 |
| 2013 | 319,761 |
| 2014 | 398,176 |
| 2015 | 372,735 |
| 2016 | 352,099 |
| 2017 | 483,332 |
| 2018 | 505,773 |
| 2019 | 307,310 |

These are the actual number of rows that will be used for the prediction of the values.

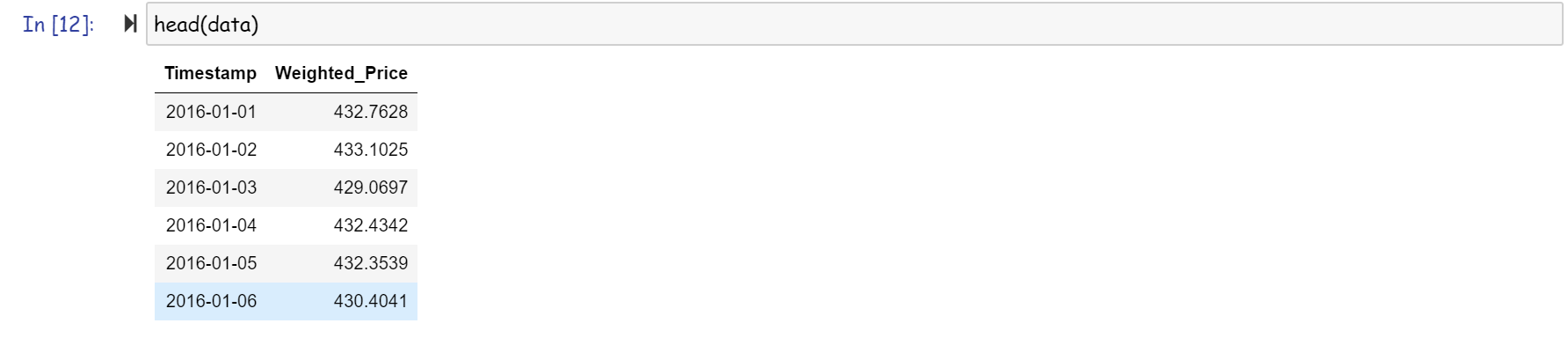
## 3.2 Data Preprocessing

The timestamp is Unix based and it is in integer format. Initial task was to convert the timestamp to a date format. We have converted it to the format “YYYY-MM-DD”.

As mentioned earlier, there are about 1,231,878 missing values in the weighted price column. To handle them, initially we have grouped all the transactions performed in a single day (since it is minute to minute transaction) and then replaced the null values in the weighted price for each day with the mean of the weighted price for that day.



We then performed a unique row check and filtered out all the unique columns.



For testing the models, we have considered the train data to be from “01-01-2016” to “01-05-2019” and the test data to be from “01-06-2019” to “12-08-2019”. There is a total of 1,217 rows in the train data and 73 rows in the test data.

For using the LSTM and GRU, we have used Min Max Scaling of the price variable and create a lookback function based on the below script for feeding into the model.

def create\_lookback(dataset, look\_back=1):

    X, Y = [], []

    for i in range(len(dataset) - look\_back):

        a = dataset[i:(i + look\_back), 0]

        X.append(a)

        Y.append(dataset[i + look\_back, 0])

    return np.array(X), np.array(Y)

# Methodology

We have chosen the four models to leverage the time-series capability of predicting the bit coin price movements. The models chosen are as follows –

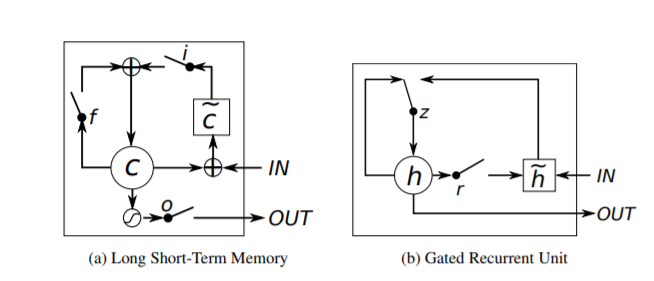
1. 2-layers long short term memory (LSTM) architecture of Recurrent neural network (RNN)
2. Gated Recurrent Unit (GRU) architecture of Recurrent neural network (RNN)
3. ARIMA
4. SARIMA

ARIMA and SARIMA are chosen due to its stochastic nature and robust way of handling the error terms and time lagging in the data points by ACF and PACF dimensions. Whereas LSTM and GRN are used because of their capability of learning long term dependencies. Also, LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically the default behavior of LSTM. Similarly, GRN make each recurrent unit to adaptively capture dependencies of different time scales. Similarly, to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell.

Figure 2 shows the core difference in the working of LSTM and GRN units, however, both comes under the family of Recurrent Neural Network.

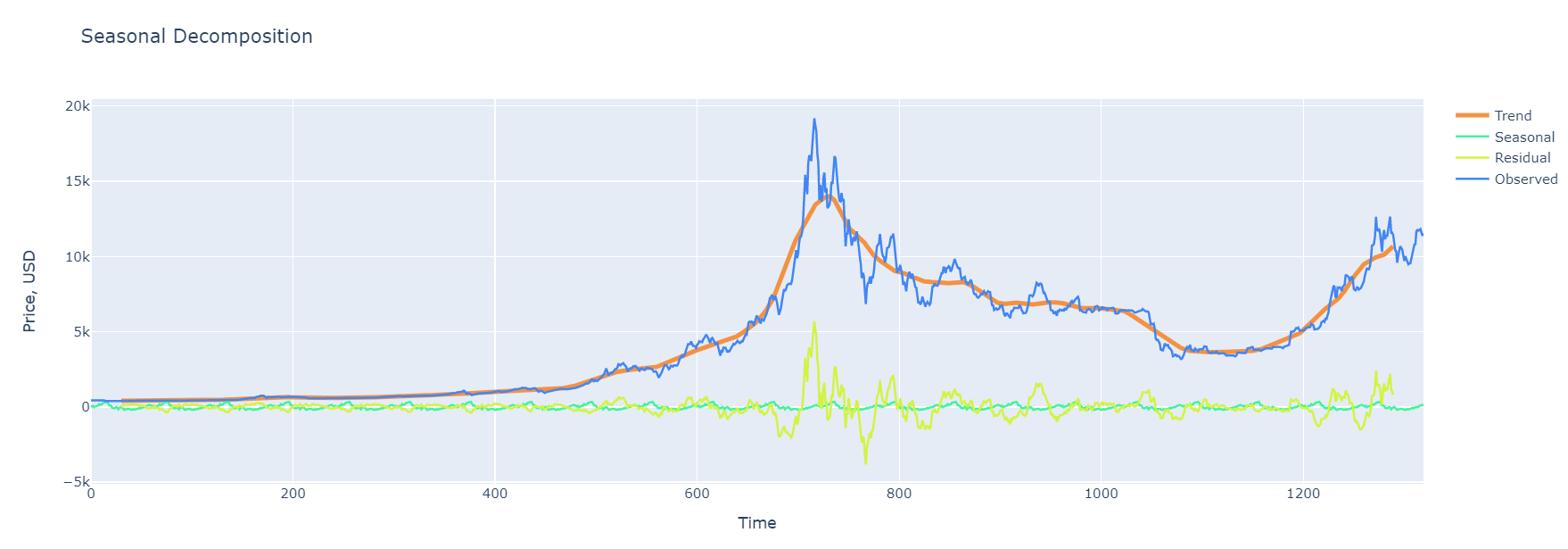
We followed the 5 step process for each algorithm to get the predicted result. The steps are as follows.

1. Collecting real-time cryptocurrency data and analyze the date (data exploratory analysis)
2. Check the stationary, seasonality and trend of the dataset
3. Prepare the data for Neural Network/ Machine Learning training
4. Model the Neural Network/ Machine Learning based on the prepared dataset
5. Test and visualize the prediction



