|  |
| --- |
| Bitcoin Price Prediction  Old Dominion University |
| INTRO TO DATA SCIENCE Varsha Ravi  [vravi002@odu.edu](mailto:vravi002@odu.edu) 01067335 |



**TABLE OF CONTENTS:**

|  |  |
| --- | --- |
| **TITLE** | **PAGE** |
| 1. INTRODUCTION | 3 |
| The target audience for the project | 4 |
| 1. tools/technologies | 4 |
| 1. DATASET DISCRIPTION | 5 |
| 1. DATA EXPLORATION | 5 |
| 1. Data mOdel | 11 |
| 1. System architecture and implementation | 13 |
| 1. REQUIREMENT/ DESIGN SATISFY THE NEEDS OF USERS | 18 |
| 1. CONCLUSION | 18 |
| 1. REFERENCES | 19 |

## BITCOIN PRICE PREDICTION

### **INTRODUCTION**

Bitcoin is a [cryptocurrency](https://en.wikipedia.org/wiki/Cryptocurrency), a form of electronic cash. It is a decentralized [digital currency](https://en.wikipedia.org/wiki/Digital_currency) without a [central bank](https://en.wikipedia.org/wiki/Central_bank) or single administrator that can be sent from user-to-user on the [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) bitcoin network without the need for intermediaries.

Transactions are verified by network [nodes](https://en.wikipedia.org/wiki/Node_(networking)) through [cryptography](https://en.wikipedia.org/wiki/Cryptography) and recorded in a public [distributed ledger](https://en.wikipedia.org/wiki/Distributed_ledger) called a [blockchain](https://en.wikipedia.org/wiki/Bitcoin#Blockchain). Bitcoin was invented by an unknown person or group of people using the name [Satoshi Nakamoto](https://en.wikipedia.org/wiki/Satoshi_Nakamoto)and released as [open-source software](https://en.wikipedia.org/wiki/Open-source_software) in 2009. Bitcoins are created as a reward for a process known as [mining](https://en.wikipedia.org/wiki/Bitcoin#Mining). They can be exchanged for other currencies, products, and services. Research produced by the [University of Cambridge](https://en.wikipedia.org/wiki/University_of_Cambridge) estimates that in 2017, there were 2.9 to 5.8 million unique users using a cryptocurrency wallet, most of them using bitcoin.

Bitcoin has been criticized for its use in illegal transactions, its high electricity consumption, price volatility, thefts from exchanges, and the possibility that bitcoin is an [economic bubble](https://en.wikipedia.org/wiki/Economic_bubble). Bitcoin has also been used as an investment, although several regulatory agencies have issued investor alerts about bitcoin.

Forecasting the value of bitcoin as forecasting the flow of stock market. Over the years many algorithms have been developed for forecasting stock markets, but very few have focused on bitcoin price prediction. The goal of this project progress is to provide an easy introduction to cryptocurrency analysis using Python. We used a simple Python script to retrieve, analyze, and visualize data on different cryptocurrencies. In the process, we uncover an interesting trend in how these volatile markets behave, and how they are evolving. This time we only procured the raw data and uncovered the stories hidden in the numbers.

Forecasting is a common statistical task in business, where it helps to inform decisions about the scheduling of production, transportation and personnel, and provides a guide to long-term strategic planning. The appropriate forecasting methods depend largely on what data are available.

If there are no data available, or if the data available are not relevant to the forecasts, then qualitative forecasting methods must be used.

Despite all the effort a detailed analysis of the forecasting performances of different models to this series has not been provided yet. We have tried to fill this gap and compares a large set of different models for point and density forecasting of the most capitalized cryptocurrencies, precisely, Bitcoin.

This project is part of my last semester's data mining project and it helped me learn a great deal about various forecasting techniques involved with stock market predictions. Though we tried out a variety of forecasting techniques including SMA, LSTM (deep learning) and random forest. The reason for this is due to the fact each forecasting technique to be used depends entirely on the data set.

#### The target audience for the project

Bitcoin is more accessible, with more exchanges, more merchants, more software and more hardware that support it. Bitcoin has two things going for it that help significantly in this respect — Stability and entrepreneurship. It has the most entrepreneurs creating companies around it with a lot of intellect, dedication and creativity going toward making it more useful.

As Bitcoin evolves, we can expect Bitcoin to grow in unexpected ways as new utility is found. Bitcoin owners can expect that its usefulness will only increase over time, hence creating a huge opportunity for investment and make huge profits. So, the target Audience for this prediction model will be people who buy, sell bitcoins i.e. merchants and the entrepreneurs who are trying to establish a business out of it.

1. **What tools/technologies will you use and why?**

When to invest and how much to invest is questionable and hence we have built this model to help predict the best time to invest. Just like most currencies, the price of Bitcoin changes every day.

The only difference is that the price of Bitcoin changes on a much greater scale than local currencies.

We will use

* **Python:** Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic memory management. It supports the multiple programing paradigms, including object oriented, imperative, functional and procedural, and has a large and standard library.
* **Jupyter Notebook:** The jupyter notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
* **Pandas:** Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the [Python](https://www.python.org/) programming language. pandas is a [NumFOCUS](https://www.numfocus.org/open-source-projects.html) sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to [donate](https://pandas.pydata.org/donate.html) to the project.
* **Matplotlib:** Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) notebook, web application servers, and four graphical user interface toolkits.
* **Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics
* Matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments
* **SciPy:** SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.
* **Stasmodels:** Statsmodels is a Python package that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator. It complements SciPy's stats module.

Statsmodels is part of the scientific Python stack that is oriented towards data analysis, data science and statistics. Statsmodels is built on top of the numerical libraries NumPy and SciPy, integrates with Pandas for data handling and uses Patsy for an R-like formula interface. Graphical functions are based on the Matplotlib library. Statsmodels provides the statistical backend for other Python libraries. Statmodels is free software released under the Modified BSD (3-clause) license.

* **Itertools:** itertools is a module for the Python language which contains high level functional constructs for working with iterable objects.

