Bitcoin Stock Prediction Using SARIMA

Harnoor Singh Oberai
QMUL-190753898
Supervisor - Dr Luk Arnaut
School of Electronic Engineering and Computer Science
London, United Kingdom
ec19572@qmul.ac.uk

***Abstract*—Bitcoin establishes itself as the most popular cryptocurrency in the market, where investors leverage the currency to make a profit. Its' volatile nature has made it challenging to predict its return. Given Bitcoin's historical data, I aim to forecast future Bitcoin stock price to yield profitable trades.**

The paper scientifically proves the historical data to contain trend and seasonality. It also shows that the data is not stationary. In light of these constraints, I work on towards two contributions: first, the historical data is cleaned and preprocessed to analyze its' content; and second, I present simple, upscaled and real-time model (SARIMA) that achieve a high return on average Bitcoin.

***Index Terms*—Seasonal Autoregressive Integrated Moving Average, Trend, Bitcoin, Blockchain**

I. Introduction

A. Background

Bitcoin was created in 2009 by an unknown persons or person using the pseudo name of Satoshi Nakamoto. It relies heavily on peer-to-peer networking and cryptography to maintain its integrity (Nakamoto, 2019). It is a cryptocurrency where the type of money is completely virtual. A digital currency which is an online version of cash (Frankenfield, 2020). Bitcoin works without any financial body (bank) associated with it. There is no middleman involved in any transaction. All bitcoin transactions are recorded in a distributed ledger called a blockchain. A person can buy a product or a service by transferring bitcoins from one entity to another. The price of bitcoin skyrocketed in 2017 into thousands of dollars.

Blockchain is a set of immutable records of data recorded at a particular timestamp interval. The data is managed by a group of computers. No single entity has access to all the computers. Every block of the data is secured using cryptographic principles called a chain.

Blockchain can disrupt every industry and business model in the world. There is no central authority required. This is known as the democratized system. All the information is shared and recorded in an immutable ledger (Antonopoulos, 2015). The information in the block is open for anyone and everyone to see, making it transparent. So, everyone is accountable for their action.

B. Objective

The first objective of this paper is to perform data analysis on bitcoin historical data. It aims to clean, transform modelling data to discover useful information to business impacting decision. The paper aims to look back via the OHLC chart and model it to see how the currency has changed during the following years.

The cleaned data will be transformed and will become stationary. It will be confirmed and tested using Augment Dicky Fuller test.

The second objective of the paper is to perform SARIMA on the cleaned data set and report predicted future prices. SARIMA will take account of seasonality factor in the time series model.

II. Literature Review

A. Related Work

Investors have tried building an algorithm to find a perfect way to predict stocks with no significant success. Many factors can alter the change in stocks which are hard to capture. Factors like an epidemic, demonetization of Rupee(India), COVID-19 may directly affect the exchange rate. Political events can also have a significant influence on stock prices (Kuo et al., 2001).

Techniques like genetic algorithm and artificial neural network (ANN) have been applied in forecasting. The ANN had a limitation in learning patterns as recorded by Kim and Hann, (2000). The stock prices are equipped with multiple dimensions having enormous noise. The quantity of stock data will eventually conflict with learning patterns.

Social media platform other networking has also been used to predict Bitcoin Prices (Matta et al., 2015). The study by (Georgoula et al., 2015) investigated the frequency of Wikipedia views by sentiment analysis via Support Vector Machine. (Greaves and Au, 2015) used Blockchain to forecast the price of Bitcoin using artificial neural networks (ANN) and support vector machines (SVM). The accuracy of 55% was achieved with regular ANN. Thus, the study concluded forecasting Bitcoin with the only Blockchain as a parameter was not a success.

B. Bitcoin

Bitcoin is a computer code that is stored in a digital wallet. The wallet can be an app on a smartphone or a computer. Exchange of bitcoin occurs from one wallet to another digital wallet. Blockchain records every single transaction. The history of bitcoin is available on public lists, ensuring security.

No one can make a copy or spend a bitcoin they don't own. There are two main ways a person can buy bitcoin:

- Buy Bitcoin Online: You need a bitcoin wallet to earn or buy bitcoins. A bitcoin wallet is a software to send, receive or stock cryptocurrency in the network securely. Bitcoin wallets are downloaded and installed in mobile, desktop, web, or hardware. The user is expected to set up an account on the cryptocurrency exchange. The wallet provider approves the cryptocurrency exchange. The exchange is a market place where bitcoin trading occurs. The sellers are connected to the buyer where each transaction is displayed publicly in exchange for an authorized amount of money or digital currency. There are many different wallets and exchanges available. Some of the popular ones are: BitFinex, Bitstamp, Coinbase, Coinmama, etc
- Bitcoin Mining: In a centralized system, the government decides when to print and distribute money. This is called paper currency. Bitcoin doesn't have a central government. People solve a mathematical problem using software, GPU, and processor and are rewarded with a certain portion or number of bitcoins. These people are called bitcoin miners. Bitcoins are incentives given to people to mine and are a smart way to issue a currency (Antonopoulos, 2015). Bitcoin miners approve transactions. Mining is an integral part of Bitcoin blockchain keeping the network solid, reliable, and protected.

C. Blockchain

Blockchain has no transaction cost included. It' has a clever way of passing information from X to Y in a secure and automated way. X will initiate a transaction by a creating block. Millions of devices verify the block distributed around the net. Once verified, a unique record with a unique history is added to the chain. It is virtually impossible to falsify a record as this would mean falsifying the whole chain in millions of instances. This model is used by bitcoin for a monetary transaction and can be deployed in multiple ways.

This is a free technology. All business model that requires a small transaction fee can be made obsolete using the bitcoin blockchain.

Ebooks can be adapted with blockchain code. A fee is taken by Amazon and credit company on every sale of an eBooks. Blockchain will encode the books circulating in the network. The consumer will transfer the money directly to the author, unlocking the book. The author received all the royalties from the book (Kozlowski, 2019). There is no third- party involved in connecting the author with a reader. The market place of Amazon becomes useless.

The music industry can become profitable again. Blockchain can cut out music distributors like Apple, Spotify, or JioSaavan. Blockchain will remove "the middlemen from the process of music sales and streaming". The music bought on streaming apps can be encoded in the blockchain, creating a cloud archive for songs purchased. The amount charged by the artist will be so small that the subscription and streaming services will render irrelevant (Madeira, 2020). The immutable ledger will find its' use in IPR and piracy. The network will create a timestamp on ledger providing proof of ownership. The information will be public and will be readily available. Thus, immune from tampering and modification.

Blockchain has achieved success due to the strong three pillars (Rosic, 2020).

- Decentralization: No single entity stores all information. Everyone in the network owns a part. Interaction with your friend can be done directly without involving a third party. The decentralized system became the building block of bitcoin. A person alone is in charge of his money. A transaction from X to Y occurs without the involvement of the bank.
- Transparency: Bitcoin blockchain uses complex cryptography technology that keeps a person's identity hidden. Only his public address is available. The transaction history will show you as "1LF1bhsFLkXzzz9vpQYEmvwT2TbyCt7NZJ sent 5 BTC" rather than "Harnoor sent 5 BTC". Transparency of such kind has never existed before.
- Immutability: No transaction can be tampered once it enters into the blockchain. Bitcoin uses a cryptographic hash function. The function converts an input string and gives an output string of fixed length. Bitcoin employs SHA-256 to achieve Immutability.

D. Time Series

A time series is a continuance flow of numerical data observed in progressive order (Kenton, 2020). Time series is a set of observations on values that variable takes at a different time interval. Time Series Analysis has found its use in many applications such as:

- Census Analysis
- Sales Forecasting
- Stock Market Analysis
- Air Passenger Traffic
- Process and Quality Control
- Inventory Studies

It can also be used to study the changes connected with the extracted data point comparing with other variables over the same period of time. Time Series Analysis will be performed on bitcoin historical data to predict its future price. The analysis will track the movements of cryptocurrency's high, low, open, and close value to determine the next value. It enables the data to be flexible enough to record any minimum and maximum time. Hence, providing information to the investors to examine or analyze the data.

Some forces are responsible for affecting the values in a time series data. These are the components of a time series which are:

1) Trend: It is a long-term movement in a time series. It reflects an underlying level that shows whether the data is increasing or decreasing. It is a smooth, long term, average tendency. The trend comprises of an upward or

downward or even both biases. We can observe trends in different sections of time in Bitcoin historical data. Further details are explained and visualized in Figure 2.