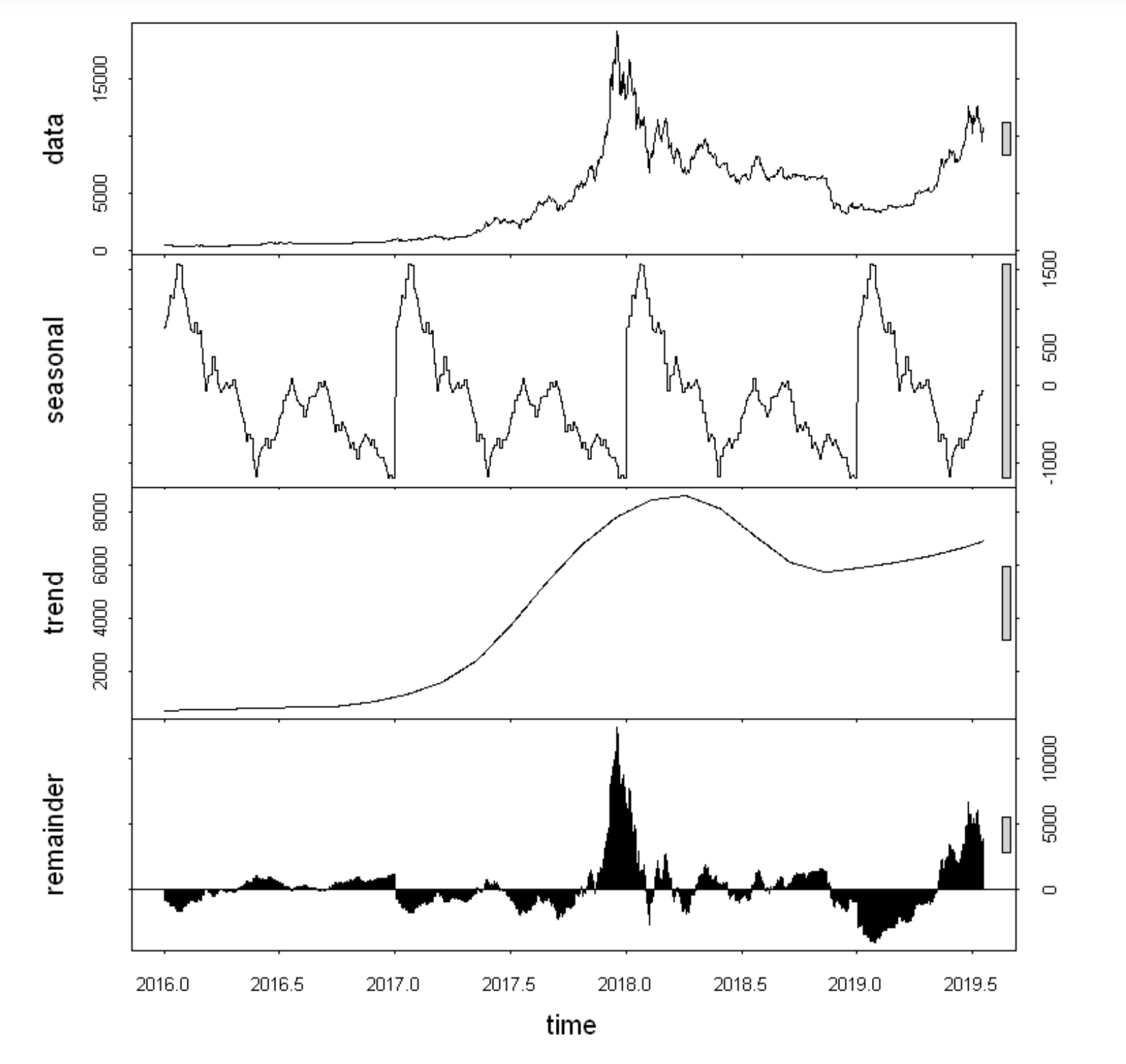
*Figure 2: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and c˜ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.*

The bitcoin data chosen is decomposed into seasonal, trend, remainder components to check for any seasonality or trend. If there is any seasonality or trend in the model, time series models don’t perform well there we first need to remove the data.



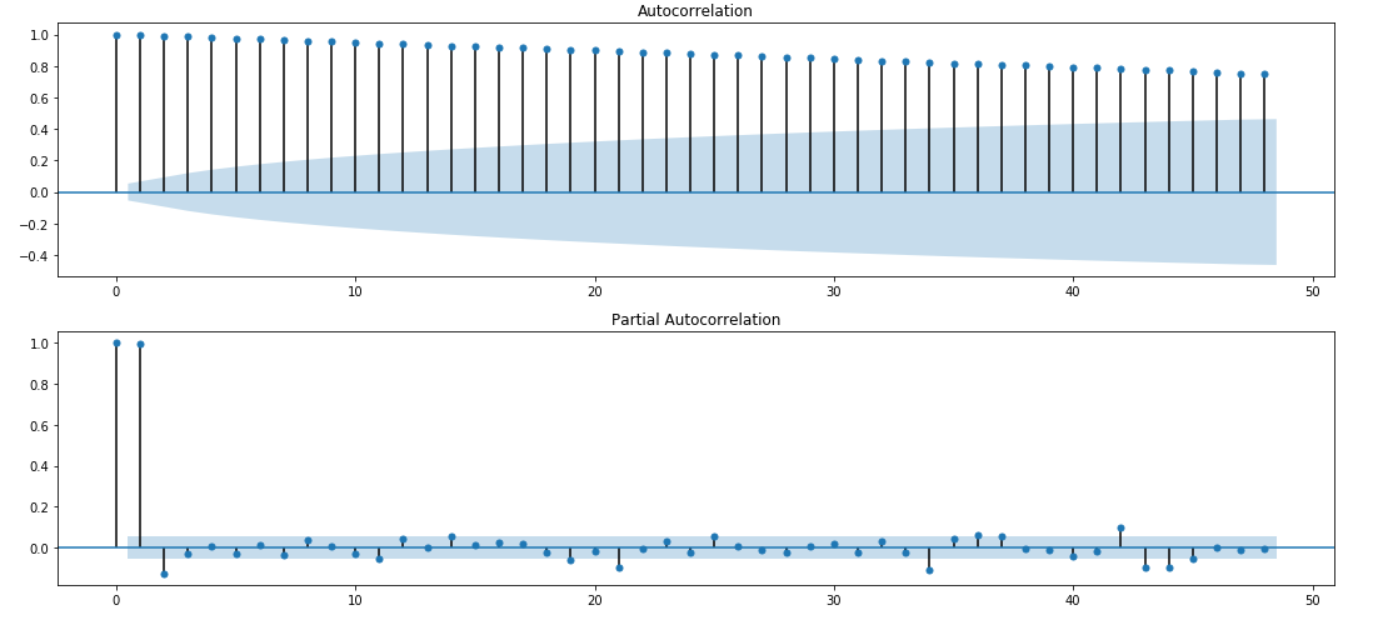
*Figure 3: Snapshot of seasonal decomposition*

By observing the time series plot in Figure 3, it is evident that there is seasonality in every month, prices are going high at a point in a month. There is no strict pattern that we can observe in the trend. We can also get a distinctive plot of all the variables as shown in Figure 4.



*Figure 4: Snapshot of individual decomposition of trend, seasonal and residual*

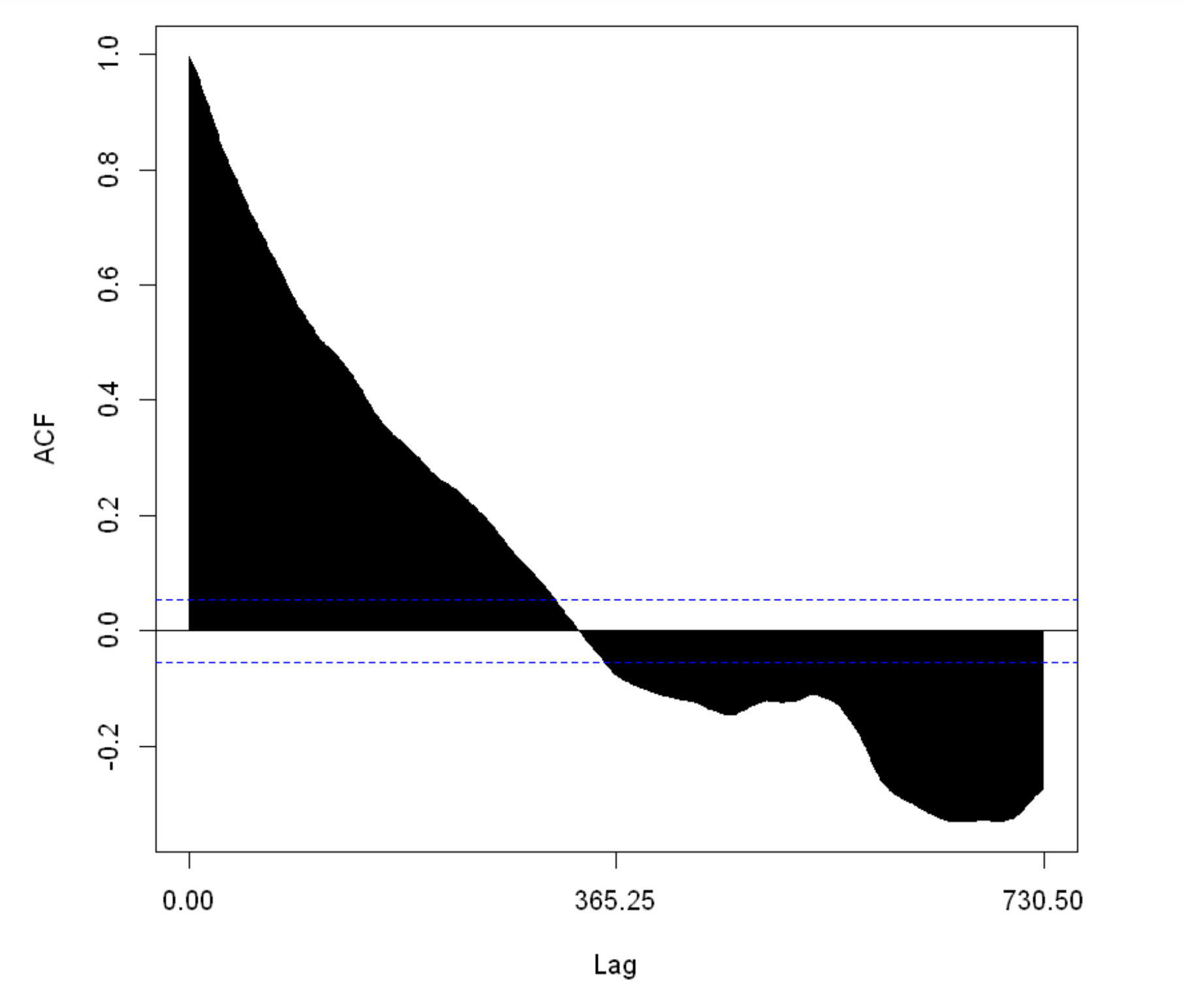
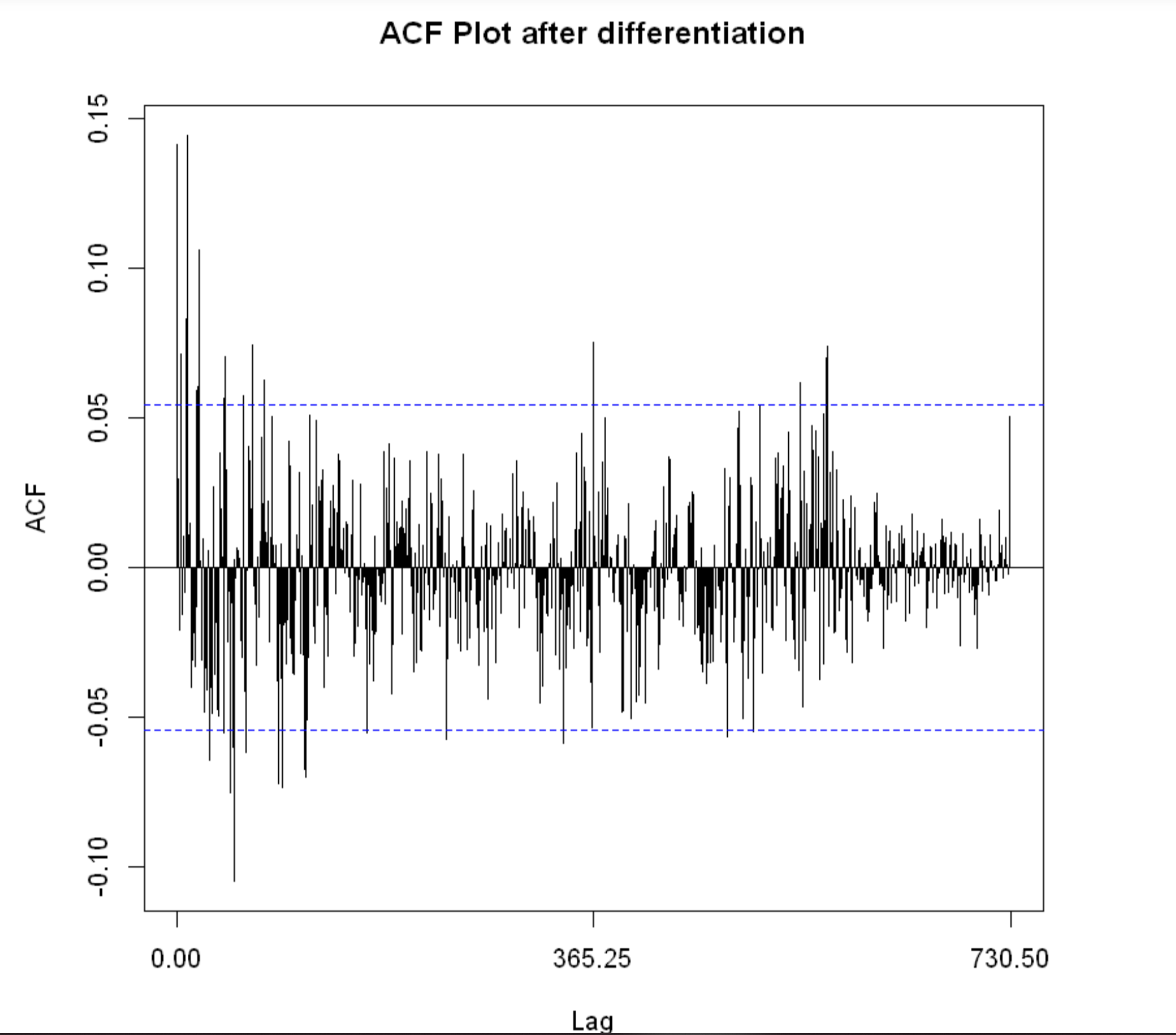
The next thing we do is the examination of the autocorrelation. It is it is the similarity between observations as a function of the time lag between them. It is important for finding repeating patterns in the data. The observed autocorrelation can be seen in Figure 5.