1. **DATASET DISCRIPTION**

The Dataset files for select bitcoin exchanges for the time period of January 2012 to November 2018, with minute to minute updates of

* OHLC (Open, High, Low, Close)
* Volume in BTC (Volume is such an important metric when analyzing cryptos and it can help you in showing a coin’s direction. The volume of a token listed on CoinMarketCap is quite simple. It’s the amount of the coin that has been traded in the last 24 hours. volume underscores how many people are buying and selling the coin.) and indicated currency, and
* Weighted bitcoin price (Weighted averages are calculated for last price, bid and ask separately. These weighted averages are shown on our frontend as currency markets data. Of all globally recognized currencies, only around a dozen are shown in the market area. This is because we recognize as markets only currencies that are directly traded to BTC anywhere in the world).
* Timestamps are in Unix time. Timestamps without any trades or activity have their data fields forward filled from the last valid time period. 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 unforseen technical error in data reporting or gathering. All effort has been made to deduplicate entries and verify the contents are correct and complete to the best of my ability.

1. **DATA EXPLORATION:**

Before we build the model, we have obtained some data for it. The dataset used is the minute by minute Bitcoin prices for the last few years. To make the predictions, we familiarized ourselves with techniques like ARMA, ARIMA, Recurrent Neural Network (RNN) with prediction and time series analysis as our main objectives. An ARIMA model is a class of statistical models for analyzing and forecasting time series data.

Unix time (also known as POSIX timeor UNIX Epoch time) is a system for describing a point in [time](https://en.wikipedia.org/wiki/Time). It is the number of [seconds](https://en.wikipedia.org/wiki/Second) that have elapsed since 00:00:00 [Coordinated Universal Time](https://en.wikipedia.org/wiki/Coordinated_Universal_Time) (UTC), Thursday, 1 January 1970, minus [leap seconds](https://en.wikipedia.org/wiki/Leap_seconds). Every day is treated as if it contains exactly 86400 seconds, so [leap seconds](https://en.wikipedia.org/wiki/Leap_second) are to be subtracted since the Epoch.It is used widely in [Unix-like](https://en.wikipedia.org/wiki/Unix-like) and many other operating systems and file formats. However, Unix time is not a true representation of UTC, as a leap second in UTC shares the same Unix time as the second which came before it. Unix time may be checked on most Unix systems by typing date +%s on the command line. We are converting to normal date by datetime library python function.

Resampling involves changing the frequency of your time series observations.

Two types of resampling are:

* Upsampling: Where you increase the frequency of the samples, such as from minutes to seconds.
* Downsampling: Where you decrease the frequency of the samples, such as from days to months.

In both cases, data must be invented.

In the case of upsampling, care may be needed in determining how the fine-grained observations are calculated using interpolation. In the case of downsampling, care may be needed in selecting the summary statistics used to calculate the new aggregated values.

There are perhaps two main reasons why you may be interested in resampling your time series data:

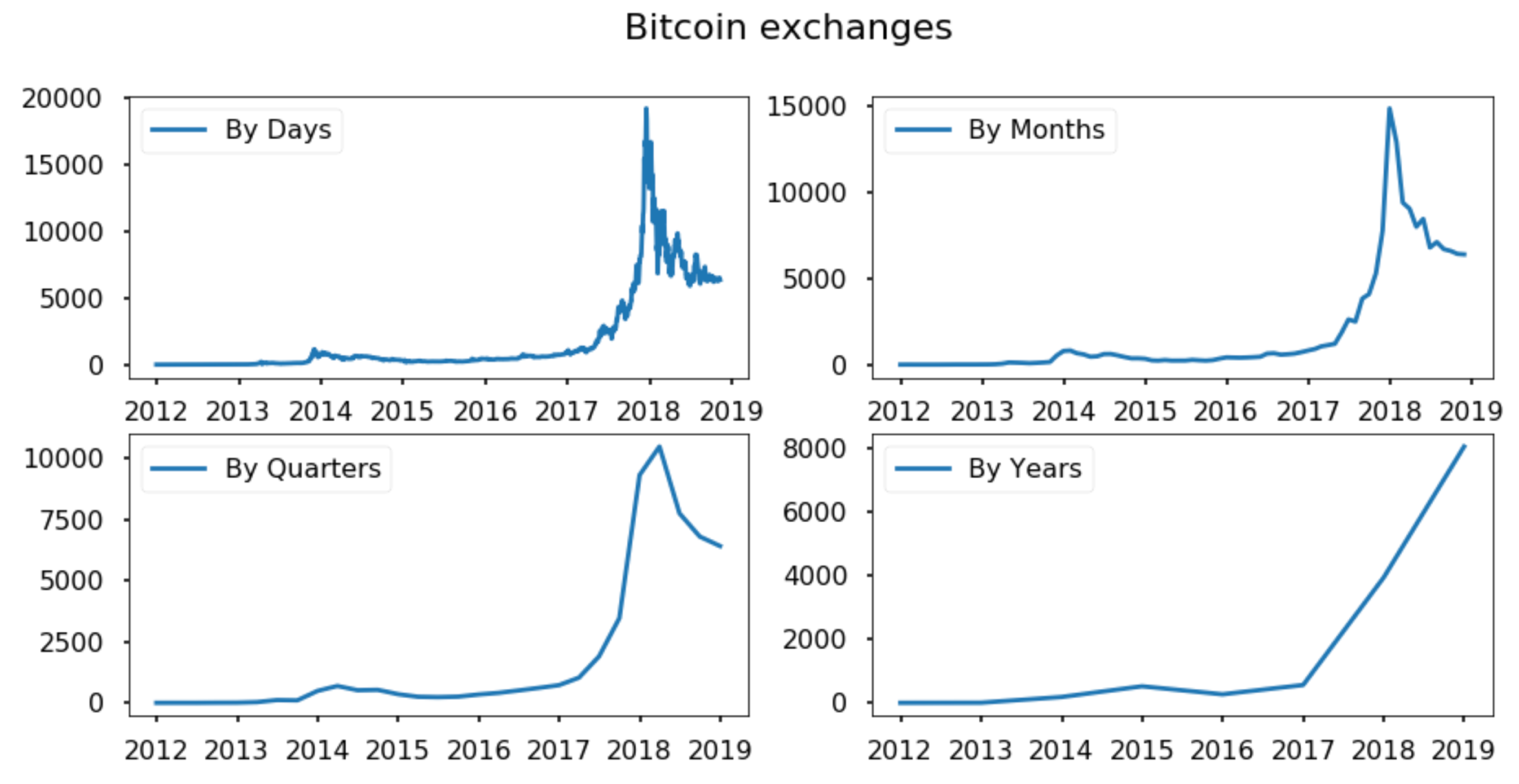
* Problem Framing: Resampling may be required if your data is available at the same frequency that you want to make predictions.
* Feature Engineering: Resampling can also be used to provide additional structure or insight into the learning problem for supervised learning models.