2) Seasonality: When some components tend to appear themselves again over a certain period, the time series data is said to have seasonality. There is a pattern observed for 12 months. Such variation occurs when the data is recorded hourly / daily / quarterly / monthly. Such variation is results of natural or man-made forces. Ex. Production of crops depends on climate condition, shooting up of A.C in summers, chocolates sales in Christmas. They reappear theme-selves time after time.
3) Cyclic: This oscillatory movement of data that has a period of more than a year. This is called 1 complete period cycle. Also, referred as 'Business Cycle'.
4) Random or Irregular movement: Variation that cannot be foreseen are categorized as random movement. They are not regular variation. Fluctuation are abrupt, unpredictable, uncontrollable, and are erratic. Examples include earthquakes, flood, war, or any disaster like COVID-19.

E. Autocorrelation

Autocorrelation is the measure of linear relationship between the "lagged values of a time series" (Hyndman and Athanasopoulos, 2018). Autocorrelation is derived from correlation. Correlation is the measures of linear relationship between two variables.

1) ACF plot Observation for Trend and Seasonality: When a time series have a trend, the autocorrelations for short lags tend to be large and positive because observations nearby in time are also nearby in size (Hyndman and Athanasopoulos, 2018). The trended time series in the historical data observes the ACF plot to have positive values which will gradually decrease as the lags increase.

The autocorrelation for seasonal data will have larger seasonal lags. It will be observed at multiple seasonal frequencies.

F. Stationary

Stationary means constant statistical properties. A stationary time series's properties are independent on time at which the series is observed. A time-series having trend and seasonality will influence the value of variables at different times. Hence, they are not stationary. Stationary doesn't imply that data doesn't change over time. Stationary data means the data change does not itself change over time.

Before working on the Time series, we need to check if the data is stationary. One can transform a time series to be stationary with the following techniques.

1) Differencing the data (*Transforming a Series to Stationary*, 2019): Given the series Zt, we create the new series.

$$Y(i) = Z(i) - Z(i-1)$$

The differenced data will include one limited value than the original data. We differentiate the data more than once to make it stationary.

2) Logarithm(*Transforming a Series to Stationary*, 2019) or the square root of the series can stabilize the variance. Adding a positive value to negative data can make all data positive. After, we can apply logarithm or square root to the series.

The difference of 1 has helped to generate bitcoin time series stationary at constant location and scale.

G. ARIMA

ARIMA is the one of the model used for time series forecasting. ARIMA is short for 'Auto Regressive Integrated Moving Average'. It is a class of models describing the time series based on the previous value. It uses its' lags and forecasts lagged error. This equation is used to forecast future values. All 'non-seasonal' time series exhibiting patterns can be represented with ARIMA models. The models should not have random white noise. ARMIA comprises of 3 models. Combing these models and choosing appropriate order (p,d,q), the bitcoin historical data is fitted to predict the future prices.

1) Auto regressive AR(p) : In this regression model, the response variable in the past period becomes the predictor. The errors observed are the same as the errors in a simple linear regression model. We use the terminology 'order' in the autoregressive model. The order(p) of an autoregression is the number of immediately preceding values in the series. The past values aare used to predict the value at present (Holmes et al., 2020). AR(1), AR(2), ... represents AR models with different orders.

2) Moving Average MA(q): There are similarities observed between past errors and present values. Such values need to be accounted for. Moving errors helps to account for certain unpredictable events. In the time-series model, the moving average is a past error model that is multiplied by a coefficient. A moving-average process of order q, or MA(q), is a weighted sum of the current random error plus the q most recent errors (*2.1 Moving Average Models (MA models): STAT 510*, n.d.), and can be written as

$$x_t = \mu + w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}$$

We are able to tweak our model by calculating our past estimation with the present value so our model performs better.

3) Backshift Notion (d) : The backward shift is a valuable notational tool when working with time series lags. It is a convenient way for differencing. The main objective of differencing is to convert the time series data to a stationary data. The first difference can be written as:

$$By_t = y_{t-1}\ .$$

A right parameter is needed to determine the right order of differencing. A right parameter will determine the minimum differencing needed to get a stationary series. The model will have a near-constant mean and the ACF plot reaches zero reasonably fast.

Further differencing is required if the autocorrelations are positive for 10 or more number of lags. Over-differencing occurs if the lag 1 autocorrelation itself is too negative (Zhao, 2019).

H. SARIMA

Although the ARIMA models can handle the data with a trend, it doesn't perform well with a seasonal component with time series.

ARIMA assumes the data is free from the seasonal component or has the seasonality removed. Seasonal Autoregressive Integrated Moving Average, Seasonal ARIMA or SARIMA, is an extension of ARIMA that supports the seasonal component of time-series.

In addition to p,q,d, it adds four new hyperparameters to specify seasonal component:

- P: Seasonal AR(p) order.
- D: Seasonal I(d) order.
- Q: Seasonal MV(q) order.
- m: Time steps for a single seasonal period.

III. Methodology

A. Data Set

The relevant historical data set of the 1-minute time interval is integrated into this paper and consists of a transaction from multiple exchanges from Jan 2012 to April 2020. This data set is at Kaggle (Appendix - Figure 1). Kaggle is a community of Data Scientist and machine learning practitioners that contains reliable datasets. The dataset following attributes:

1) Timestamp: Start time of time window (60s window), in Unix time.
2) Open: Open price at the start time window.
3) High: High price within the time window.
4) Low: Low price within the time window.
5) Close: Close price at the end of the time window.
6) Volume_(BTC): Amount of BTC transacted in the time window.
7) Volume_(Currency): Amount of Currency transacted in the time window.
8) Weighted_Price: Volume Weighted average price.

The data set contains 436457 transactions over the years. Before training SARIMA, the data is cleaned. The data set detected 28% records to contains NaN values, i.e., there was no information available. Since these transactions add no value to SARIMA, they are removed. Any other missing attribute from Open, High, Low, Close is replaced with preceding value in the data. Figure 6 in Appendix contains function "treat_missing(df, choice)" handles all the missing values.

After cleaning the data, the data set contains 3126480 transactions.

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00	4.39	4.39	4.39	4.39	0.455581	2.000000	4.390000
1	2011-12-31 15:50:00	4.39	4.39	4.39	4.39	48.000000	210.720000	4.390000
2	2011-12-31 16:59:00	4.50	4.57	4.50	4.57	37.862297	171.380338	4.526411
3	2011-12-31 17:00:00	4.58	4.58	4.58	4.58	9.000000	41.220000	4.580000
4	2012-01-01 04:16:00	4.58	4.58	4.58	4.58	1.502000	6.879160	4.580000

Fig. 1. Data Set

B. Bitcoin over the years

Bitcoin over the years: Figure 2 and 3 shows the high value of bitcoin over the years. Trading in bitcoin started in 2013. It started trading at $13.50 per bitcoin.

Key Takeaway:

1) Bitcoin in the first year (2013) of trading started rallying up from $100. In November end, the price went over $1075 (Ofir, 2020).
2) The graph shows a Meteoric Rise and Fall of Bitcoin. 2017-19 encountered maximum number of highs and lows in the price. Bitcoin broke at $5000 and was doubled again in November to $10000. Bitcoin almost reached $20000 in December.
3) Bitcoin rapidly started falling to $7000 by April 2018 and below $3500 by November 2018.
4) Bitcoin started recovering in 2019 and saw an increase in its price and Volume around $10000 by June (*Bitcoin price index — coindesk*, 2020).
5) The highest value ever recorded by bitcoin was $19,783.