*Figure 5: Snapshot of autocorrelation and partial autocorrelation*

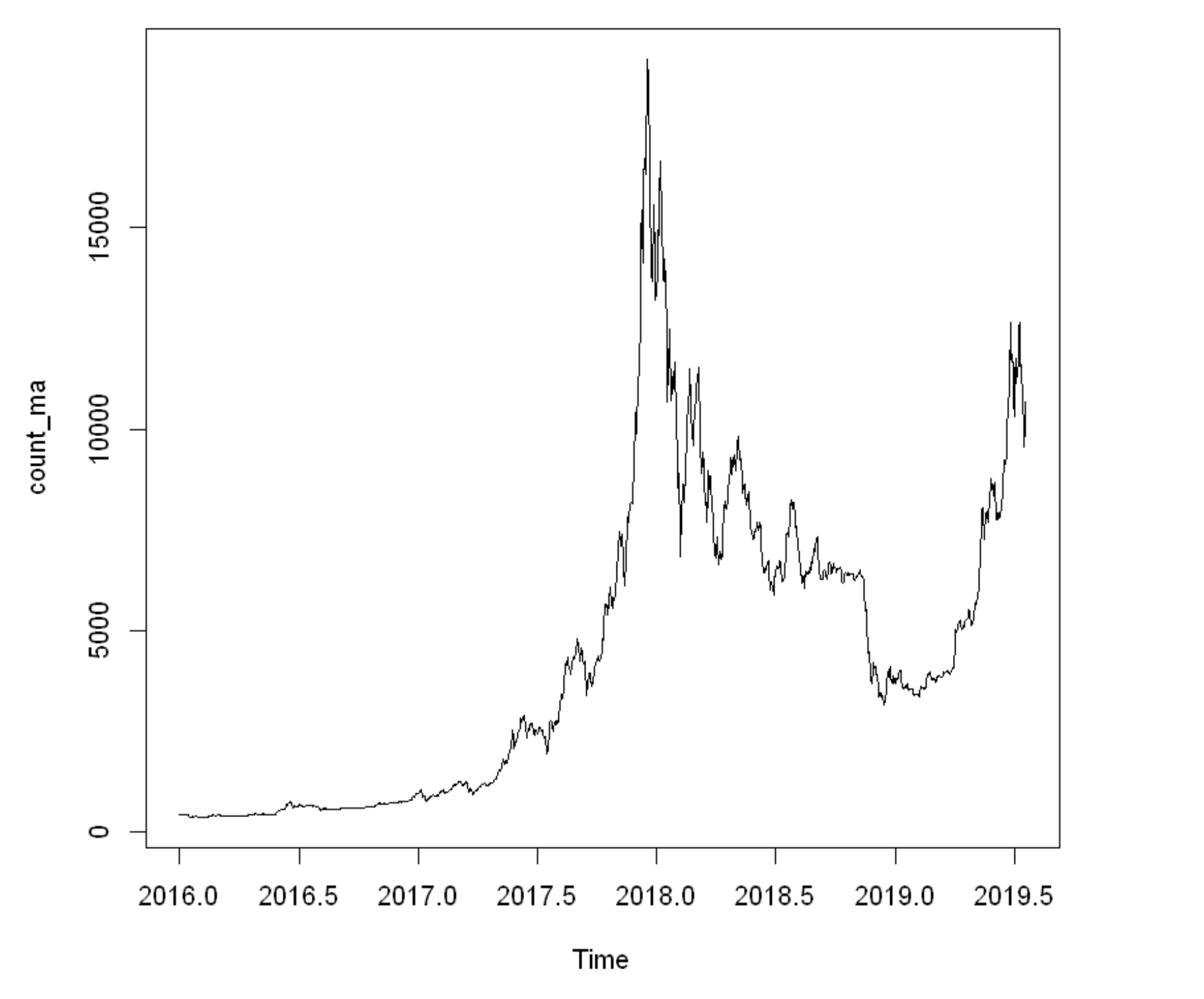
## 4.1 ARIMA and SARIMA Model

To apply the ARIMA and the SARIMA model, we first need to take out the seasonality and the trend that we can see in the figures above. To achieve the same, we perform a first order differentiation and the results can be seen in the Figure 6.



*Figure 6: Snapshot of first order differentiation*

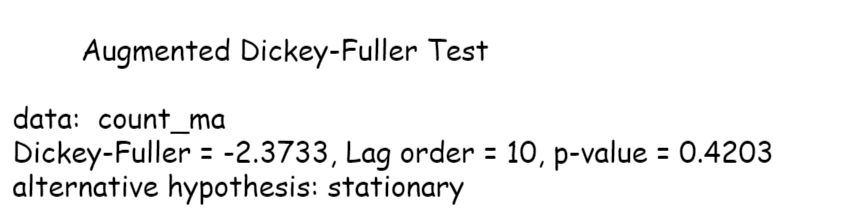
Now we preform moving average to remove the noise and the same can be seen in Figure 7. The plot shows the bit coin weighted process with a frequency of 1 month.



*Figure 7: Snapshot of moving average plot*

We then check for the stationary condition of the time series. To achieve the same, we perform Dickey-fuller test was performed to check whether the wave was stationary or not.

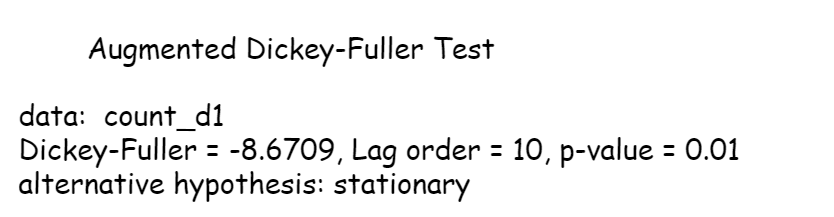
The output of the ADF test for the original data after seasonality is removed can be seen in the Figure 8.