There is a lot of overlap between these two cases.

For example, you may have daily data and want to predict a monthly problem. You could use the daily data directly or you could downsample it to monthly data and develop your model.

A feature engineering perspective may use observations and summaries of observations from both time scales and more in developing a model.

**5.1 VIEWING DATASET WITH RESPECT TO WEIGHTED PRICE:**



As we can observe in the above diagram, we have visualized the weighted price in 4 different graphs.

The first being based in the days, which gives 365 points per year and 365 X 7 years, because we are considering past 7 years data,

The second was by months, which gave a pretty decent curve according to the weighted price.

Thirdly we have the done by quarterly and lastly we have yearly trend. When we considered all the 4 graphs, we decided on predicting the price according to months.

## 5.2 STATIONARITY CHECK AND STL-DECOMPOSITION OF THE SERIES

## 

So far we have explained that stationarity (second order or strict) is about imposing constancy of certain time series quantities. Why is this a useful concept? Certainly, much data seem to obey this rule in that future statistical behaviour is identical to past behaviour.

On the other hand, much data is **not** stationary or at least only approximately stationary. A real problem is that although there are tests for stationarity we submit that they are not used much in practice. Why is this?

Why do analysts persist with stationary models that are not appropriate and potentially risky? We offer four reasons. 1. Fear of diversity. There is a single mathematical model (the Fourier-Cramer model) for stationary time series. For nonstationary series the situation can be complex and the diversity of potential models can be daunting. 2. Education. Many undergraduate time series courses only have time or the ambition to consider stationary models, and 3. Mathematical expediency. Stationary models are mathematically easier to study and develop asymptotic theories for (that is, mathematically we understand how our modelling works for larger and larger samples). 4. Maturity. Stationary theory is mature, widely known and widely applicable.

However, it is the case that many real time series are just not stationary. Series often display trends (invalidating first-order stationarity), or seasonalities, or changes in variance (invalidating second-order stationarity). Hence, statistics and related fields have a second armoury of techniques that can manipulate time series to become stationary (differencing, variable transformations such as taking logs or square roots). After manipulation the series can be treated as stationary and standard methods used.

Time series decomposition is a mathematical procedure which transforms a time series into multiple different time series. The original time series is often split into 3 component series:

**Seasonal**: Patterns that repeat with a fixed period of time. For example, a website might receive more visits during weekends; this would produce data with a seasonality of 7 days.

**Trend:** The underlying trend of the metrics. A website increasing in popularity should show a general trend that goes up.

**Random:** Also call “noise”, “irregular” or “remainder,” this is the residuals of the original time series after the seasonal and trend series are removed.

By the above figure we got to know that the data is not stationary, to get the data to stationary form we performed different medoths.

Given that the classical modeling approaches assume stationarity, we check for it using the Augmented Dickey-Fuller (DF) Test and the Kwiatkowski-Phillips-Schmidt-Shin test Both are hypothesis tests where DF assumes non-stationarity as the null hypothesis while KPSS assumes the opposite.

They reveal that the Bitcoin price time series is not stationary. However, as is often observed with econometric data, the time series produced by the first-differences, i.e. y[t] = p[t] − p[t − 1], is stationary (p-value < 0.01 for DF; p-value > 0.1 for KPSS).

In [statistics](https://en.wikipedia.org/wiki/Statistics), the Dickey–Fuller test tests the [null hypothesis](https://en.wikipedia.org/wiki/Null_hypothesis) that a [unit root](https://en.wikipedia.org/wiki/Unit_root) is present in an [autoregressive](https://en.wikipedia.org/wiki/Autoregressive) model. The [alternative hypothesis](https://en.wikipedia.org/wiki/Alternative_hypothesis) is different depending on which version of the test is used, but is usually [stationarity](https://en.wikipedia.org/wiki/Stationarity_(statistics)) or [trend-stationarity](https://en.wikipedia.org/wiki/Trend_stationary). It is named after the [statisticians](https://en.wikipedia.org/wiki/Statistician) [David Dickey](https://en.wikipedia.org/wiki/David_Dickey) and [Wayne Fuller](https://en.wikipedia.org/wiki/Wayne_Fuller), who developed the test in 1979.

A simple [AR](https://en.wikipedia.org/wiki/Autoregressive)(1) model is



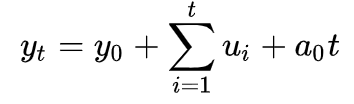
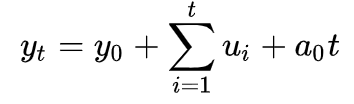
Each version of the test has its own critical value which depends on the size of the sample. In each case, the [null hypothesis](https://en.wikipedia.org/wiki/Null_hypothesis) is that there is a unit root {\displaystyle \delta =0}.

The tests have low [statistical power](https://en.wikipedia.org/wiki/Statistical_power) in that they often cannot distinguish between true unit-root processes {\displaystyle \delta =0}and near unit-root processes This is called the "near observation equivalence" problem.

The intuition behind the test is as follows. If the series {\displaystyle y} is [stationary](https://en.wikipedia.org/wiki/Stationary_process) (or [trend stationary](https://en.wikipedia.org/wiki/Trend_stationary)), then it has a tendency to return to a constant (or deterministically trending) mean.

Therefore, large values will tend to be followed by smaller values (negative changes), and small values by larger values (positive changes). Accordingly, the level of the series will be a significant predictor of next period's change, and will have a negative coefficient.

If, on the other hand, the series is integrated, then positive changes and negative changes will occur with probabilities that do not depend on the current level of the series; in a [random walk](https://en.wikipedia.org/wiki/Random_walk), where you are now does not affect which way you will go next.