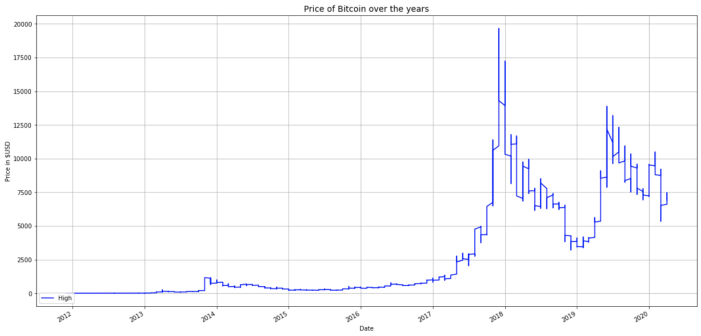


Fig. 2. BTC High Value over the years

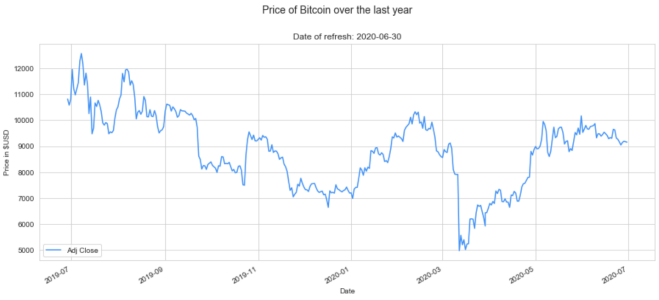


Fig. 3. BTC High Value 2019-20

C. Time Series Stats

Figure 4 shows the presence of trend, seasonality and staionarity. There are two ways to check whether BTC historical data is stationary.

1) Rolling Statistics: By plotting moving average or moving variance and see if it varies with time. Rolling statistics is a visual technique which is demonstrated in Figure 4. Figure 4 shows the Rolling Mean in red

colour and rolling standard deviation in black colour plotted against the 12-month time window. The mean and variance displayed are not constant.

2) Augment Dick Fuller Test (ADFT Test): It is a unit root test for stationarity (Stephanie, 2020). Augment Dicky Fuller test calculates ADF Statistics and the critical values of the original graph and test against the Null hypothesis. If the ADF Statistics is less than the critical value, we reject the null hypothesis. Alternatively, If the ADF Statistics is greater than the critical value, we reject the alternate hypothesis. The data should reject null hypotheses to be stationary.
   - Null Hypothesis: Time Series in Non – Stationary.
   - Alternate Hypothesis: Time Series is Stationary.

The table 1 shows the relation between Weighted Average with time . It appears to have an ADF statistics of -1.722854 and critical value (5%) of -2.891208, rejecting the alternate hypothesis. Hence, the data is not stationary. The next steps is to make our data stationary and normalized using Box-Cox Transformations.

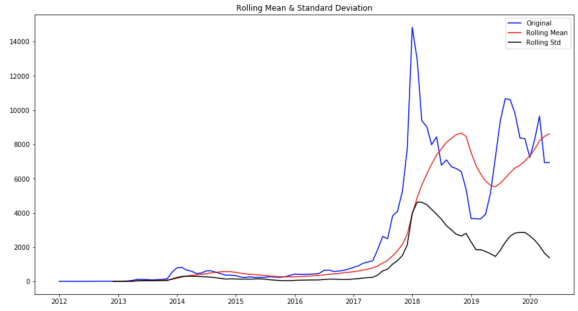


Fig. 4. Weighted Average Rolling Mean and Standard Deviation

TABLE I
AUGMENT DICK FULLER TEST

Test Statistic	-1.722854
p-value	0.419378
#Lags Used	1.000000
Number of Observations Used	99.000000
Critical Value (1%)	-3.498198
Critical Value (5%)	-2.891208
Critical Value (10%)	-2.582596

ADFT test generated Figure 5 that confirms the presence of trend, seasonality and residues of Weighted Average. Function ”seasonal_decompose” is useful in extracting trend, series and residuals from the data. Figure 5 shows some interesting facts. Bitcoin gradually increased until mid-2017. After 2017 there is a positive exponential trend observed, but by April 2018, there is a negative trend observed. Bitcoin gained positive confidence in 2019.

The residuals are also interesting, showing periods of low variability in early years and high variability from 2017 of the series.

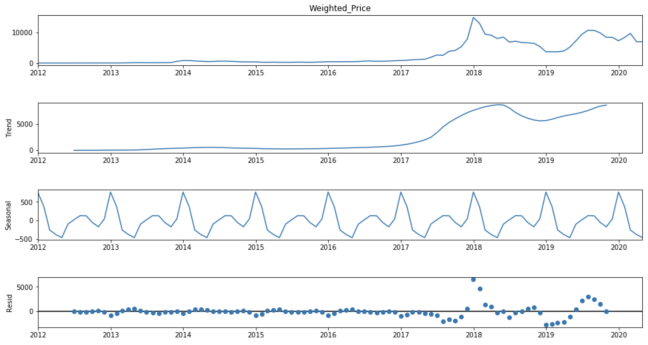


Fig. 5. Statsmodels - ADFT

IV. IMPLEMENTATION

A. Box-Cox Transformation

Statistical test and interval are based on normality that gives an accurate test. Unfortunately, the Weighted Average is not distributed normally. The data has to undergo transformation to be distributed normally. Weighted Average undergoes a box-cox transformation. The statisticians George Box and David Cox in 1964 contracted a method to identify a Lambda exponent to use to transform data into a normal distribution. The Lambda value shows the power to which all data should be raised. Box-Cox transformation is defined as:

$$T(Y) = (Y^{\lambda} - 1)/\lambda$$

Where Y is the response variable, and lambda is the transformation parameter. For lambda = 0, the natural log of the data is taken instead of using the above formula.

Following steps were taken to transform the Weighted Average Data:

1) Weighted Average underwent box transformation and was stored in new attribute “Weighted_Price_box”.
2) Each value of “Weighted_Price_box” was subtracted with last 12th value (window value=12).
3) Lastly Regular differention was performed for ”prices_box_diff” by subtracting the previous value of ”prices_box_diff”and is stored in new attribute “prices_box_diff2”.

B. Augment Dicky Fuller Test

Visually, we can see the new Weighted Average (”prices_box_diff2”) by plotting rolling moving average and standard deviation. Figure 6 shows a near constant mean and variance.

Next we perform Augmented Dicky Fuller Test on ”prices_box_diff”. The table 2 shows the relation between ”prices_box_diff” with time . The new ADF Test statistics of -4.636599 and critical value (5%) of -2.908645 is observered. The Test statistics is less than critical value, thus rejecting the null hypothesis. Hence, the data is has now become stationary.

ADFT test for box transformation is supported by Figure 7 that confirms that the data has become stationary. The data

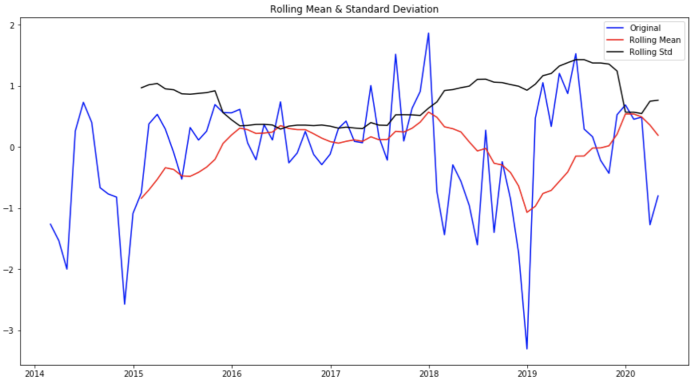


Fig. 6. Weighted Average Rolling Mean and Standard Deviation

TABLE II
AUGMENT DICK FULLER TEST AFTER BOX-COX TRANSFORMATION

Test Statistic	-4.636599
p-value	0.000111
#Lags Used	11.000000
Number of Observations Used	63.000000
Critical Value (1%)	-3.538695
Critical Value (5%)	-2.908645
Critical Value (10%)	-2.591897

has preprocessd by removing staionarity and is ready to be fitted with SARIMA.

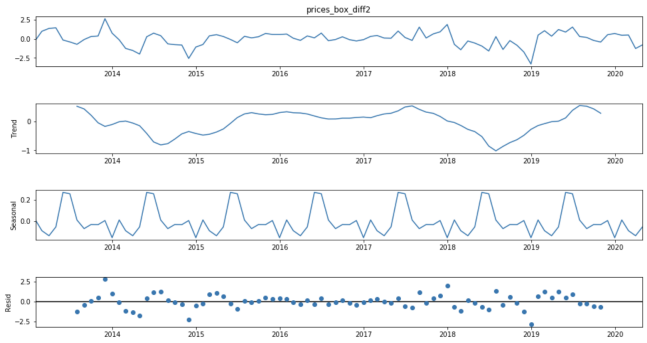


Fig. 7. Statsmodels - ADFT(Box-Cox)

C. Auto-correlation (ACF plots)

Figure 8 shows auto-correlation (ACF plots) for Weighted Average with itself. The observation for the lag of 48 periods is described in the graph. The plot shows the lag along the x-axis and the correlation on the y-axis. The historical data observers a high auto-correlation as there is slow decay for the upcoming lags. It shows that our data contains trend.