*Figure 8: Snapshot of the ADF test before first order differentiation*

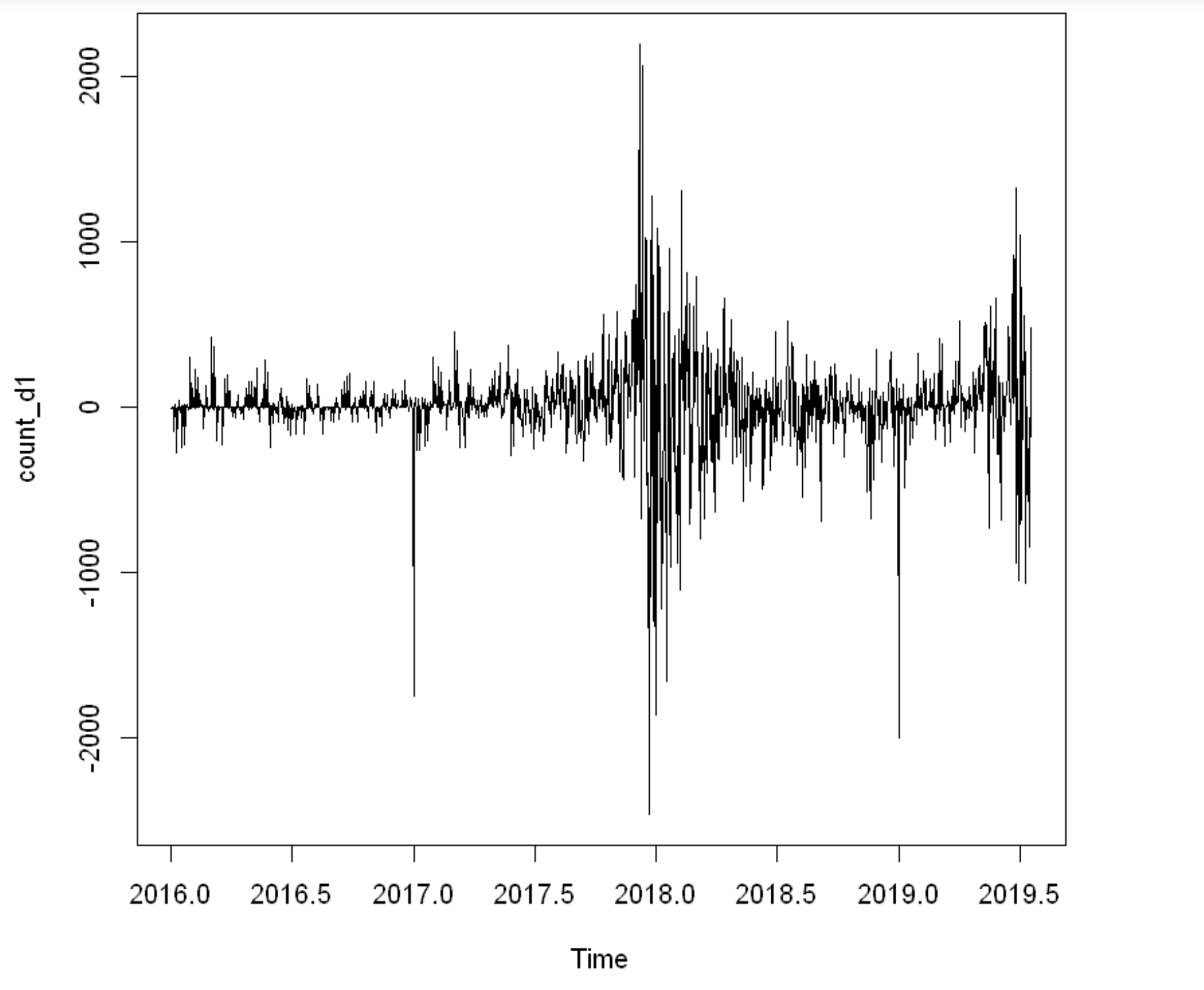
Since the P-value is greater than 0.05, we do not reject the null hypothesis that the wave is not stationary. Therefore, we performed a first order differentiation.

Now we perform the same ADF test after performing first order differentiation. The result can be found in the Figure 9.



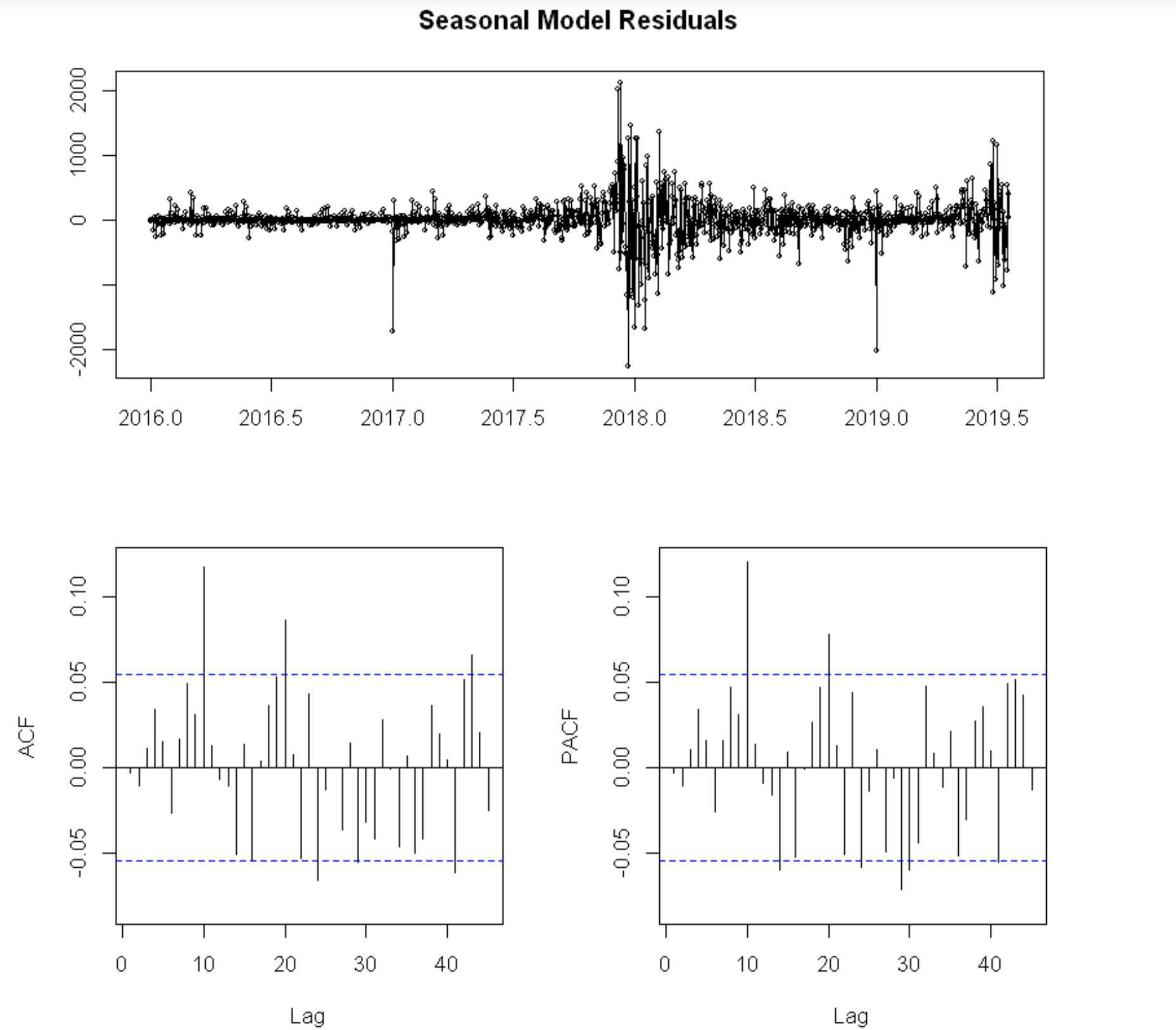
*Figure 9: Snapshot of the ADF test before first order differentiation*

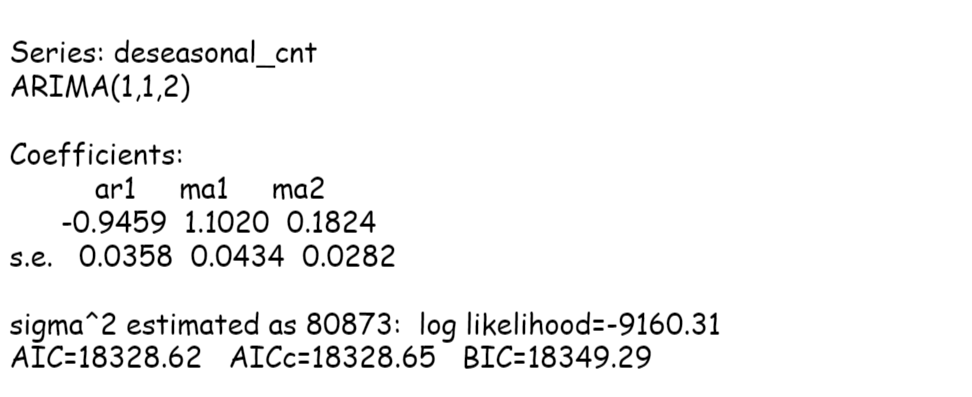
The P-value is significant, so we reject the null hypothesis. This means that now the wave is stationary. Figure 10 shows the stationary wave after the first order differentiation.



*Figure 10: Snapshot of the stationary wave after first order differentiation*

Next for the differentiated result, we have built an auto ARIMA model and the results can be seen in the Figure 11.