{\displaystyle \Delta y\_{t}=a\_{0}+u\_{t}\,}

## 5.3 BOX-COX TRANSFORMATIONS

A Box Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn’t normal, applying a Box-Cox means that you are able to run a broader number of tests.

The Box Cox transformation is named after statisticians George Box and Sir David Roxbee Cox who collaborated on a 1964 paper and developed the technique.

At the core of the Box Cox transformation is an exponent, lambda (λ), which varies from -5 to 5. All values of λ are considered and the optimal value for your data is selected; The “optimal value” is the one which results in the best approximation of a normal distribution curve.

* 1. **SEASONAL DIFFERENTIATION**

The (P,D,Q,s) order of the seasonal component of the model for the AR parameters, differences, MA parameters, and periodicity.

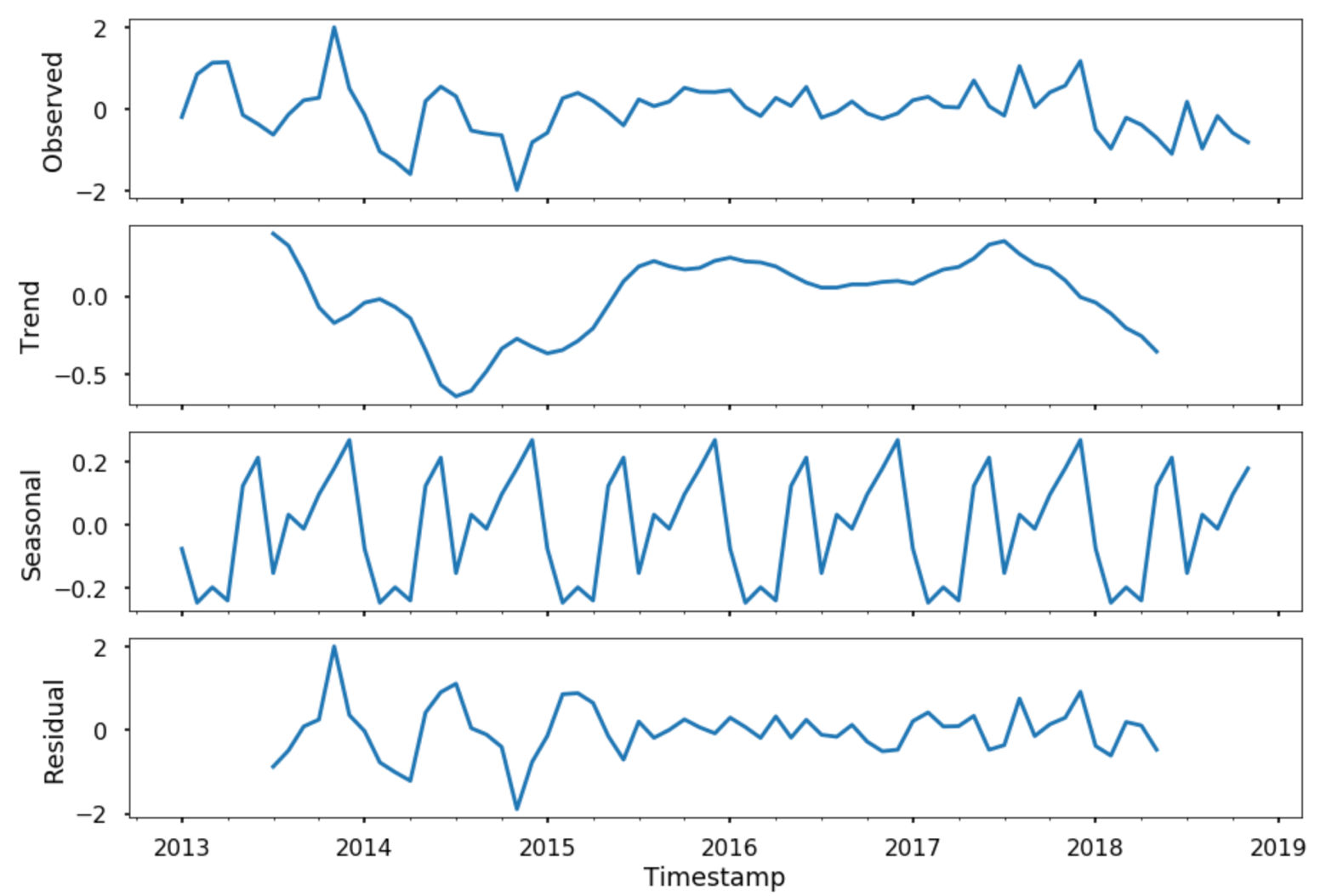
d must be an integer indicating the integration order of the process, while pand q may either be an integers indicating the AR and MA orders (so that all lags up to those orders are included) or else iterables giving specific AR and / or MA lags to include.

s is an integer giving the periodicity (number of periods in season), often it is 4 for quarterly data or 12 for monthly data. Default is no seasonal effect.

The Akaike information criterion (AIC) is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data,AIC estimates the quality of each model, relative to each of the other models. Thus,AIC provides a means for model selection.

In both the cases the stationarity failed so, tried one last method which. Is called the regular differenciation.

5.5 REGULAR DIFFERENTIATION

 we do is the examination of the autocorrelation. It is it is the

In the above figure we can see that the stationarity check was satisfied.

Time series datasets may contain trends and seasonality, which may need to be removed prior to modeling.