Figure 4 and 7 provide strong signal of non stationarity, but the ACF plot in Figure 8 helps us ascertain this implication. Streams like those seen in this observation is a sign of seasonality and noise in the dataset.

ACF plot for prices_box_diff2" is represented in Figure 9. The new ACF plot for the data is dropping to zero relatively quickly, showing stationarity. The ACF plot does not expect values to be above the significance range for upcoming lags.

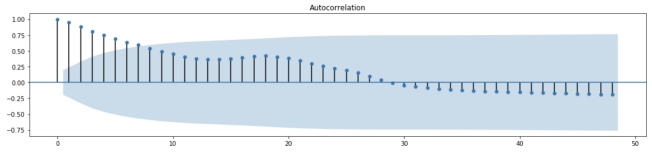


Fig. 8. ACF plots without Box-Cox Transformation)

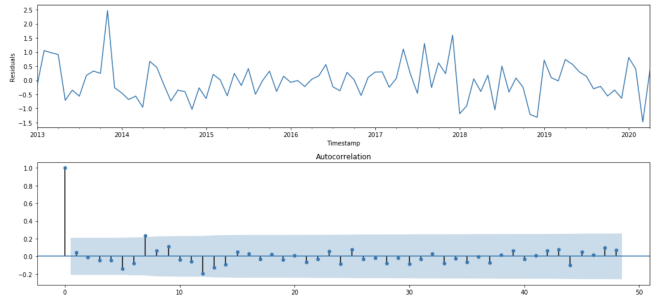


Fig. 9. ACF plots with Box-Cox Transformation

D. SARIMA

The SARIMA model eliminates non-stationarity of the series by using seasonal differencing of relevant order. Fitting a SARIMA model to data involves the following four-step iterative cycles:

1) Identify the SARIMA parameters (p, d, q) (P, D, Q).
2) Estimate unknown parameters environment if any.
3) Perform goodness-of-fit tests on the estimated residuals.
4) Forecast future outcomes based on the known data.

The parameters of the SARIMA model must be estimated using methods that generate estimates relevant for the data. When applied in optimization problems, the spread of the scenario tree is of great importance, implying that the variance of the SARIMA model is essential. The ACF, PACF and variance of the SARIMA model also support in estimating the parameters. A total of 81 different combination of 6 parameter are used to calculate AIC(Akaike's 'An Information Criterion). Table 3 shows the top 5 parameters with the lowest AIC.

The parameters used is SARIMAX(1, 1, 0)x(0, 1, [1], 12) having lowest AIC of 175.9571.

E. Libraries Used

The primary language to conduct the analysis and build model is python. Python encompasses a vast ecosystem containing powerful libraries and packages. It is a high-level, interpreted language following the principles of object-oriented for clean and logical code (*Python Features*, 2020). Following are the libraries used to build the SARIMA model:

1) Numpy: It is a necessary package for scientific computing with Python (*NumPy Documentation*, 2020). Some features of the packages are:

TABLE III
SARIMA PARAMETERS (P, D, Q)(PS, D, QS)

p	d	q	Ps	D	Qs	AIC
1	1	0	0	1	1	175.9571
1	1	0	1	1	1	176.3145
1	1	0	2	1	1	176.6804
0	1	1	0	1	1	177.0991
0	1	1	1	1	1	177.1094
0	1	1	2	1	1	177.1645
2	1	0	0	1	1	177.843
1	1	1	0	1	1	177.8573
0	1	2	0	1	1	177.87
2	1	0	1	1	1	178.2539

- Powerful n-dimensional arrays.
- Numerical computing tools.
- Open-source
- Interoperable

2) Pandas: An open-source, high-performance library with built-in data structure and data analysis tool. Helpful in interpreting time series model and historical data of bitcoin (*Pandas Doc*, 2020).
3) Matplotlib: Matplotlib is an interactive environment that helps in visualizing plots, bar charts, scatter plots, OHLC charts.
4) Seaborn: Data visualization library built on top of matplotlib. Helpful in plotting and interpretation charts
5) Scikit-learn: Machine -Learning library supporting both; supervised and unsupervised learning. Provides support for:
   - Data pre-processing.
   - Model selection.
   - Model fitting.
   - Evaluation.

The code is in with Project Jupyter. Project Jupyter is an open source software offering services for interactive computing. Project Jupyter's product Jupyter Notebook supports python3 environment. The statistical model is built in Jupyter Notebook.

V. RESULTS

Figure 10 shows a summary of the best SARIMA model. The max AIC value of 300 is reduced to 175.917. The figure also shows the coefficient table where the coefficient of the individual contributor is represented by 'coef'. The p-value of the AR1 is highly significant ($p \ll 0.05$).

Figure 11 shows the Weighted_price_box plotted against predicted price using SARIMA model. The model gets trained up until previous value to make prediction. To validate prediction, the model will need the Out-of-Time cross validation.

The data set is split into training, and test data where the training data consist of weighted_values from 2011-12-31 until 2019-01-31 and the test data contains weighted_values from 2019-01-31. The data will not be shuffled. The order of sequence should be intact in time series SARIMA. The model forecast into the future by taking the last 15 steps

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                Weighted_Price_box   No. Observations:                  101
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood                -84.979
Date:                              Sun, 09 Aug 2020   AIC                            175.957
Time:                                      18:43:39   BIC                            183.389
Sample:                                  12-31-2011   HQIC                           178.951
                                       - 04-30-2020
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3823      0.092      4.178      0.000       0.203       0.562
ma.S.L12      -0.9955      8.047     -0.124      0.902     -16.768      14.777
sigma2         0.3034      2.425      0.125      0.900      -4.450       5.057
===================================================================================
Ljung-Box (Q):                       22.93   Jarque-Bera (JB):                 4.88
Prob(Q):                              0.99   Prob(JB):                         0.09
Heteroskedasticity (H):               1.51   Skew:                             0.33
Prob(H) (two-sided):                  0.27   Kurtosis:                         3.95
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Fig. 10. SARIMA - Results

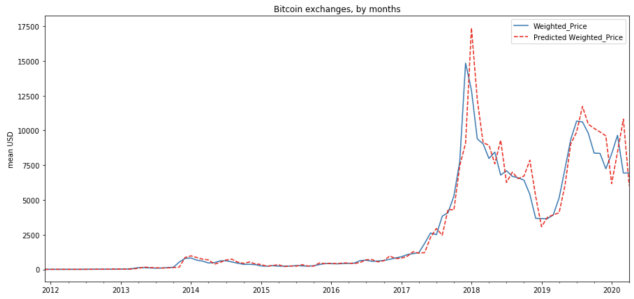


Fig. 11. SARIMA- Model

back and forecasting the next 15 values. Figure 12 is a plot against training, test and actual data. From the chart, the SARIMA model gives the directionally correct forecast. The observed data lie within 95% confidence band where 2019-20 forecasted values are consistently below. Forecast value in 2020 has observed the least error. This paper uses Mean

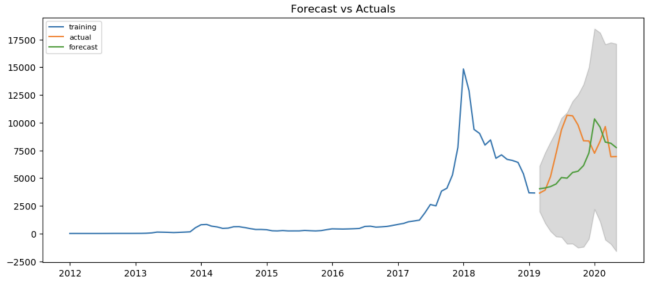


Fig. 12. SARIMA - Forecast VS Actual

absolute percentage error (MAPE) as an accuracy metrics for the time series. The mean absolute percentage error (MAPE) is the mean or average of the absolute percentage errors of forecasts. Error is defined as actual or observed value minus the forecasted value (Swamidass, 2006). These metrics compute the error in terms of percentage. Thus we can judge how good is the forecast of historical data. MAPE is given by:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

SARIMA model observers around 26.8% MAPE which imply the model is accurate with 73.2% in predicting the next 15 observations.

VI. Conclusion

In conclusion, the SARIMA model, was able to forecast Weighted Price with accuracy around 31% to 73%. Train: Test ratio of 92:9 and 93:8 had the lowest accuracy(30%). The accuracy heavily depended on the ratio of training to validation set. The performance of this prediction indicates that SARIMA model is useful for calculating next month change in price.

