Bitcoin Price Prediction using ARIMA Model

Dr. Jinan Fiaidhi
Department of Computer Science
Lakehead University
jfiaidhi@lakeheadu.ca

Ahmer Sabah
Department of Computer Science
Lakehead University
sabaha@lakeheadu.ca

Mahpara Anwer Ansari
Department of Computer Science
Lakehead University
manwer@lakeheadu.ca

Zeba Ayaz
Department of Computer Science
Lakehead University
zayaz@lakeheadu.ca

Abstract —Bitcoin is considered to be most valuable and expensive currency in the world. Besides being first decentralized digital currency, its value has also experienced a steep increase, from around 1 dollar in 2010 to around 18000 in 2017. In recent years, it has attracted considerable attention in a diverse set of fields, including economics, finance and computer science. In economics, the primary focus has always been on studying how it affects the market, determining reasons behinds its price fluctuations, and predicting its future prices. In computer science, the focus is on its vulnerabilities, scalability, and other techno-cryptoeconomic issues. Firstly, we are going to collect the historical data of Bitcoin prices over the years 2013 to 2019 and do prediction for the year 2020. We have aimed to justify the usefulness of traditional Autoregressive Integrative Moving Average (ARIMA) model for predicting bitcoin prices. We have predicted the closing price of bitcoin for first seven days of January 2020. Further, we have created web services using ASP.NET to make the predictions on bitcoin price online and lastly, we have plotted the results in a responsive chart using Highcharts.

Keywords — Bitcoin(BTC), ARIMA, Highcharts, Web Services.

I. Introduction

Time series prediction is not a new phenomenon. Prediction of mature financial markets such as the stock market has been researched at length[7][8]. BTC presents an interesting proof to this as it is a time series prediction problem in a market still in its transient stage. As a result, it is highly unpredictable in the market[9] and this provides an opportunity in terms of prediction. A BTC and Blockchain are one of the latest technologies that have a huge impact on the banking industry. Due to the open nature of BTC it also poses another example as opposed to traditional financial markets. BTC is in the center of attention lately, and everybody talks about how its price raised rapidly in the end of 2017, but only a small proportion of people would know that there is a need for comprehensive studies about the possibilities it can bring in the next few years to be prepared for the risks the technology involves. Blockchain technology which is the backbone of BTC is basically an encrypted database of transactions (ledger), shared among all participants of a public network and therefore also verified by the users. Each transaction has digital information called “block” which is stored in database called “chain” which is encrypted. It is known as a distributed open ledger that keep tracks of transactions between two parties efficiently and in a verifiable and permanent way. There are three primary types of blockchains namely, public blockchain like BTC and Ethereum, private blockchain like Hyperledger and R3 Corda and hybrid blockchain like Dragonchain. In Public Blockchain all transactions are fully transparent, which means anyone can examine or see the transaction details also are fully decentralized

which means no individual or entity can have control over transactions which are recorded in the blockchain or the order in which they have been processed. Another type of blockchain are Private Blockchains, also known as Permissioned Blockchains, where the transactions are private and secured as they are only available to ecosystem participants that have been given permission to join the network. Dragonchain has a unique status within the blockchain ecosystem in that it's a hybrid blockchain. This means that it combines the privacy benefits which private blockchain offers with the security and transparency benefits of a public blockchain. That gives businesses a significant flexibility to choose what data they want to make public and what data they want to keep private. BTC is a peer-to-peer, fully decentralized crypto currency system designed to allow online users to process transactions through digital units of exchange called bitcoins. It is neither controlled nor regulated by any central authority making it to be decentralized. BTC payments are processed through a private network of computers linked together through a shared ledger. Traditional time series prediction methods such as Holt-Winters exponential smoothing models depend on linear assumptions and require data that could be broken down into trend, seasonal and noise to be effective[10]. This type of methodology is preferably suitable for a task such as forecasting sales where seasonal effects are present. Since BTC market lacks seasonality and since it is unpredictable, these methods are not very effective for this task. Given the complexity of the task, Machine Learning provides technological solution based on its performance in similar areas.

In time series data analysis, ARIMA is widely used in forecasting BTC prices. This model is considered to be one of the easiest and effective machine learning algorithms to perform time series forecasting which is the combination of Auto Regression and Moving average. The AR part of ARIMA stands for Autoregression which is a time series model that uses observations from previous time steps as input to the regression equation to predict the value at the next time step. In simple terminology, it performs regression in previous time step t-1 to predict t. MA is an acronym which stands for moving average which is also called as rolling mean. Basically, we are calculating the simple average in a particular time frame and dividing it by the total number of time frames taken. There are various ways of visualizing the predicted closing price plots, the most affective one with higher quality would be high charts.

A web service is language, protocol and platform which is independent. It is Scalable (e.g. multiplying two numbers together to an entire customer relationship management system) and Programmable (encapsulates a task).It is based on XML (open, text-based standard). Since it has ability to search for and locate desired web services through registries by any applications and developers.

