

**Department of Computer Science and Engineering**

**PES University,Bangalore**

Machine Learning Mini Project

BITCOIN PRICING

|  |  |
| --- | --- |
| Name | USN |
| Bhaarat Chetty | 01FB15ECS064 |
| DJ Pooja | 01FB15ECS083 |
| Devika Mishra | 01FB15ECS093 |
| Divya Jyothi | 01FB15ECS096 |

**PROBLEM STATEMENT**

Bitcoin is the longest running and most well known cryptocurrency. Cryptocurrencies are relatively unpredictable compared to traditional financial instruments. The increase/decrease in Bitcoin’s price with large percentages over short periods of time is an interesting phenomenon which cannot be predicted at all.This Project aims to predict the price of these Cryptocurrencies with Deep Learning using Bitcoin as an example so as to provide insight into the future trend of Bitcoin.

These predictions could be used as the foundation of a bitcoin trading strategy. To make these predictions, first take we will have to familiarize yourself with a machine learning techniques ARMA, ARIMA, Recurrent Neural Network (RNN) .

**Objectives**

* Prediction
* Time series analysis

**Dataset** **details**:  
Before we build the model, we need to obtain some data for it. There’s a dataset on Kaggle that details minute by minute Bitcoin prices for the last few years. Over this timescale, noise could overwhelm the signal, so we’ll opt for daily prices.

**a) Dataset name**

* coincheckJPY\_1-min\_data\_2014-10-31\_to\_2018-01-08.csv
* bitflyerJPY\_1-min\_data\_2017-07-04\_to\_2018-01-08.csv
* coinbaseUSD\_1-min\_data\_2014-12-01\_to\_2018-01-08.csv
* bitstampUSD\_1-min\_data\_2012-01-01\_to\_2017-01-08.csv

CSV files for select bitcoin exchanges for the time period of Jan 2012 to Jan 2018, with minute to minute updates of OHLC (Open, High, Low, Close), Volume in BTC and indicated currency, and weighted bitcoin price.

**b) Attributes count and Attributes list.**

There are 8 attributes.

* + - Timestamp (Unix time)
    - Open
    - High
    - Low
    - Close
    - Volume\_(BTC)
    - Volume\_(Currency)
    - Weighted\_Price

All the attributes are of numeric type.

**c) Class count and class list**

We are predicting continuous numerical values.

**d) Instance count**

3,161,057 instances approximately**.**

**ML techniques - Survey**

**ARMA: AutoRegressive Moving Average(ARMA)**

Forecasting model or process in which both autoregression analysis and moving average methods are applied : ARMA. It assumes that the time series is stationary-fluctuates more or less uniformly around a time-invariant mean. Non-stationary series need to be differenced one or more times to achieve stationarity. ARMA models are considered inappropriate for impact analysis or for data that incorporates random 'shocks.'

**ARIMA: Autoregressive Integrated Moving Average (ARIMA)**

The ARIMA model is a generalization of ARMA which are either used to study the data or predict/forecast the values. We prefer using ARIMA over ARMA as in case of non-stationary data the integrated part of ARIMA can help in predicting and studying patterns in a better way. As our data is completely a time series data, ARIMA model could be the best model for predicting the values linearly (Considering Moving Average) .

ARIMA is specified by these three order parameters: *(p, d, q)*. The process of fitting an ARIMA model is sometimes referred to as the Box-Jenkins method.

An **auto-regressive (AR(p))** component is referring to the use of past values in the regression equation for the series *Y*. The auto-regressive parameter *p* specifies the number of lags used in the model.For example, AR(2) or, equivalently, ARIMA(2,0,0), is represented as

where *φ*1, *φ*2 are parameters for the model.

The *d* represents the degree of differencing in the **integrated (***I(d)***)** component. Differencing a series involves simply subtracting its current and previous values *d* times. Often, differencing is used to stabilize the series when the stationarity assumption is not met, which we will discuss below.

A **moving average (MA(q))** component represents the error of the model as a combination of previous error terms *et*. The order *q* determines the number of terms to include in the model.

Differencing, autoregressive, and moving average components make up a non-seasonal ARIMA model which can be written as a linear equation:where *yd* is *Y* differenced *d* times and *c* is a constant.