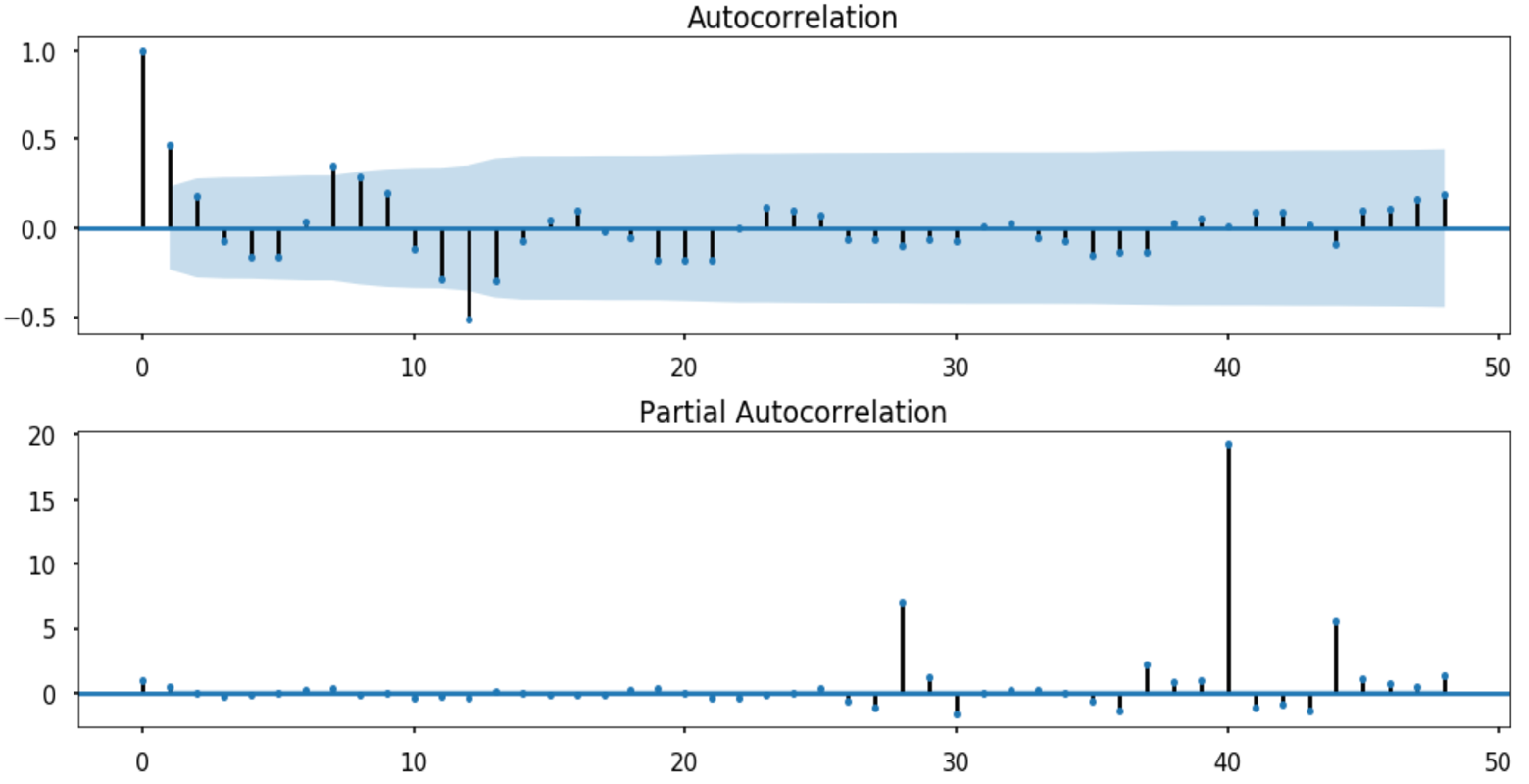
Trends can result in a varying mean over time, whereas seasonality can result in a changing variance over time, both which define a time series as being non-stationary. Stationary datasets are those that have a stable mean and variance, and are in turn much easier to model.

Differencing is a popular and widely used data transform for making time series data stationary.

**6 Data mOdel**

Based on the previous works by other data scientists on bitcoins, we came up with the following algorithms to access the forecast are:

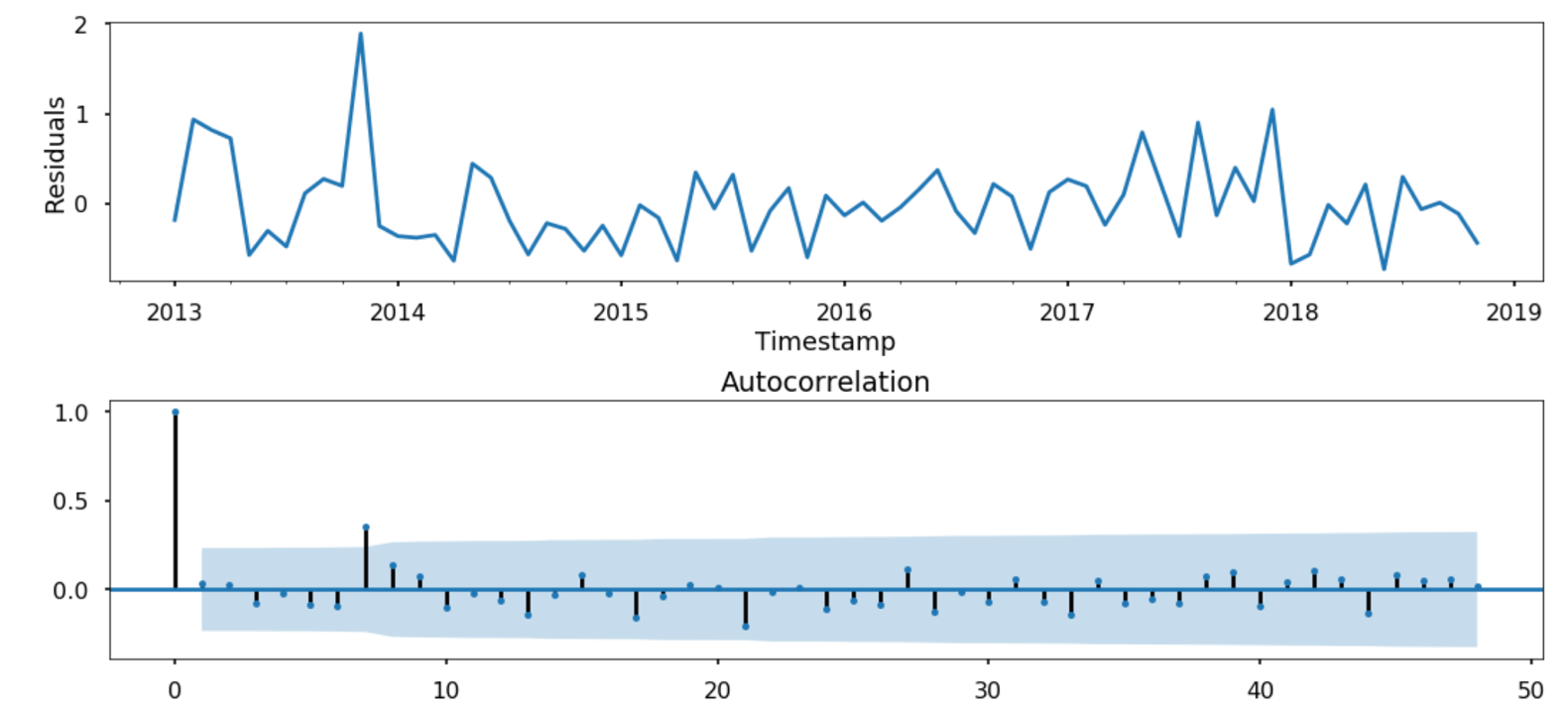
Initial approximation of parameters using Autocorrelation and Partial Autocorrelation Plots