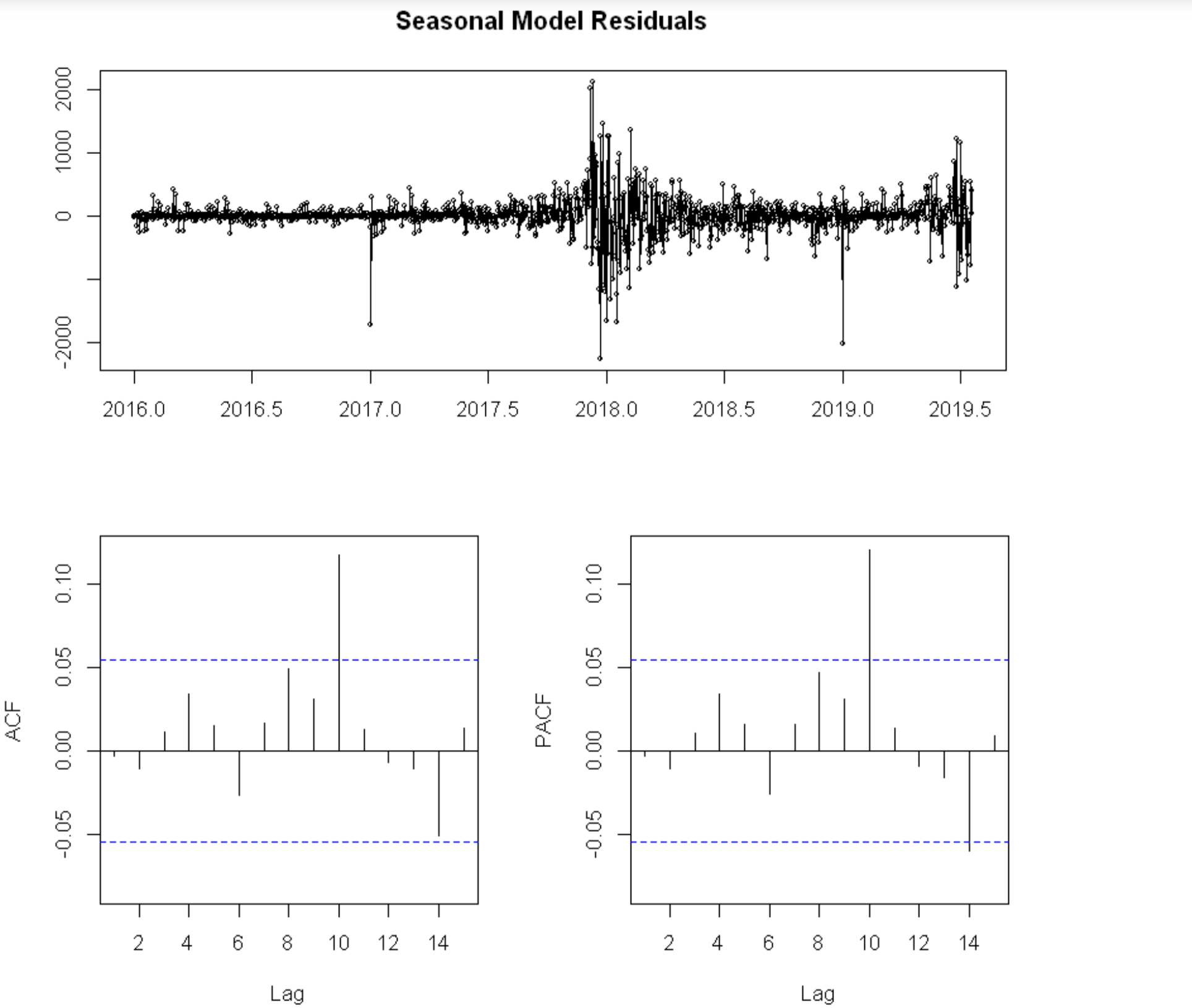


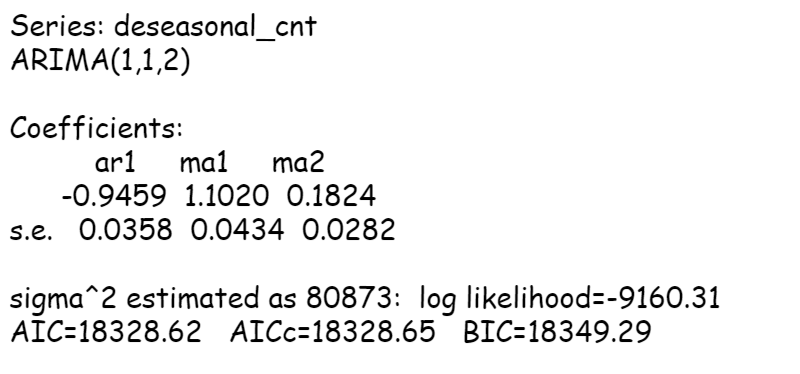


*Figure 11: Snapshot of the ARIMA Model result*

The Auto ARIMA model gave an AIC value of 16,815. Since the seasonality component was removed, we wanted to build a model with seasonality introduced in it since the original data showed seasonality.

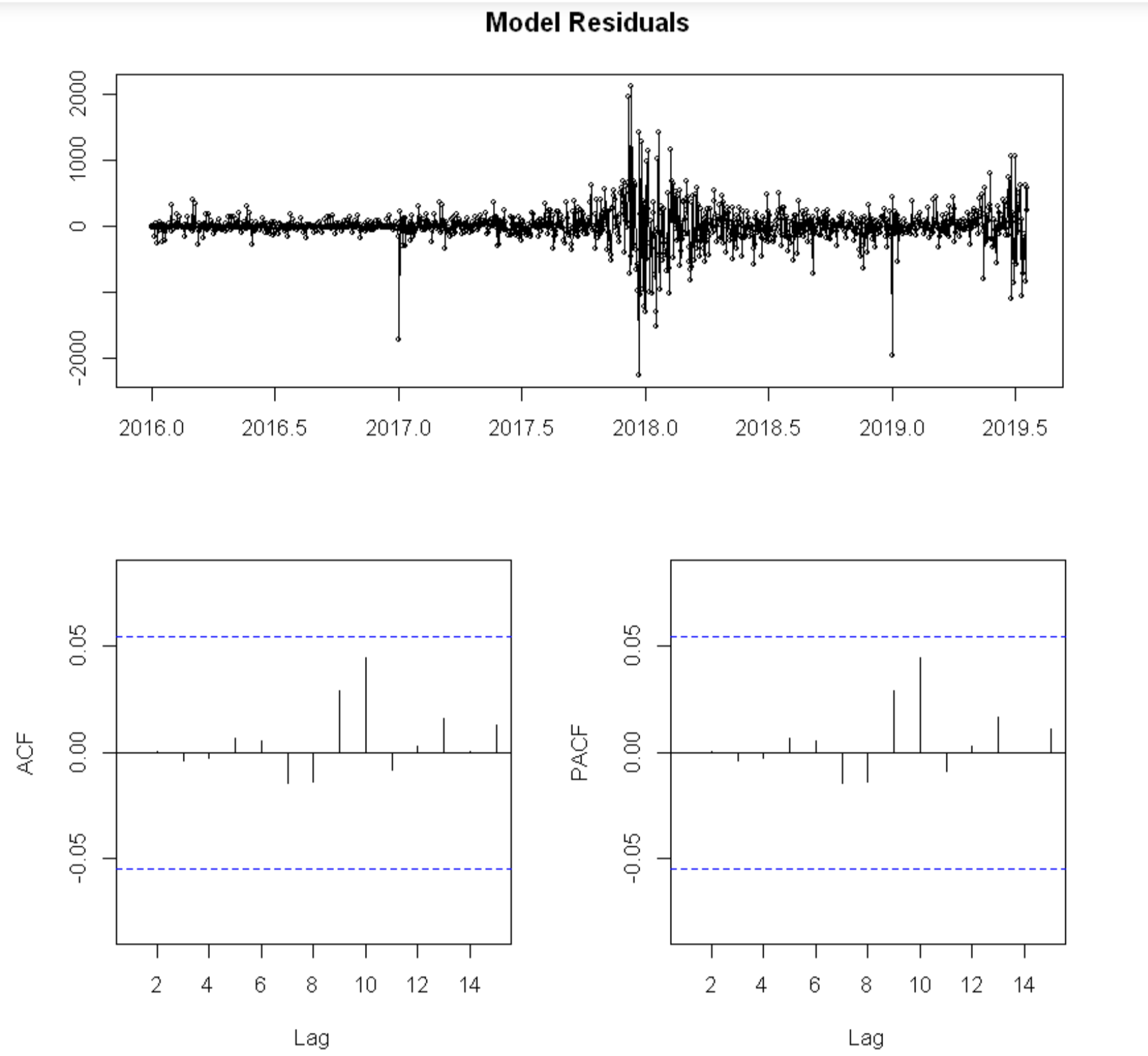
Now, we build the SARIMA model to include the seasonality of the dataset. The result for the SARIMA model can be seen in the Figure 12.