ARIMA methodology does have its limitations. These models directly rely on past values, and therefore work best on long and stable series. Also note that ARIMA simply approximates historical patterns and therefore does not aim to explain the structure of the underlying data mechanism.

**RNN: Recurrent Neural Network (RNN)**

A Recurrent Neural Network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforwardneuralnetworks, RNNs can use their internal state (memory) to process sequences of inputs.As the dataset is huge the Neural net model might outperform ARIMA model.

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks.



The above diagram shows a RNN being *unrolled* (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence.

### **Training RNNs**

Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at t=4 we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). This doesn’t make a whole lot of sense yet. We should be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.

### **RNN Extensions**

Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model.

**LSTM networks** are quite popular these days and we briefly talked about them above. LSTMs don’t have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called *cells* and you can think of them as black boxes that take as input the previous state h_{t-1} and current input x_t. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies.

Why LSTM RNNs for time series data?

Time series prediction problems are a difficult type of predictive modeling problem.

Unlike regression predictive modeling, time series also adds the complexity of a sequence dependence among the input variables.

A powerful type of neural network designed to handle sequence dependence is called recurrent neural networks. The Long Short-Term Memory network or LSTM network is a type of recurrent neural network used in deep learning because very large architectures can be successfully trained

We also plan to compare the results given by both the algorithms.

**a) Architectural Design**

Libraries to be used:

1. Numpy
2. Pandas
3. Statsmodels
4. Matplotlib
5. Keras
6. Sklearn

Plan of execution for ARIMA:

* To divide the data into testing and training samples.
* Create multiple testing samples.
* For the ARIMA model, plot graphs based on daily, monthly and quarterly basis in order to determine trend, seasonality and residuals (using matplotlib)
* Choose the best option among days, months and quarters to build a model.
* Use inbuilt functions in statsmodels to build and train the model.
* Building the model would include:
* Identifying Trends, Seasonality and Residuals.
* Eliminating trends and making the data stationary.
* Removing seasonality and decomposing the data.
* Finally forecasting/ prediction using test data.

Plan of execution for RNN:

* Use the same training and testing sets created for ARIMA.
* A basic neural net model is built.
* A single time step of the input is supplied to the network i.e. xt is supplied to the network
* We then calculate its current state using a combination of the current input and the previous state i.e. we calculate ht. Current state formula: 
* Once all the time steps are completed the final current state is used to calculate the output yt.
* Basic Algorithm is to Initialize the neural net, add input layers , add output layer. Then compile the model and Fit RNN to the training data. Once the error rate and accuracy are found adjust the weights and try and improve the accuracy i.e. changing the type of optimizer in the code.

Initialization of NN: using regressor = sequential(... ) defined in keras.

Input layer: regressor.add(LSTM( ... ) )

Output layer: regressor.add( dense( ... ))

Compilation: regressor.compile( optimiser … )

* Fitting: regressor.fit (...)

Steps to be followed:

*1.Importing keras and its packages*

*2.Scaling using normalisation*

*3. Reshaping into required shape for Keras*

*4.Initialize the RNN*

*5.Adding input layer and the LSTM layer*

*6.adding output layers*

*7.compiling the RNN*

*8.fitting the RNN to the training set*

*OUTPUT:*

*Epoch 61/100*

*2694/2694 [==============================] - 0s 104us/step - loss: 1.1604e-10*

*Epoch 62/100*

*2694/2694 [==============================] - 0s 104us/step - loss: 1.2234e-10*

*Epoch 63/100*

*2694/2694 [==============================] - 0s 110us/step - loss: 1.1954e-10*

*Epoch 64/100*

*2694/2694 [==============================] - 0s 99us/step - loss: 1.1891e-10*

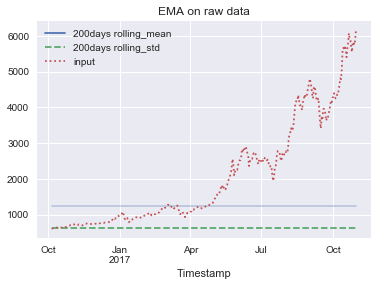
*Epoch 65/100*

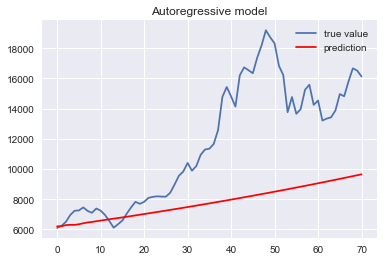
*9.Getting the predicted BTC value of the first week of Dec 2017 10.Visualising the result.*