Autocorrelation, also known as serial correlation, is the correlation of a signal with a delayed copy of itself as a function of delay. ... It is often used in signal processing for analyzing functions or series of values, such as time domain signals.

In time series analysis, the partial autocorrelation function (PACF) gives the partial correlation of a time series with its own lagged values, controlling for the values of the time series at all shorter lags. It contrasts with the autocorrelation function, which does not control for other lags.

**6.1 ANALYSIS OF RESIDUE**

****

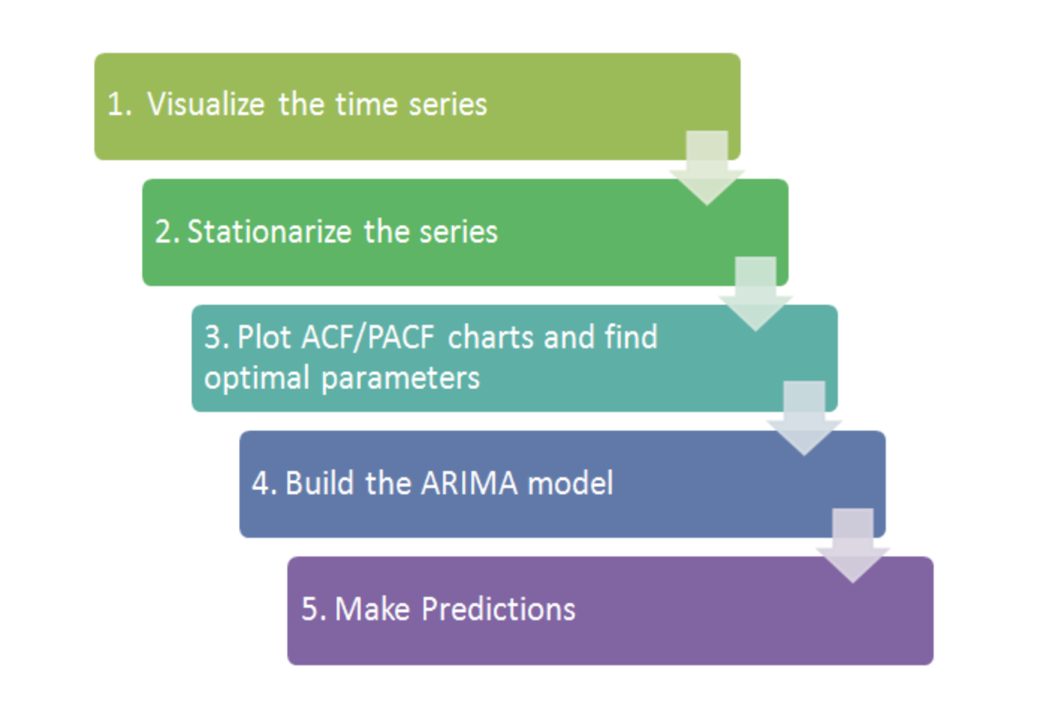
First, we get a line plot of the residual errors, suggesting that there may still be some trend information not captured by the model.

Note, that although above we used the entire dataset for time series analysis, ideally we would perform this analysis on just the training dataset when developing a predictive model.

**7 System architecture and implementation details?**

Our dataset contains time series of hourly volatility spanning more than one year, which outlines the fluctuation of Bitcoin market over time. We will use anaconda distribution version of python, which includes most of the libraries. Conda command will be used to install rest libraries.

**7.1 ARCITECTURE :**



An [ARIMA model](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average) is a class of statistical models for analyzing and forecasting time series data.

It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration.

This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

* **AR**: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
* **I**: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
* **MA**: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

* **p**: The number of lag observations included in the model, also called the lag order.
* **d**: The number of times that the raw observations are differenced, also called the degree of differencing.
* **q**: The size of the moving average window, also called the order of moving average.

A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model.

A value of 0 can be used for a parameter, which indicates to not use that element of the model. This way, the ARIMA model can be configured to perform the function of an ARMA model, and even a simple AR, I, or MA model.

Adopting an ARIMA model for a time series assumes that the underlying process that generated the observations is an ARIMA process. This may seem obvious, but helps to motivate the need to confirm the assumptions of the model in the raw observations and in the residual errors of forecasts from the model.

Next, let’s take a look at how we can use the ARIMA model in Python. We will start with loading a simple univariate time series.

The statsmodels library provides the capability to fit an ARIMA model.

An ARIMA model can be created using the statsmodels library as follows:

* Define the model by calling [ARIMA()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.html) and passing in the p, d, and q parameters.
* The model is prepared on the training data by calling the [fit()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.fit.html) function.
* Predictions can be made by calling the [predict()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.predict.html) function and specifying the index of the time or times to be predicted.

Let’s start off with something simple. We will fit an ARIMA model to the entire Shampoo Sales dataset and review the residual errors.

First, we fit an ARIMA(5,1,0) model. This sets the lag value to 5 for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 0.

When fitting the model, a lot of debug information is provided about the fit of the linear regression model. We can turn this off by setting the disp argument to 0.

The statsmodels library provides the capability to fit an ARIMA model.

An ARIMA model can be created using the statsmodels library as follows:

* Define the model by calling [ARIMA()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.html) and passing in the p, d, and q parameters.
* The model is prepared on the training data by calling the [fit()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.fit.html) function.
* Predictions can be made by calling the [predict()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMA.predict.html) function and specifying the index of the time or times to be predicted.
* Let’s start off with something simple. We will fit an ARIMA model to the entire Shampoo Sales dataset and review the residual errors.