The model performed good predicting immediate next 5 month values with an accuracy of (76%).

VII. Future work

MLOps of continuous delivery and automation will be implemented so the model automatically feeds in the new data, updates the model parameters and predicts future price again. Further studies can be carried out in future. This study focuses on the Weighted Price on the SARIMA model. Prediction are based on monthly prices of Bitcoin. In real-life, the monthly prediction might not be enough for investors. Prediction prices by day will be more beneficial and be helpful for short-term and long term investment. Monthly prediction can accommodate long-term investment. Support Vector Machine models can enrich the analyses of the best predictive method for bitcoin on the top of SARIMA.

Acknowledgement

I wish to express my deepest gratitude to my supervisor, Dr Luk Arnaut, for his guidance and dedicated support. Dr Luk Arnaut is a substance of genius: he encouraged me to be professional and giving ample space for trial and error. Without his feedback and persistent help, the goal of the project would not have been realized.

References

2.1 Moving Average Models (MA models): STAT 510 (n.d.).
URL: *https://online.stat.psu.edu/stat510/lesson/2/2.1*

Antonopoulos, A. M. (2015), *Mastering Bitcoin*, 2 edn, O'Reilly Media, Incorporated.

Bitcoin price index — coindesk (2020).
URL: *https://www.coindesk.com/price/bitcoin*

Box, G. E. P. and Cox, D. R. (1964), 'An analysis of transformations', *Journal of the Royal Statistical Society: Series B (Methodological)* **26**(2), 211–243.

Frankenfield, J. (2020), 'Digital currency'.
URL: *https://www.investopedia.com/terms/d/digital-currency.asp*

Georgoula, I., Pournarakis, D., Bilanakos, C., Sotiropoulos, D. N. and Giaglis, G. M. (2015), 'Using time-series and sentiment analysis to detect the determinants of bitcoin prices', *SSRN Electronic Journal* .

Greaves, A. and Au, B. (2015), 'Using the bitcoin transaction graph to predict the price of bitcoin', *No Data* .

Holmes, E. E., Scheuerell, M. D. and Ward, E. J. (2020), 'Applied time series analysis for fisheries and environmental sciences'.
URL: *https://nwfsc-timeseries.github.io/atsa-labs/sec-tslab-autoregressive-ar-models.html*

Hyndman, R. J. and Athanasopoulos, G. (2018), *Forecasting*, 2 edn.

Kenton, W. (2020), 'Understanding time series'.
URL: *https://www.investopedia.com/terms/t/timeseries.asp*

Kim, K.-J. and Han, I. (2000), 'Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index', *Expert Systems with Applications* **19**(2), 125–132.

Kozlowski, M. (2019), 'Blockchain technology may lead to true ebook ownership'.
URL: *https://goodereader.com/blog/e-book-news/blockchain-technology-may-lead-to-true-ebook-ownership*

Kuo, R., Chen, C. and Hwang, Y. (2001), 'An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network', *Fuzzy Sets and Systems* **118**(1), 21.

Madeira, A. (2020), 'Blockchain to disrupt music industry and make it change tune'.
URL: *https://cointelegraph.com/news/blockchain-to-disrupt-music-industry-and-make-it-change-tune*

Matta, M., Lunesu, I. and Marchesi, M. (2015), 'The predictor impact of web search media on bitcoin trading volumes', *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management* .

Nakamoto, S. (2019), Bitcoin: A peer-to-peer electronic cash system, Technical report, Manubot.

NumPy Documentation (2020).
URL: *https://numpy.org/*

Ofir, B. (2020), 'Bitcoin historical price'.
URL: *https://99bitcoins.com/bitcoin/historical-price/*

Pandas Doc (2020).
URL: *https://pandas.pydata.org/docs/*

Python Features (2020).
URL: *https://www.geeksforgeeks.org/python-features*

Rosic, A. (2020), 'What is blockchain technology? a step-by-step guide for beginners'.
URL: *https://blockgeeks.com/guides/what-is-blockchain-technology/*

Stephanie (2020), 'Adf - augmented dickey fuller test'.
URL: *https://www.statisticshowto.com/adf-augmented-dickey-fuller-test*

Swamidass, P. M. (2006), *Encyclopedia of production and manufacturing management*.
URL: *https://link.springer.com/referenceworkentry/10.1007/1-4020-0612-8*$_5$80

Transforming a Series to Stationary (2019).
URL: *https://financetrain.com/transforming-a-series-to-*

stationary/

Zhao, M. (2019), 'Arima models — business analytics 1.0 documentation'.
URL: *https://ming-zhao.github.io/Business-Analytics/html/docs/time*$_s eries/arima.html$

APPENDIX

The code has been uploaded to Git where you can access the jupyter notebook at:
https://github.com/HarnoorOberai/BitcoinStockPrediction

Converting CSV to pandas.DataFrame

In [3]:
```
#define a conversion function for the native timestamps in the csv file
def dateparse (time_in_secs):
    return datetime.datetime.fromtimestamp(float(time_in_secs))
```

The historical data is huge. I saved the data in my google drive. The data is accesible via 'https://drive.google.com/file/d/1aPutuBRJVkx3ufJcMm60mQBM7Zhf2Ffi/view?usp=sharing' ¶

Download the data file and update the URL variable

In [4]:
```
url = 'bitstampUSD_1-min_data_2012-01-01_to_2020-04-22.csv'
```

In [5]:
```
bitcoin_EDA = pd.read_csv(filepath_or_buffer = url,parse_dates=[0], date_parser=dateparse)
bitcoin_EDA.head()
```

Out[5]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
1	2011-12-31 07:53:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2011-12-31 07:54:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2011-12-31 07:55:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2011-12-31 07:56:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig. 1. Setting Up The DATA

In [6]:
```
bitcoin_EDA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4363457 entries, 0 to 4363456
Data columns (total 8 columns):
 #   Column             Dtype
---  ------             -----
 0   Timestamp          datetime64[ns]
 1   Open               float64
 2   High               float64
 3   Low                float64
 4   Close              float64
 5   Volume_(BTC)       float64
 6   Volume_(Currency)  float64
 7   Weighted_Price     float64
dtypes: datetime64[ns](1), float64(7)
memory usage: 266.3 MB
```

In [7]:
```
bitcoin_EDA.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Open	3126480.0	3674.655957	3935.577977	3.8	410.000000	1175.140000	6931.175000	1.966576e+04
High	3126480.0	3677.365916	3939.076640	3.8	410.240000	1175.770000	6935.780000	1.966600e+04
Low	3126480.0	3671.730040	3931.712757	1.5	409.830000	1174.825000	6926.790000	1.964996e+04
Close	3126480.0	3674.595305	3935.490392	1.5	410.000000	1175.140000	6931.225000	1.966575e+04
Volume_(BTC)	3126480.0	9.855040	32.292724	0.0	0.398812	1.990000	7.639098	5.853852e+03
Volume_(Currency)	3126480.0	28844.590321	101027.675797	0.0	350.375910	2620.490826	17600.566162	7.569437e+06
Weighted_Price	3126480.0	3674.570455	3935.457615	3.8	409.999788	1175.199592	6931.179505	1.966330e+04

Fig. 2. Attributes Information

No of instances = 4363457 ¶

```
In [8]: bitcoin_EDA.shape[0]
```

```
Out[8]: 4363457
```

Fig. 3. Initial Number of instances

Checking Null values

```
In [9]: def missing_values(df):
            total = df.isnull().sum().sort_values(ascending=False)
            percent = df.isnull().sum().sort_values(ascending=False) / df.shape[0] *100
            missing_values = pd.concat([total,percent],axis=1,keys = ['Total', 'Percentage'])
            return missing_values
```

```
In [10]: missing_values(bitcoin_EDA)
```

Out[10]:

	Total	Percentage
Weighted_Price	1236977	28.348555
Volume_(Currency)	1236977	28.348555
Volume_(BTC)	1236977	28.348555
Close	1236977	28.348555
Low	1236977	28.348555
High	1236977	28.348555
Open	1236977	28.348555
Timestamp	0	0.000000

Fig. 4. Missing Value Function

Dropping rows that have no information about OHLC. We know only timestamp for that instance.