**Results**

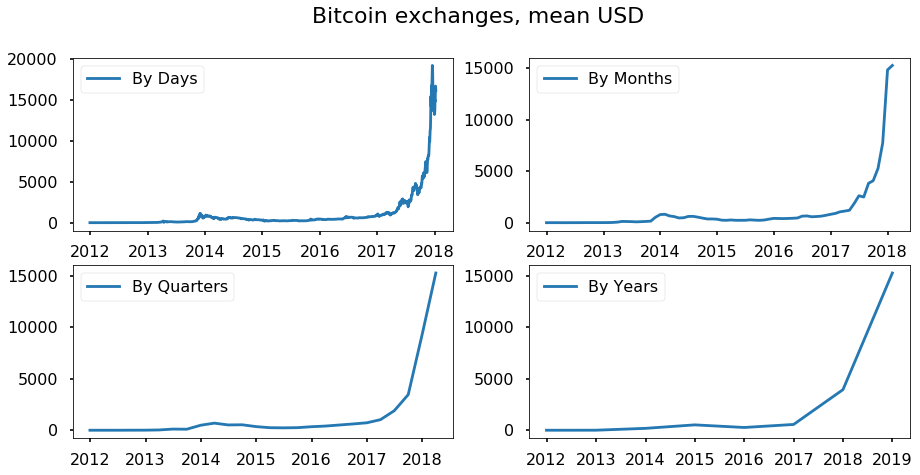
ARMA :