First, we fit an ARIMA model. This sets the lag value for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 0.

When fitting the model, a lot of debug information is provided about the fit of the linear regression model. We can turn this off by setting the disp argument to 0.

Running the example prints a summary of the fit model. This summarizes the coefficient values used as well as the skill of the fit on the on the in-sample observations.

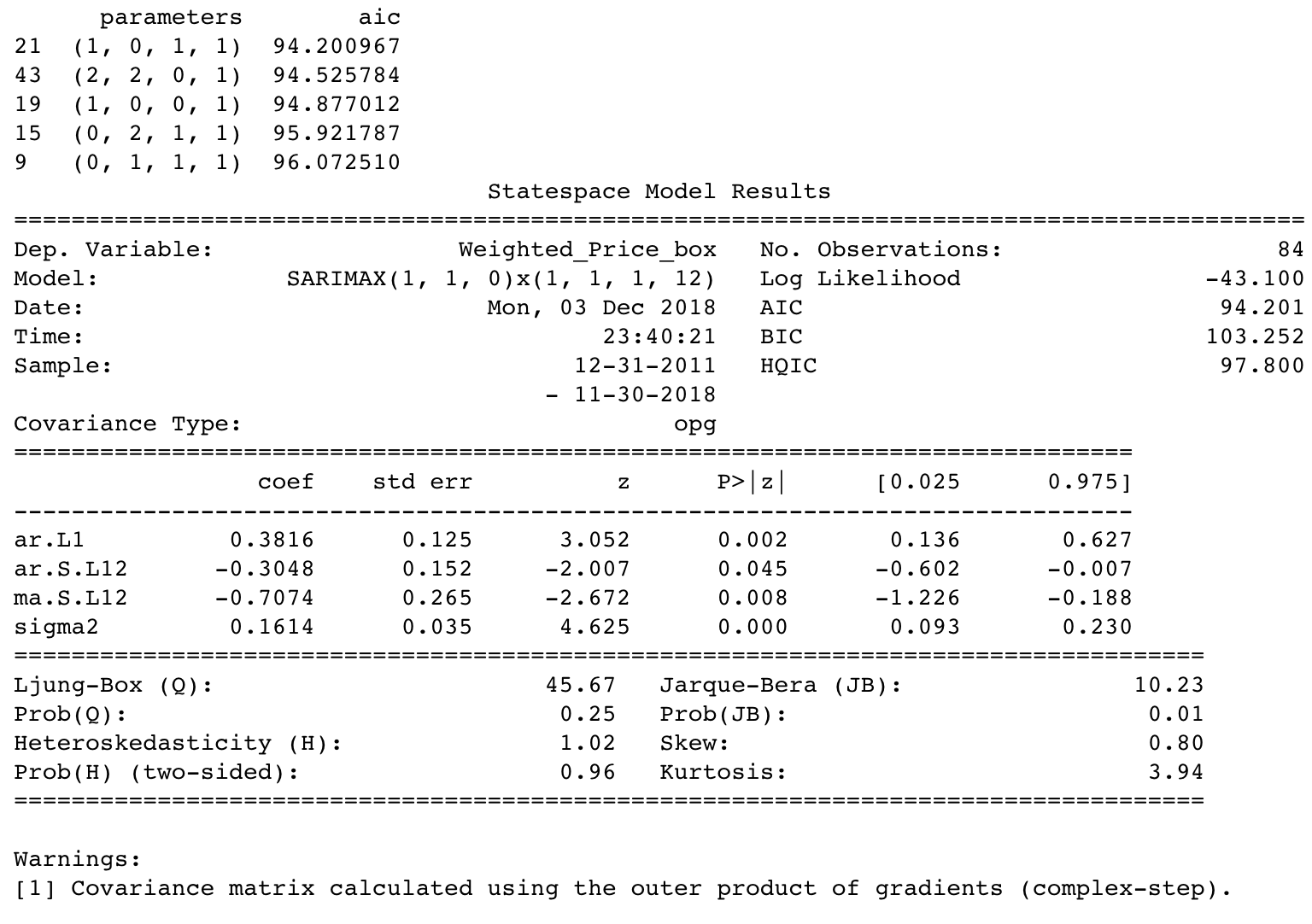
If the partial autocorrelation function (PACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is positive--i.e., if the series appears slightly "underdifferenced"--then consider adding one or more AR terms to the model.

The lag beyond which the PACF cuts off is the indicated number of AR terms.

If the autocorrelation function (ACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is negative--i.e., if the series appears slightly "overdifferenced"--then consider adding an MA term to the model.

The lag beyond which the ACF cuts off is the indicated number of MA terms.

**7.2 DATASET SUMMARY**



The ARIMA model can be used to forecast future time steps.

We can use the predict() function on the [ARIMAResults](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMAResults.html) object to make predictions. It accepts the index of the time steps to make predictions as arguments. These indexes are relative to the start of the training dataset used to make predictions.

If we used 100 observations in the training dataset to fit the model, then the index of the next time step for making a prediction would be specified to the prediction function as start=101, end=101. This would return an array with one element containing the prediction.

We also would prefer the forecasted values to be in the original scale, in case we performed any differencing (d>0 when configuring the model). This can be specified by setting the typargument to the value ‘levels’: typ=’levels’.

Alternately, we can avoid all of these specifications by using the [forecast()](http://statsmodels.sourceforge.net/devel/generated/statsmodels.tsa.arima_model.ARIMAResults.forecast.html) function, which performs a one-step forecast using the model.

We can split the training dataset into train and test sets, use the train set to fit the model, and generate a prediction for each element on the test set.

A rolling forecast is required given the dependence on observations in prior time steps for differencing and the AR model. A crude way to perform this rolling forecast is to re-create the ARIMA model after each new observation is received.