```
In [11]: toDropRows = bitcoin_EDA[(bitcoin_EDA['Open'].isnull()) &
                                  (bitcoin_EDA['High'].isnull() &
                                   bitcoin_EDA['Low'].isnull()) &
                                  (bitcoin_EDA['Close'].isnull())]
         bitcoin_EDA = bitcoin_EDA.drop(toDropRows.index)
         bitcoin_EDA = bitcoin_EDA.reset_index()
         bitcoin_EDA = bitcoin_EDA.drop(['index'], axis=1)
         bitcoin_EDA.head()
```

Out[11]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00	4.39	4.39	4.39	4.39	0.455581	2.000000	4.390000
1	2011-12-31 15:50:00	4.39	4.39	4.39	4.39	48.000000	210.720000	4.390000
2	2011-12-31 16:59:00	4.50	4.57	4.50	4.57	37.862297	171.380338	4.526411
3	2011-12-31 17:00:00	4.58	4.58	4.58	4.58	9.000000	41.220000	4.580000
4	2012-01-01 04:16:00	4.58	4.58	4.58	4.58	1.502000	6.879160	4.580000

Fig. 5. Dropping Rows with no info about OHLC values

To cleanse future data

In [12]:
```
def treat_missing(df, choice):
    if(choice==1):
        df = df.dropna()
    elif(choice==2):
        df = df.fillna(df.mean())
    elif(choice==3):
        df = df.fillna(df.mode())
    elif(choice==4):
        df = df.fillna(df.median())
    elif(choice==5):
        df = df.fillna(method='ffill')
    elif(choice==6):
        df = df.fillna(method='bfill')
    else:
        df = df.fillna(0)
    return df
```

Replacing OHLC value with precedding values

In [13]:
```
bitcoin_EDA['Volume_(BTC)'] =treat_missing(bitcoin_EDA['Volume_(BTC)'], 7)
bitcoin_EDA['Volume_(Currency)'] =treat_missing(bitcoin_EDA['Volume_(Currency)'], 7)
bitcoin_EDA['Weighted_Price'] =treat_missing(bitcoin_EDA['Weighted_Price'], 7)
bitcoin_EDA['Open'] =treat_missing(bitcoin_EDA['Open'], 5)
bitcoin_EDA['High'] =treat_missing(bitcoin_EDA['High'], 5)
bitcoin_EDA['Low'] =treat_missing(bitcoin_EDA['Low'], 5)
bitcoin_EDA['Close'] =treat_missing(bitcoin_EDA['Close'], 5)
```

Fig. 6. Replacing OHLC with preceding value

After cleansing data

In [14]:
```
missing_values(bitcoin_EDA)
```

Out[14]:

	Total	Percentage
Weighted_Price	0	0.0
Volume_(Currency)	0	0.0
Volume_(BTC)	0	0.0
Close	0	0.0
Low	0	0.0
High	0	0.0
Open	0	0.0
Timestamp	0	0.0

Fig. 7. Cleansed Data to contain no NULL values

Number of instances in clean data = 3126480

In [15]:
```
bitcoin_EDA.shape[0]
```

Out[15]:
```
3126480
```

Fig. 8. Clean Data Instances

Open

In [16]:
```
# Line Plot usage
bitcoin_EDA.plot(kind='line', x = 'Timestamp', y = 'Open' ,color='g',legend = True, label='Open', figsize=(20, 10),grid
plt.legend(loc='lower left')
plt.title('Price of Bitcoin over the last year',fontsize=14, y=1)
plt.suptitle('Opening Price')
plt.ylabel('Price in $USD')
plt.xlabel('Date')
plt.show()
```

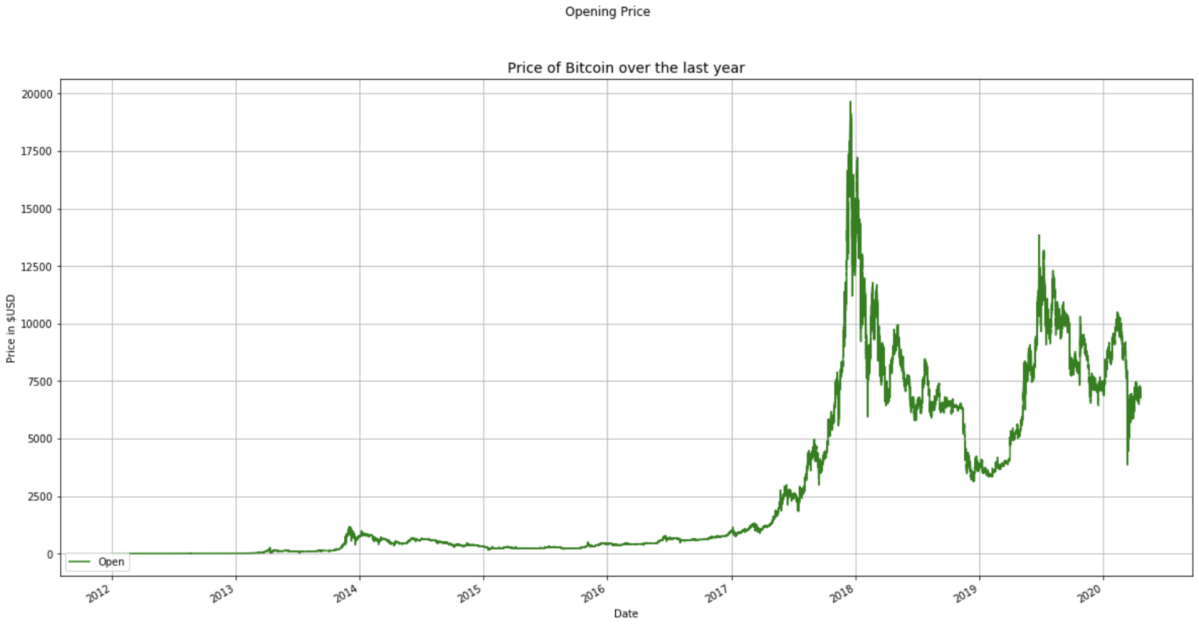


Fig. 9. Bitcoin - Open Graph

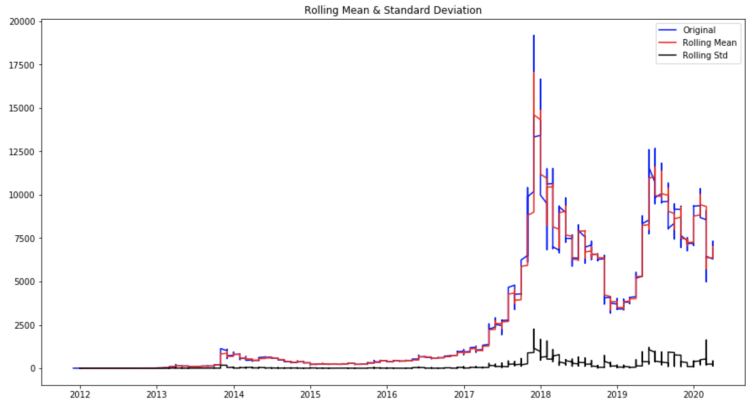
In [21]:
```
def test_stationarity(timeseries):
  # Determing rolling statistics

    rolmean = timeseries.rolling(12).mean()
    rolstd = timeseries.rolling(12).std()

    #Plot rolling statistics:
    plt.figure(figsize=(15,8))
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)
    #Perform Dickey-Fuller test:
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)
    if dftest[0] < dftest[4]["5%"]:
        print ("Reject Ho - Time Series is Stationary")
    else:
        print ("Failed to Reject Ho - Time Series is Non-Stationary")
```

Fig. 10. Test Stationarity Function

```
In [27]: test_stationarity(time_series)
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.640764
p-value                          0.461862
#Lags Used                      29.000000
Number of Observations Used    3003.000000
Critical Value (1%)              -3.432529
Critical Value (5%)              -2.862503
Critical Value (10%)             -2.567283
dtype: float64
Failed to Reject Ho - Time Series is Non-Stationary
```

Fig. 11. Initial Stationary test - Rejected Alternate Hypothesis

High price by DAY