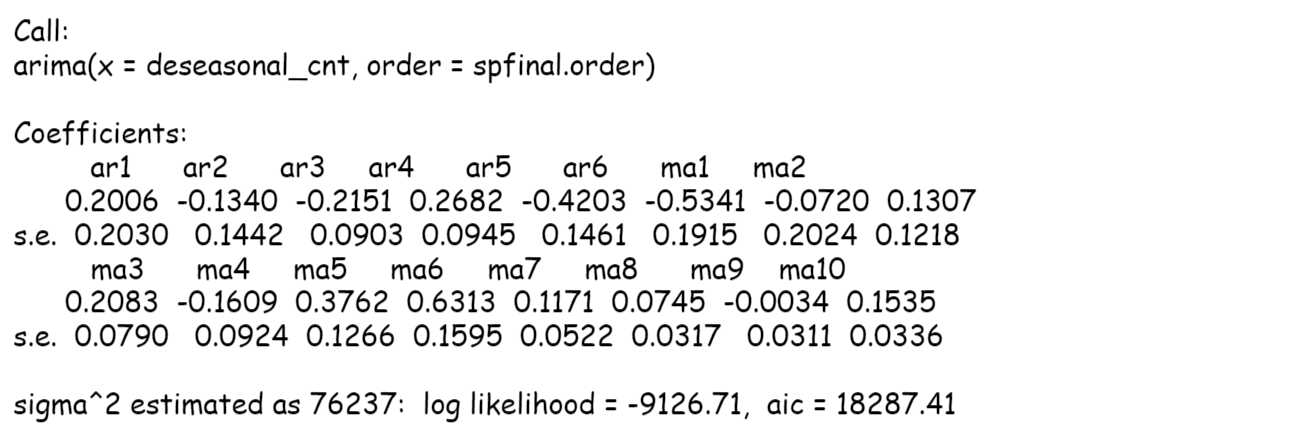




*Figure 12: Snapshot of the SARIMA Model result*

We wanted to check if there any other p,d,q that will get a better AIC value. We decided to check for the p,d,q values manually. We tried with p,d,q values with (8,1,6) with reference to the ACF and PACF plots. Following are the results:





*Figure 13: Snapshot of the SARIMA Model result with optimal p,d,q*

The AIC value got reduced from 16815 to 16737. So, we decided to predict the weighted prices using this model.

## LSTM

For training the model we leveraged Keras framework for deep learning. Our model consists of two stacked LSTM layers with 256 units each and the densely connected output layer with one neuron. We are using Adam optimizer and MSE as a loss. Also, we use an early stopping if the result doesn’t improve during 20 training iterations (epochs).

The below code was used to setup the environment for two stacked LSTM layer.

model = Sequential()

model.add(LSTM(256, return\_sequences=True, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(LSTM(256))

model.add(Dense(1))

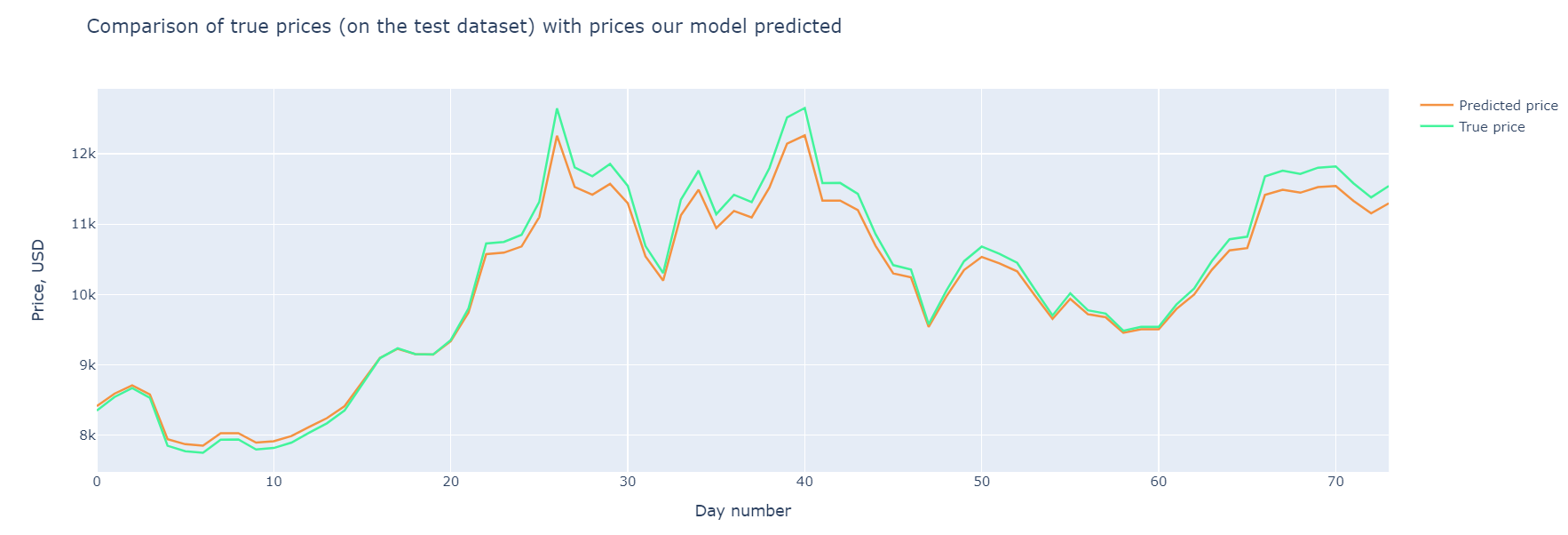
model.compile(loss='mean\_squared\_error', optimizer='adam')

Now we look at train and test loss of the model, and we could clearly see that with the increased number of epochs the loss has been decreasing. The same can be seen in the Figure 14.



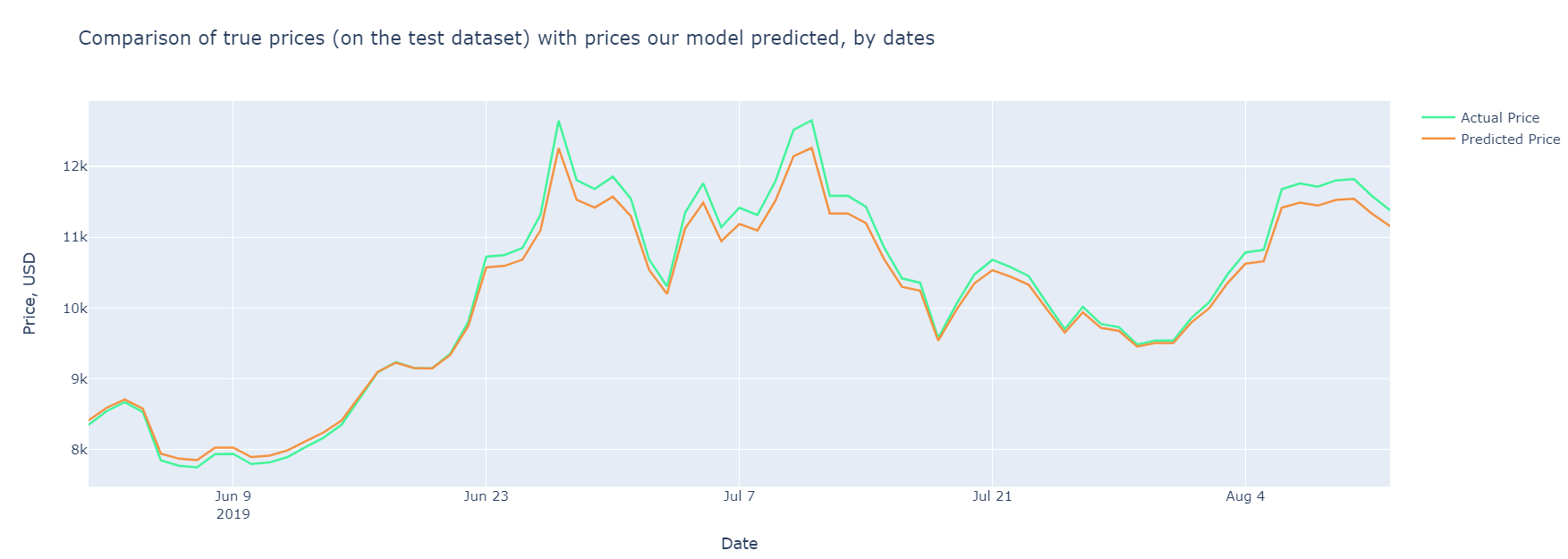
*Figure 14: Snapshot of the train and test loss of the model*

We now compare the true price and test prices with our model by day. The same can be seen in the Figure 15.



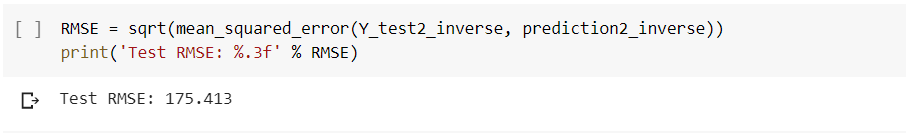
*Figure 15: Snapshot of the true price and test price by day*

We now compare the true price and test prices with our model by date. The same can be seen in the Figure 16.



*Figure 15: Snapshot of the true price and test price by date*

The root mean square error of the model is 175.413 and the output snapshot can be seen in the Figure 16.



*Figure 16: Snapshot of the Test RMSE for LSTM*

## GRU

To compare the output to the LSTM model, we performed the time series modeling using GRU. The below code was used to setup the environment for GRU.

model = Sequential()

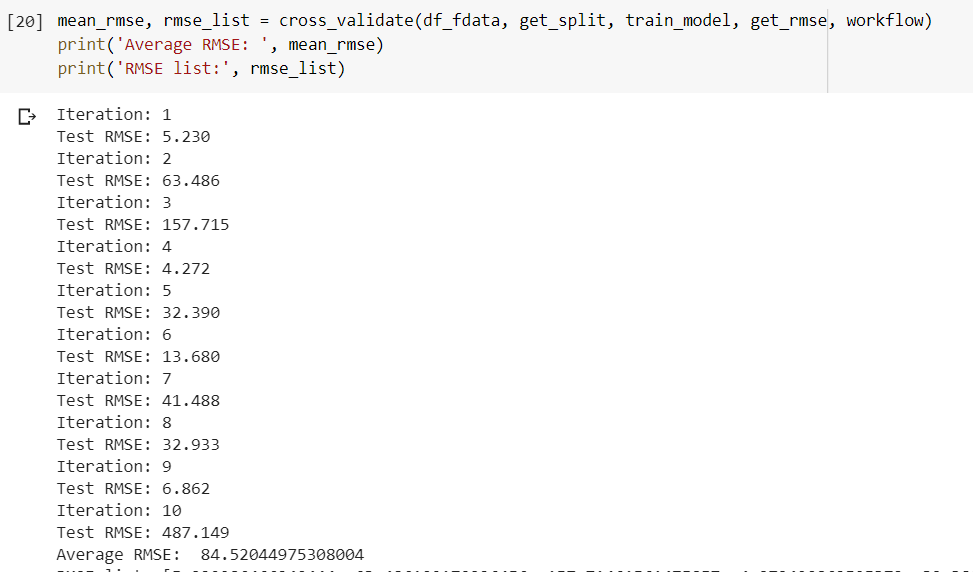
model.add(GRU(256, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

We have also used cross validation to get the optimal hyperparameters.