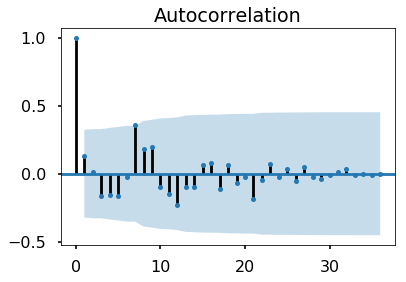


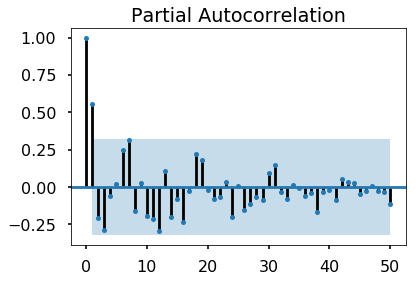


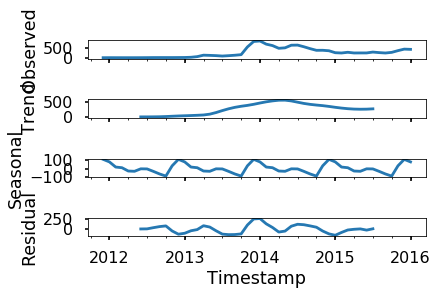


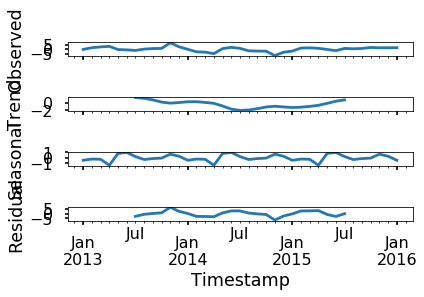
ARIMA

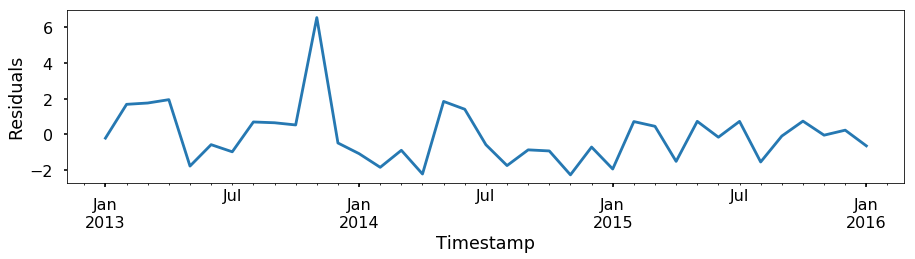
****

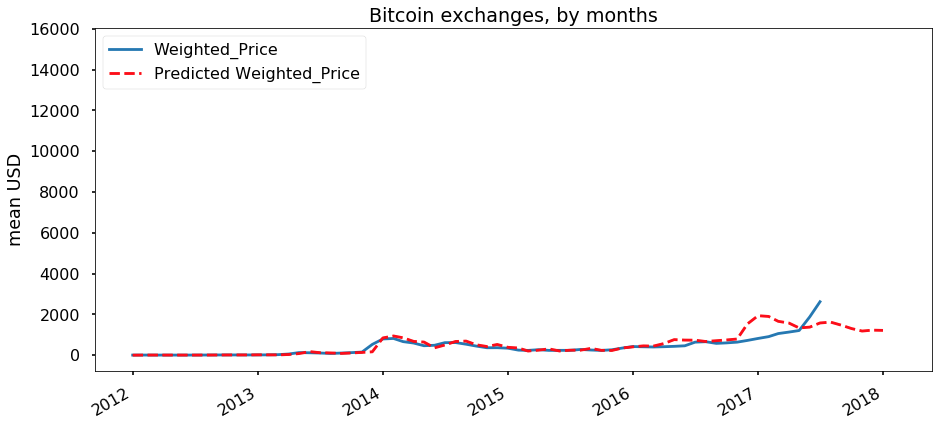
****

****

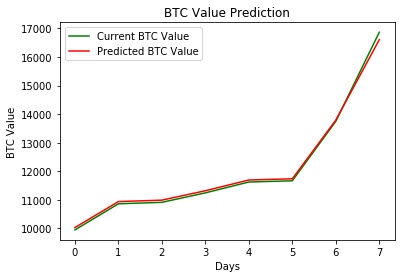
****

****

****

****

* RNN



On applying we can observe that actual and predicted price is almost similar. Prediction of the bitcoin price with respect to days yield better results

**Concluding Remarks**

* Bitcoin is a successful cryptocurrency, and it has been extensively studied in fields of economics and computer science. In this study, we analyze the time series of Bitcoin price with ARIMA, Recurrent Neural Network (RNN) andBayesian Regression, using Blockchain information in addition to macroeconomic variables and address the recent highly volatile Bitcoin prices.
* We wrote a custom algorithm to hopefully predict future prices for all of our listed digital cryptocurrencies similar to Bitcoin. If you are looking for crypto currencies with a good return on your investment, BTC could potentially be a profitable investment option for you.

You have learned:

1. How to gather real-time Bitcoin data.
2. How to prepare data for training and testing.
3. How to predict the price of Bitcoin using Deep Learning.
4. How to visualize the prediction result.

References

* [https://ieeexplore.ieee.org/ielx7/6287639/8274985/08125674.pdftp=&arnumber=8125674&isnumber=8274985](https://ieeexplore.ieee.org/ielx7/6287639/8274985/08125674.pdf?tp=&arnumber=8125674&isnumber=8274985)
* <https://arxiv.org/ftp/arxiv/papers/1302/1302.6613.pdf>
* <https://www.kaggle.com/myonin/bitcoin-price-prediction-by-arima/data>
* <https://www.kaggle.com/mczielinski/bitcoin-historical-data>
* <https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/>

Research papers about RNNs:

* [A Recursive Recurrent Neural Network for Statistical Machine Translation](http://www.aclweb.org/anthology/P14-1140.pdf)
* [Sequence to Sequence Learning with Neural Networks](http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf)
* [Joint Language and Translation Modeling with Recurrent Neural Networks](http://research.microsoft.com/en-us/um/people/gzweig/Pubs/EMNLP2013RNNMT.pdf)