```
In [28]: # Line Plot usage
bitcoin_EDA_High_Low.plot(kind='line', x = 'Date', y = 'High' ,color='b',legend = True, label='High', figsize=(20, 10),
plt.legend(loc='lower left')
plt.title('Price of Bitcoin over the years',fontsize=14, y=1)
plt.suptitle('High Price')
plt.ylabel('Price in $USD')
plt.xlabel('Date')
plt.show()
```

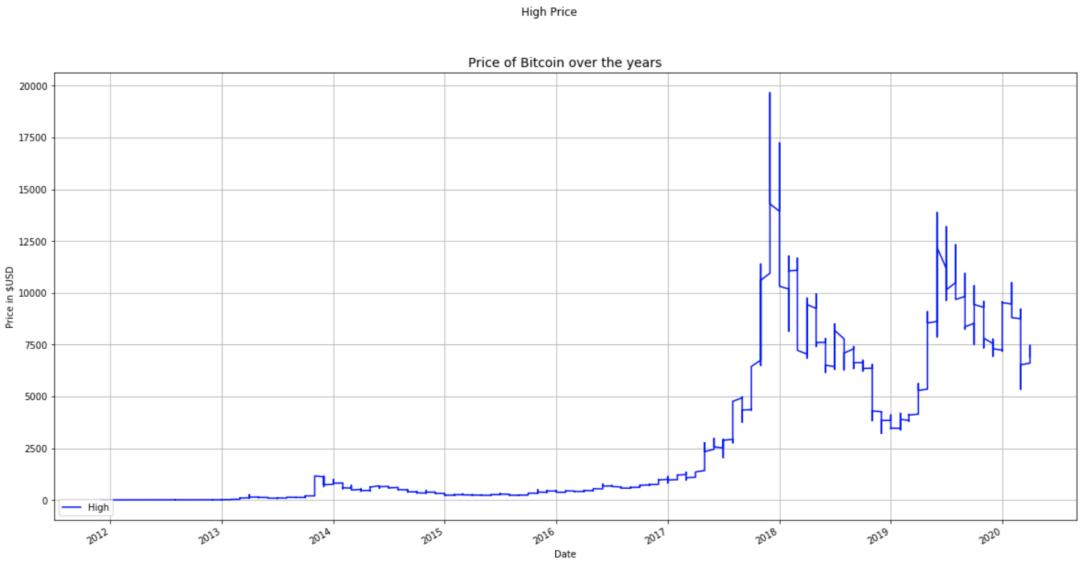


Fig. 12. High Price By Day

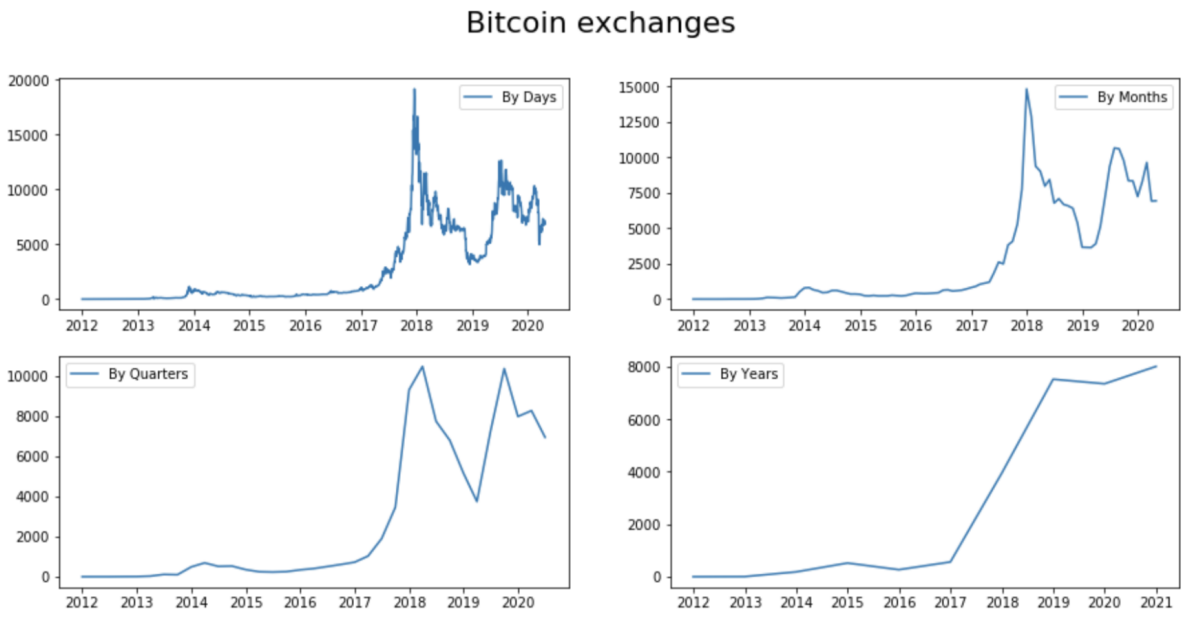


Fig. 13. Bitcoin Exchanges

Number of parameters = 81

In [41]:
```
Qs = range(0, 3)
qs = range(0, 3)
Ps = range(0, 3)
ps = range(0, 3)
D= range(1)
d= range(1)
parameters = product(ps, qs, Ps, Qs, d, D)
parameters_list = list(parameters)
len(parameters_list)
# type(ps)


# (1, 0, 0, 2, 1, 1)  175.917083
```

Out[41]: 81

Fig. 14. Number of Parameters = 81

SARIMA model

In [42]:
```
Qs = range(0, 3)
qs = range(0, 3)
Ps = range(0, 3)
ps = range(0, 3)
D= range(1,3)
d= range(1,3)
parameters = product(ps, qs, Ps, Qs, d, D)
parameters_list = list(parameters)
len(parameters_list)

# Model Selection
results = []
best_aic = float("inf")
warnings.filterwarnings('ignore')
for param in parameters_list:
    try:
        model=sm.tsa.statespace.SARIMAX(df_month.Weighted_Price_box, order=(param[0], param[4], param[1]),
                                        seasonal_order=(param[2], param[5], param[3], 12)).fit(disp=-1)
    except ValueError:
        print('wrong parameters:', param)
        continue
    aic = model.aic
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param, model.aic])
```

Fig. 15. SARIMA Model Building

In [43]:
```
result_table = pd.DataFrame(results)
result_table.columns = ['parameters', 'aic']
print(result_table.sort_values(by = 'aic', ascending=True).head())
print(best_model.summary())
```

```
             parameters         aic
116  (1, 0, 0, 2, 1, 1)  175.917083
112  (1, 0, 0, 1, 1, 1)  175.957124
124  (1, 0, 1, 1, 1, 1)  176.314466
44   (0, 1, 0, 2, 1, 1)  176.593096
136  (1, 0, 2, 1, 1, 1)  176.680382
                                      SARIMAX Results
==========================================================================================================
Dep. Variable:                      Weighted_Price_box   No. Observations:                  101
Model:             SARIMAX(1, 1, 0)x(0, 1, [1, 2], 12)   Log Likelihood                -83.959
Date:                                 Tue, 01 Sep 2020   AIC                            175.917
Time:                                         13:31:03   BIC                            185.826
Sample:                                     12-31-2011   HQIC                           179.909
                                          - 04-30-2020
Covariance Type:                                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3508      0.097      3.621      0.000       0.161       0.541
ma.S.L12      -1.0485      0.168     -6.249      0.000      -1.377      -0.720
ma.S.L24       0.2137      0.128      1.672      0.095      -0.037       0.464
sigma2         0.3236      0.056      5.811      0.000       0.214       0.433
===================================================================================
Ljung-Box (Q):                       19.98   Jarque-Bera (JB):                 4.46
Prob(Q):                              1.00   Prob(JB):                         0.11
Heteroskedasticity (H):               1.69   Skew:                             0.40
Prob(H) (two-sided):                  0.16   Kurtosis:                         3.76
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Fig. 16. Best Model Summary

In [47]:
```
# Inverse Box-Cox Transformation Function
def invboxcox(y,lmbda):
   if lmbda == 0:
      return(np.exp(y))
   else:
      return(np.exp(np.log(lmbda*y+1)/lmbda))
```