After cross validation, we could see the result of the RMSE after iteration as shown in Figure 17.



*Figure 17: Snapshot of the Test RMSE for GRU*

# Results

After considering the train data to be from “01-01-2016” to “01-05-2019” and the test data to be from “01-06-2019” to “12-08-2019”, the results for each of the model is below -

1. **LSTM**

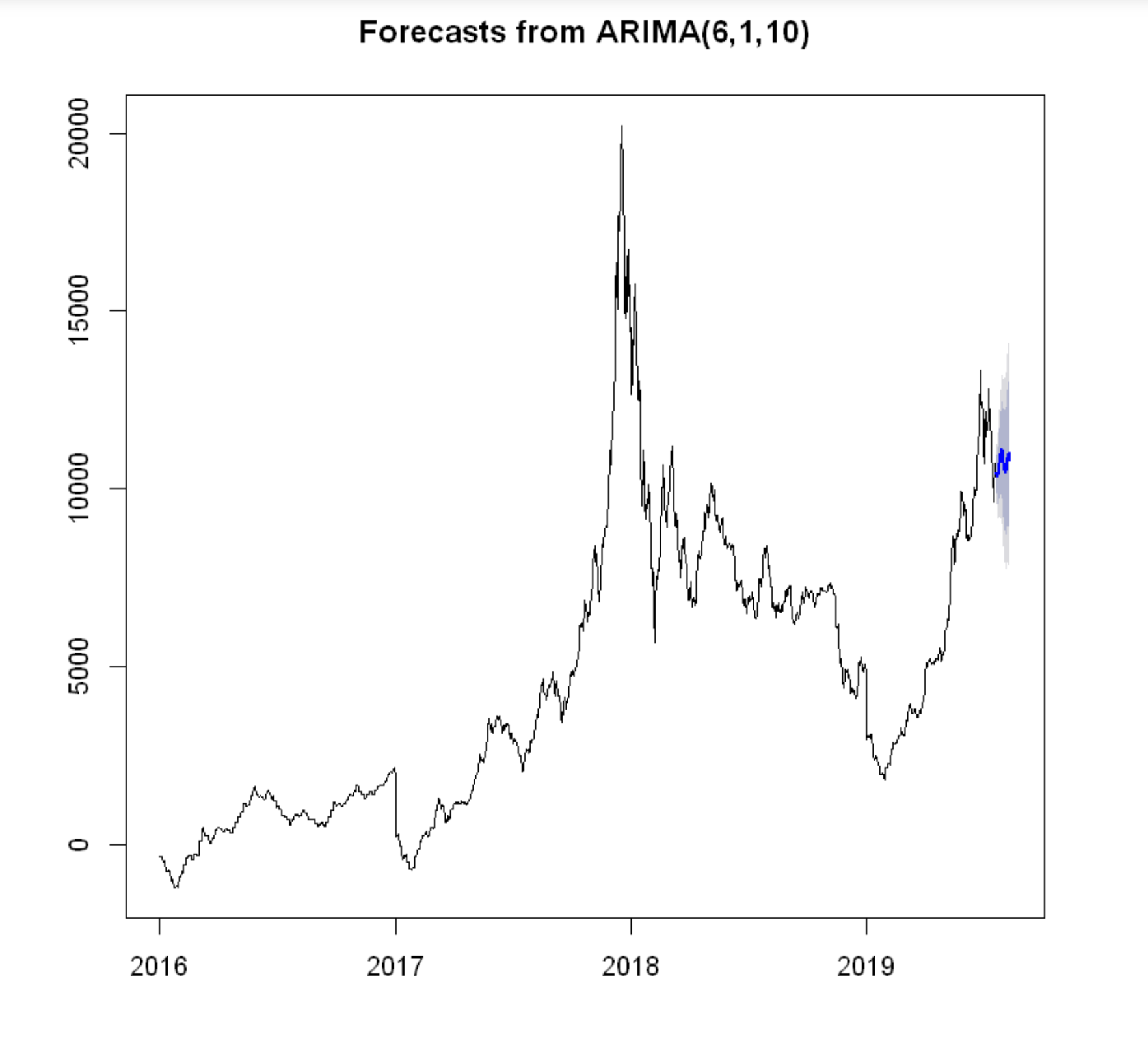
LSTM gives a root mean square error of the model is 175.413 and the output snapshot can be seen in the Figure 15 and Figure 16.

1. **GRN**

GRN gives the average root mean square error of the model is 84.41 and the output snapshot can be seen in the Figure 17.

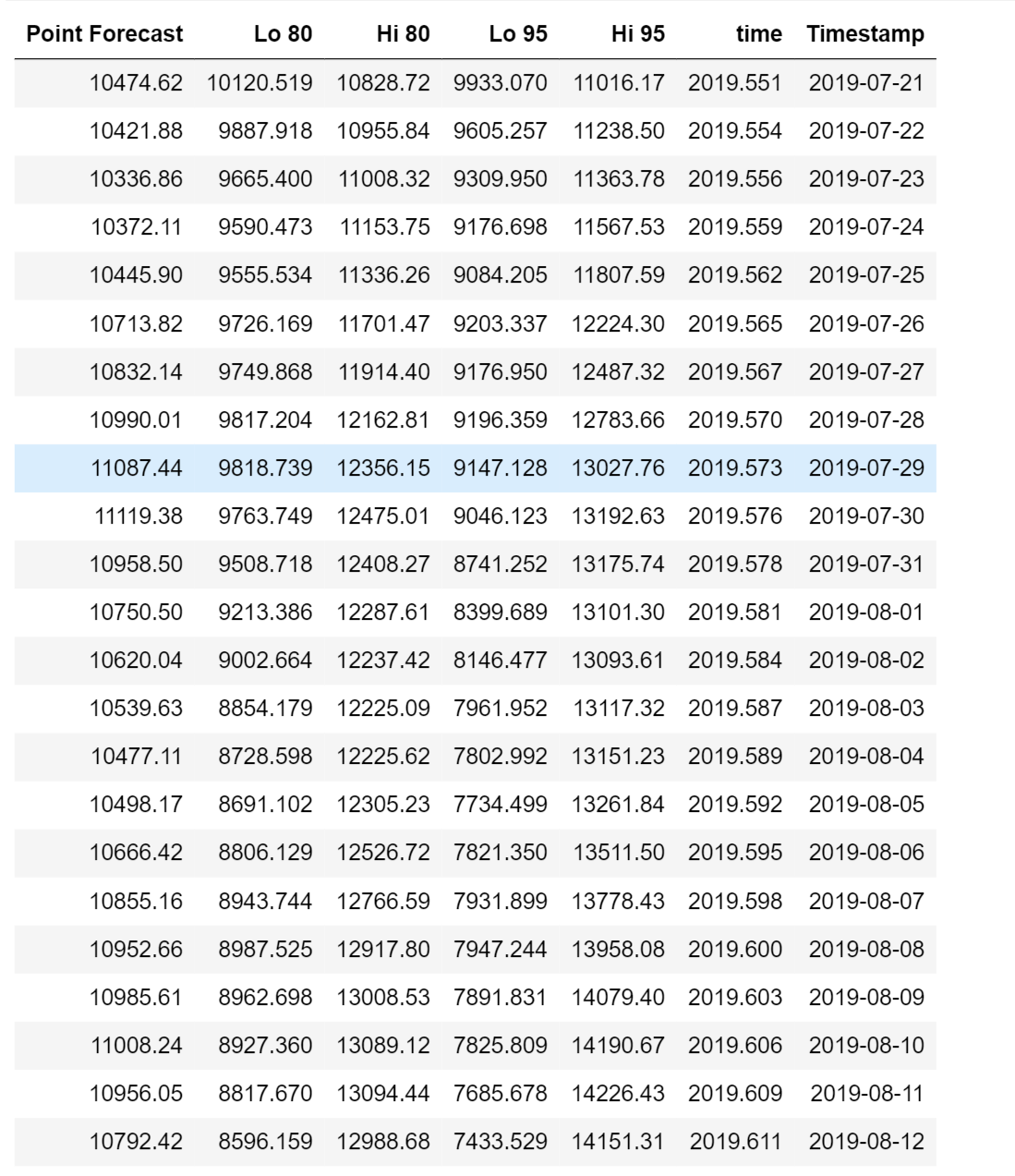
1. **ARIMA**

Following are the results from the predictions made by the ARIMA model.