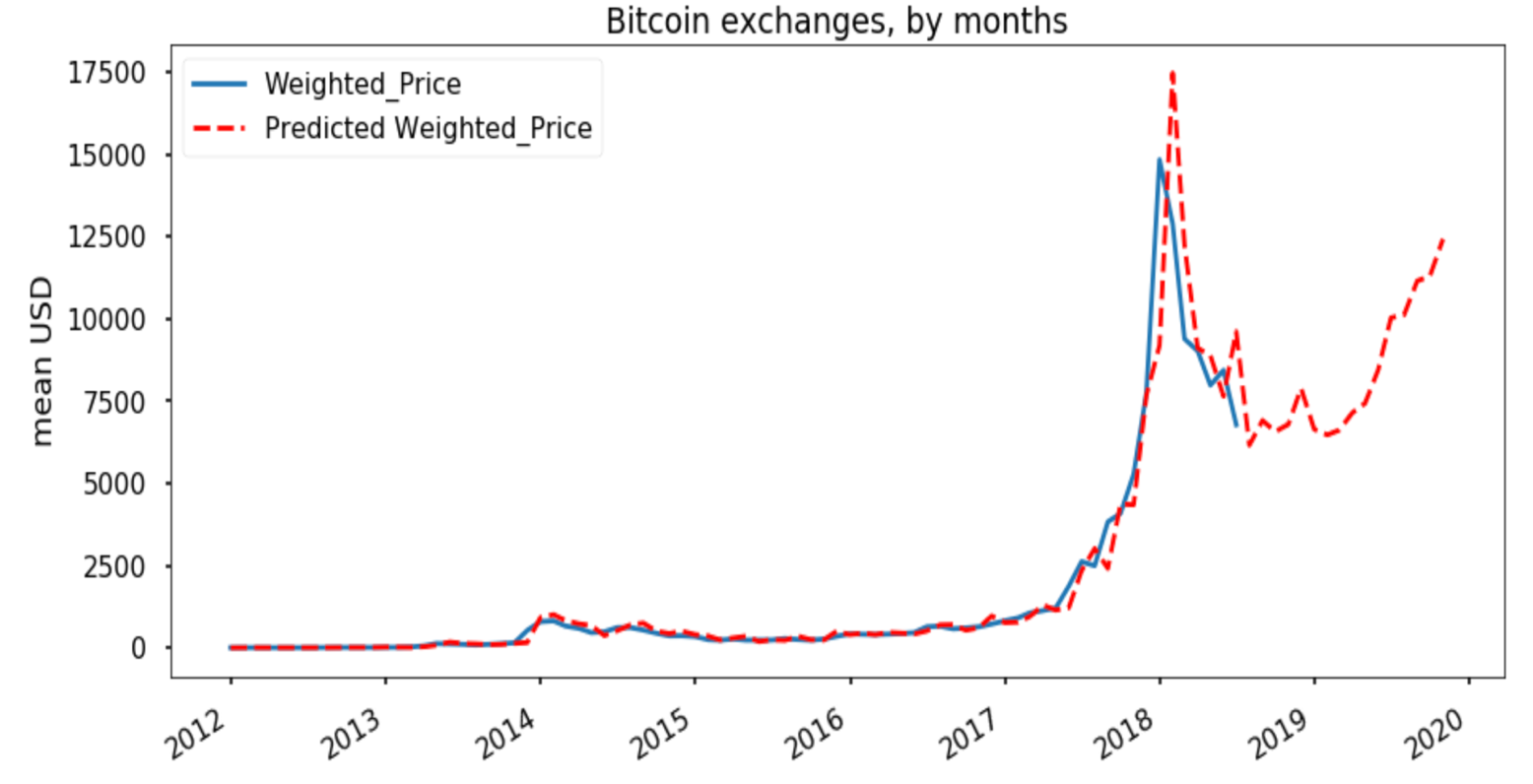
We manually keep track of all observations in a list called history that is seeded with the training data and to which new observations are appended each iteration.

Putting this all together, below is an example of a rolling forecast with the ARIMA model in Python.

Running the example prints the prediction and expected value each iteration.

We can also calculate a final mean squared error score (MSE) for the predictions, providing a point of comparison for other ARIMA configurations.

A line plot is created showing the expected values (blue) compared to the rolling forecast predictions (red). We can see the values show some trend and are in the correct scale



The classical approach for fitting an ARIMA model is to follow the [Box-Jenkins Methodology](https://en.wikipedia.org/wiki/Box%E2%80%93Jenkins_method).

This is a process that uses time series analysis and diagnostics to discover good parameters for the ARIMA model.

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

**8 . HOW DOES THE REQUIREMENT/ DESIGN SATISFY THE NEEDS OF USERS?**

The model will focus on the bitcoin use, that is, on the demand side of the bitcoin market. And bitcoin demand that has been growing slowly. Investing in crypto coins or tokens is highly speculative and the market is largely unregulated.

Any users considering it should be prepared and understand the risk involved rather than depending on just the forecast results.

**9. CONCLUSION**

In this, we have investigated bitcoin price prediction by using an ARIMA model. Towards this end, at first we have preprocessed data to make it stationary, and then, have searched over feasible (p, q, d) tuples for finding the ARIMA model which minimizes the MSE of prediction.

Our results indicate that the bitcoin price prediction using its weighted history could results in large values due to bitcoin’s price vulnerability to high jumps and fall-downs. on the other hand, the results confirm that the ARIMA model could be still used for price prediction in sub-periods of the timespan, i.e. by dividing the timespan to several time-spans over which, dataset has a unique trend.

Furthermore, we have investigated the impact of location of the test time window and its length on the achieved in price prediction.

**REFERENCES:**

[1] M. Amjad and D. Shah, “Trading bitcoin and online time series prediction,” in NIPS 2016 Time Series Workshop, 2017, pp. 1–15.

[2] Wikipedia. (2018) The world’s billionaires. [Online]. Available: <https://en.wikipedia.org>

[3] S. McNally, J. Roche, and S. Caton, “Predicting the price of bitcoin using machine learning,” in IEEE 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2018, pp. 339–343.

[4] H. Jang and J. Lee, “An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information,” IEEE Access, vol. 6, pp. 5427–5437, 2018. [5] P. J. Brockwell, R. A. Davis, and M. V. Calder, Introduction to time series and forecasting. Springer, 2002, vol. 2.

[6] J. Rebane, I. Karlsson, S. Denic, and P. Papapetrou, “Seq2Seq RNNs and ARIMA models for cryptocurrency prediction: A comparative study,” in rative study,” 2018