Fig. 17. Inverse Box Cox Function

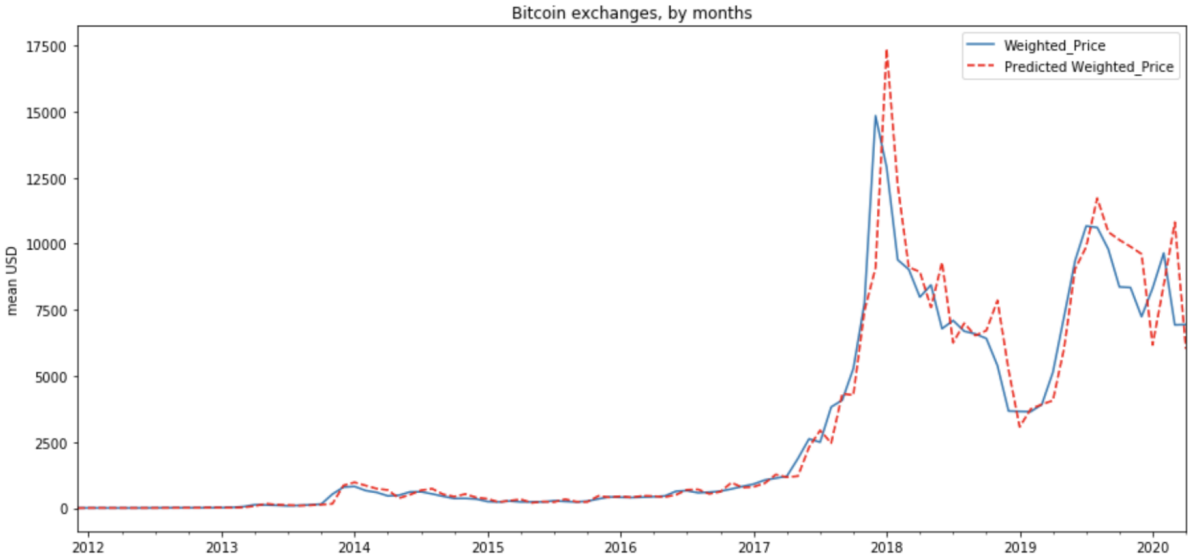


Fig. 18. SARIMA - Forecast VS Actual

In [51]:
```
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    return({'mape':mape, 'Accuracy':(1-round(mape,2))*100})
```

Fig. 19. Forecast Accuracy Function

In [52]:
```
acc =[]
for i in range(86,101):
    train = df_month2.Weighted_Price[:i]
    test = df_month2.Weighted_Price[i:]
    trainModel = model=sm.tsa.statespace.SARIMAX(train, order=(1, 1, 0),
                                    seasonal_order=(0, 1, 2, 12)).fit(disp=-1)
    fcast = trainModel.get_forecast(test.shape[0], alpha =0.5)
    fc = fcast.predicted_mean
    lower_series = fcast.conf_int()['lower Weighted_Price']
    upper_series = fcast.conf_int()['upper Weighted_Price']
    forecastResult = fc.to_frame()
    forecastResult['test'] = test
    forecastResult['error'] = (np.abs(fc - test)/np.abs(test))
    forecastResult = forecastResult.rename(columns = {0:'train'})
    res = forecast_accuracy(fc, test)
    acc.append(res['Accuracy'])
    print("train:",train.shape[0],"test:",test.shape[0], res)

acc
```

```
train: 86 test: 15 {'mape': 0.2683689878946185, 'Accuracy': 73.0}
train: 87 test: 14 {'mape': 0.2997295743613884, 'Accuracy': 70.0}
train: 88 test: 13 {'mape': 0.3042525141381508, 'Accuracy': 70.0}
train: 89 test: 12 {'mape': 0.27782817843709323, 'Accuracy': 72.0}
train: 90 test: 11 {'mape': 0.3568239856407437, 'Accuracy': 64.0}
train: 91 test: 10 {'mape': 0.4956811802189856, 'Accuracy': 50.0}
train: 92 test: 9 {'mape': 0.6932558367496333, 'Accuracy': 31.000000000000007}
train: 93 test: 8 {'mape': 0.6052282590468643, 'Accuracy': 39.0}
train: 94 test: 7 {'mape': 0.5485163128971591, 'Accuracy': 44.99999999999999}
train: 95 test: 6 {'mape': 0.2885520059921562, 'Accuracy': 71.0}
train: 96 test: 5 {'mape': 0.24838113676647477, 'Accuracy': 75.0}
train: 97 test: 4 {'mape': 0.23247108961488552, 'Accuracy': 77.0}
train: 98 test: 3 {'mape': 0.16713446977216898, 'Accuracy': 83.0}
train: 99 test: 2 {'mape': 0.41730001609650047, 'Accuracy': 58.00000000000001}
train: 100 test: 1 {'mape': 0.12854810464584682, 'Accuracy': 87.0}
```

Fig. 20. Multipe Train : Test Ratio with Accuracy

Avergae accuracy with mulitple ratios of train : test => 64.3%

```
In [53]: import statistics
         statistics.mean(acc)

Out[53]: 64.33333333333333
```

Fig. 21. Average Accuracy

Avergae accuracy for last 5 months : 76%

```
In [54]: statistics.mean(acc[-5:])

Out[54]: 76.0
```

Fig. 22. 5 Month Accuracy

```
In [59]: fcast = trainModel.get_forecast(15, alpha =0.5)
         fc = fcast.predicted_mean
         lower_series = fcast.conf_int()['lower Weighted_Price']
         upper_series = fcast.conf_int()['upper Weighted_Price']

         plt.figure(figsize=(12,5), dpi=100)
         plt.plot(train, label='training')
         plt.plot(test, label='actual')
         plt.plot(fc, label='forecast')
         plt.fill_between(lower_series.index, lower_series, upper_series,
                          color='k', alpha=.15)
         plt.title('Forecast vs Actuals')
         plt.legend(loc='upper left', fontsize=8)
         plt.show()
```

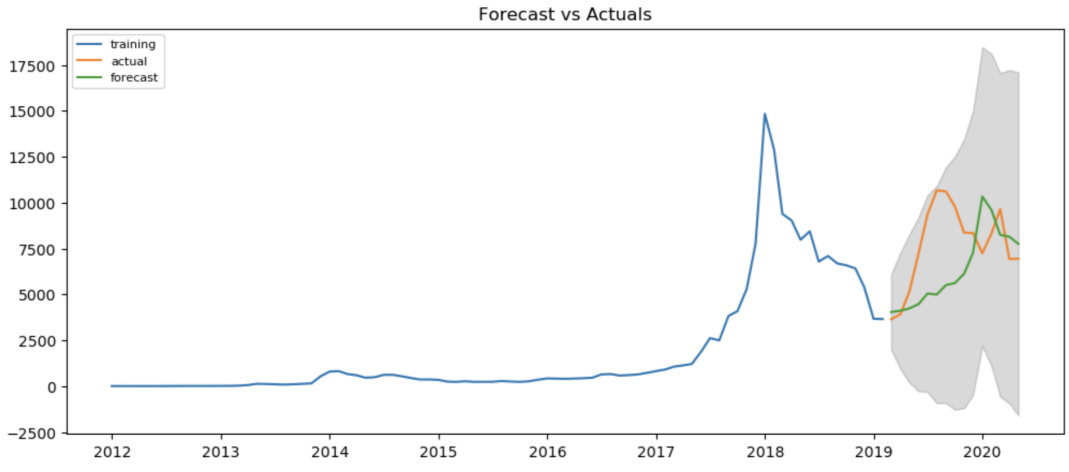


Fig. 23. SARIMA - Forecast VS Actual =¿ Forecasting last 15 months

In [60]:
```
forecastResult = fc.to_frame()
forecastResult['test'] = test
forecastResult['error'] = (np.abs(fc - test)/np.abs(test))
forecastResult = forecastResult.rename(columns = {0:'train'})
forecastResult
```

Out[60]:

	train	test	error
2019-02-28	4039.911882	3648.527128	0.107272
2019-03-31	4111.522662	3918.429353	0.049278
2019-04-30	4235.827787	5137.582914	0.175521
2019-05-31	4469.161205	7224.408577	0.381380
2019-06-30	5049.077396	9355.098647	0.460286
2019-07-31	4992.079557	10668.034536	0.532053
2019-08-31	5508.500634	10614.394673	0.481035
2019-09-30	5617.482254	9793.375249	0.426400
2019-10-31	6132.066399	8366.291324	0.267051
2019-11-30	7273.819483	8342.739513	0.128126
2019-12-31	10334.258199	7241.337904	0.427120
2020-01-31	9595.794042	8321.725828	0.153101
2020-02-29	8248.323424	9647.446186	0.145025
2020-03-31	8142.446184	6928.829208	0.175155
2020-04-30	7753.980644	6943.458255	0.116732

In [61]:
```
res = forecast_accuracy(fc, test)
res
```

Out[61]:
```
{'mape': 0.2683689878946185, 'Accuracy': 73.0}
```

Fig. 24. Last 15 forecasted value results