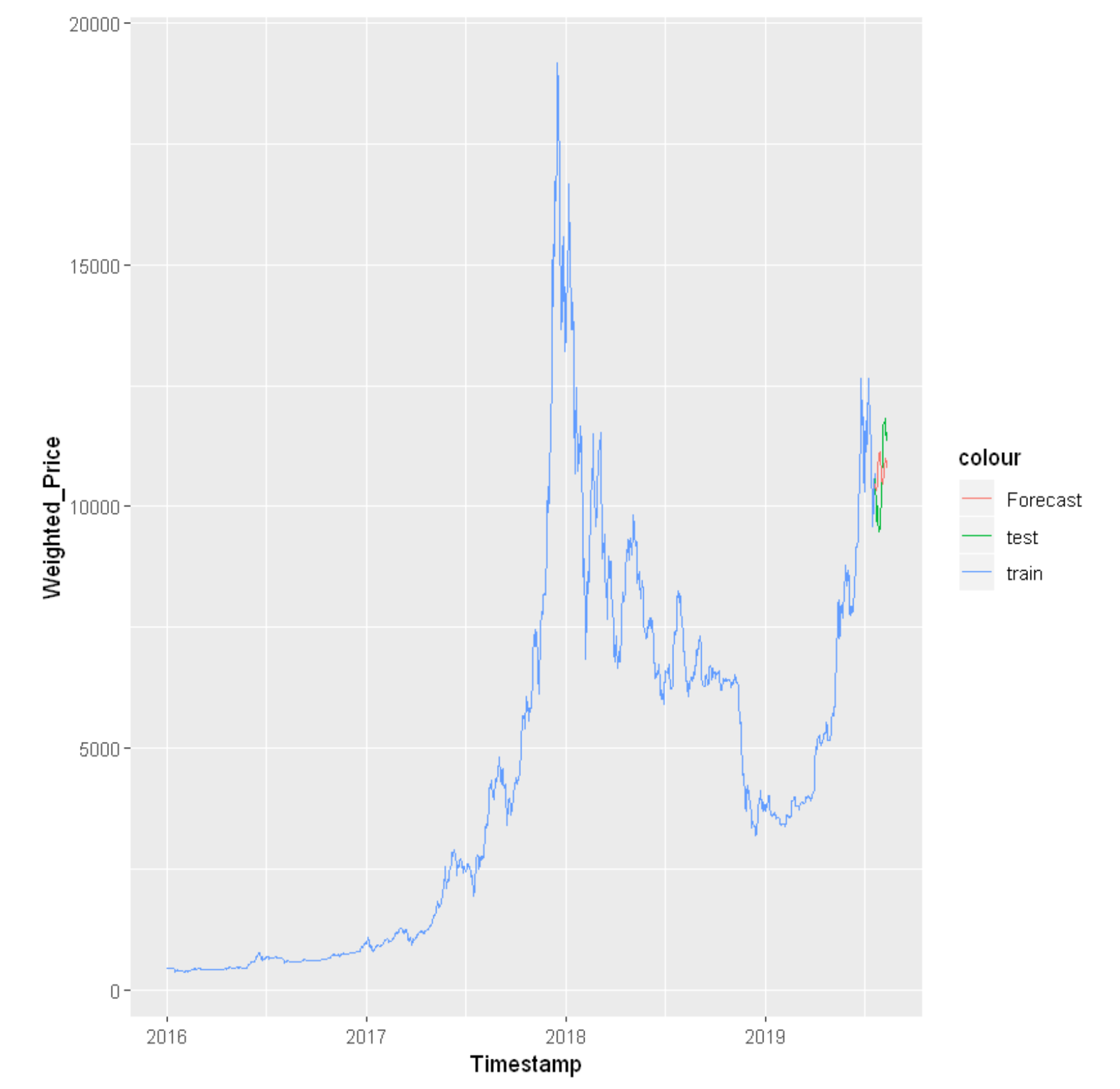
*Figure 18: Snapshot of the ARIMA forecast*

Below data fame explains the predictions made from 21-07-2019 to 12-08-2019 similar to the ones in the test data.



*Figure 19: Snapshot of the predictions*

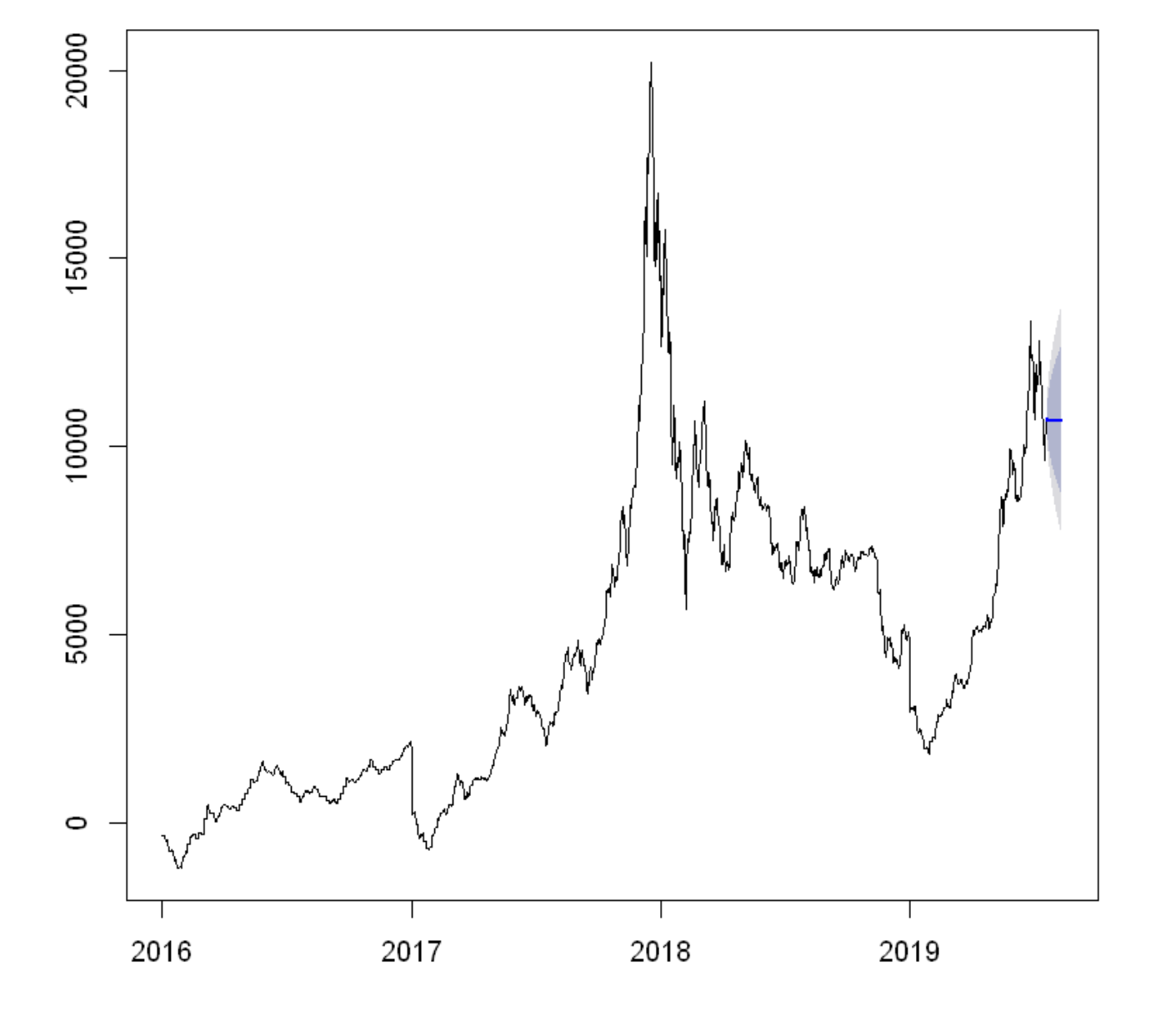
Following timeseries graph describes the train, test and forecasted bitcoin weighted prices pattern.



*Figure 20: Snapshot of the predicted and actual data*

1. **SARIMA**

Though introducing seasonality component in the ARIMA model, it gave the same predicted plot as the ARIMA plot without seasonality.



*Figure 21: Snapshot of the SARIMA forecast*

Discussion of Results

We have performed three time series models. Although auto ARIMA couldn’t give us good forecasting, grid ARIMA gave us a much better prediction which was built by manually inputting the p,d,q values with an AIC of 18287.

LSTM and GRN performed well on the available dataset when looking at the average RMSE. We could see for GRN the test RMSE got increase to 487 on the 10th iteration, which should that model is still not good to be used in a real world use. To make the result more concrete these models require further hyperparameter tuning as mentioned in the future work.

## Future Work

From the time-series algorithms perspective, we have used ARIMA and SARIMA. The analysis was focused on using the statistical methods for predicting daily weighted prices. Although the models didn’t give expected results, more complex analysis could be done using deep learning models.

From the available deep learning models, we trained the 2-layers Long Short Term Memory Neural Network as well as Gated Recurrent Unit Neural Network. The performance of the models is quite good on average, however, a hyperparameter tuning can be done further to minimize the RMSE. We can also train the model with more layers of LSTM and GRU and check the performance of the model.

# References

Below are the references that have been used for the successful implementation of the project -

1. <https://www.kaggle.com/mczielinski/bitcoin-historical-data/data>
2. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
3. <https://arxiv.org/pdf/1412.3555v1.pdf>
4. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
5. <https://medium.com/@randerson112358/build-a-bitcoin-price-prediction-program-using-machine-learning-and-python-89f3dc6cb3b1>
6. <https://medium.com/activewizards-machine-learning-company/bitcoin-price-forecasting-with-deep-learning-algorithms-eb578a2387a3>
7. <https://arxiv.org/pdf/1909.04293.pdf>
8. <https://stackoverflow.com/questions/41206181/ggplot-multiple-years-on-same-plot-by-month>
9. <https://otexts.com/fpp2/ts-objects.html>
10. <https://robjhyndman.com/hyndsight/dailydata/>