II. Literature Review

Traditionally, the richest people in a society have an average age of more than 60 years[1]. The tremendous rise in the information and communications technology (ICT) have not only changed many of our beliefs, habits and traditions, but also has introduced a new wave of young billionaires. By the emergence of BTC in the year 2009 with an initial value of around one dollar, no one predicted that in 8 years it would pass all previous records and would reach to the unbelievable value of $18000. Due to an exponential increase in the price of BTC and its following price correction, i.e., steep down of the price after this huge rise, many

economists, mathematicians and computer scientists tried to explore the time series of the BTC price[2–4]. Our focus in this research is to investigate application of the traditional ARIMA model in prediction of the BTC prices and make it available to public by using Highcharts library using web services. ARIMA model has been widely used in the prediction of stationary datasets in the literature[5]and its references. It has been recently investigated, about application of ARIMA in prediction of BTC, and it has been shown that a Recurrent Neural Network model (RNN), augmented by additional data like Tweeter's hashtags can significantly outperform the performance, especially while talking about long-term predictions[6]. his stimulating discovery consists in the fact that in a very short span of time, ARIMA outperforms RNN, even though both work on a single input, i.e. the price of BTC. Regarding the explanation of results achieved by using ARIMA models and computational complexity associated with the neural networks, here we aim at further investigation of prediction of BTC price using ARIMA model for seven-day prediction of the BTC price. More specifically, we aim at investigating the seasonality in the BTC prices , the choice of ARIMA parameters (p, q, d), the length of time window that the prediction is carried out over it, i.e. the BTC price for the day after the window is predicted, and creating webservice in order to plot high charts for better visualization of predictions . The remainder of this paper is as follows. In the next section, we briefly present the system model and formulate the problem. In Section IV, we present our approach for solving the problem. Section V presents the performance evaluation results. Concluding remarks are given in last Section.

III. Methodology

In this section we are going to describe about proposed model, creation of web services and high charts.

A. Dataset Analysis

The dataset has been taken from the crypto-currencies website[13], it has record of all the cryptocurrencies available in the market.

We selected BTC data of 5 months from August 2019 to December 2019 as our training set. We tested our model on January 2020 data. It has seven features namely like Date, Open, High, Low, Close, Volume and Market capital of BTCs. The following figure shows the first five rows of the Dataset which is been sorted by Date.

Date	Open	High	Low	Close	Volume	Market Cap
Dec 31, 2019	7294.44	7335.29	7169.78	7193.60	21,167,946,112	130,446,112,598
Dec 30, 2019	7420.27	7454.82	7276.31	7293.22	22,874,131,672	132,235,128,152
Dec 29, 2019	7317.65	7513.95	7279.87	7422.65	22,445,257,702	134,570,835,775
Dec 28, 2019	7289.03	7399.04	7286.91	7317.99	21,365,673,026	132,659,059,740
Dec 27, 2019	7238.14	7363.53	7189.93	7290.09	22,777,360,996	132,139,502,950

Fig. 1: First five rows of the Training Data

Before we even start implementing our proposed ARIMA model it is important for us to understand the trend in BTC prices. Hence, we have obtained the plots of closing BTC prices over the year 2013 to 2019 for daily and yearly basis. As shown in Fig. 2a and 2b in the plots we can see that the BTC prices are constant from 2013 to 2016 after which there is an unusual trend. To understand that we should convert the data into time series and eliminate the trend

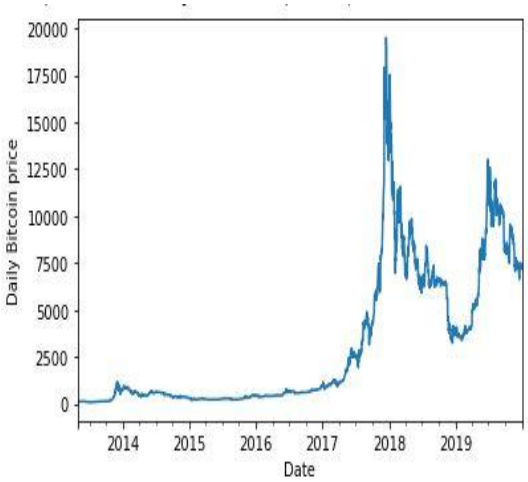


Fig. 2a : Daily BTC Price

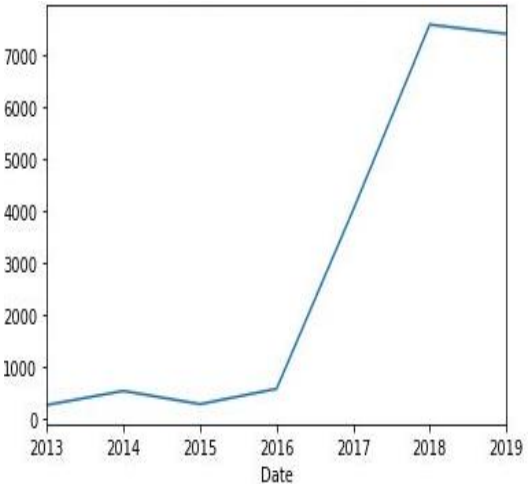


Fig. 2b: Yearly BTC Price

B. Seasonality Adjustment by Differencing

Seasonality is variations at specific timeframes which should be removed. As shown in Fig. 3, the plot of time series showsthat rolling mean value varies with time and is not stationary. The series needs to be made stationery. Hence, we need to eliminate the trend and seasonality from the series to make it stationary by using log function to transform the data. As shown in Fig. 4, there the results are not satisfactory, so we need to use one of the most common methods to deal with both trend and seasonality is differencing. In this approach, we take the difference of the observation at a specific interval of time with that at the previous instant. This mostly works well in improving stationarity. If there is a seasonal component at level of one month, then it can be removed on an observation today by subtracting the value from last month (Ex: Value (Oct 1)-Value (Sep 1), value (oct 2) - Value(Sep 2)...) We can subtract last month's data to the presents (giving a gap of 30 days) and first month's data would not be available for modeling. We are trying to find the difference in seasonality and test for stationery data in the following code.

```
timeSeriesDiffLogTransform
=
logTimeSeriesTransformed (-)
logTimeSeriesTransformed.shift
```

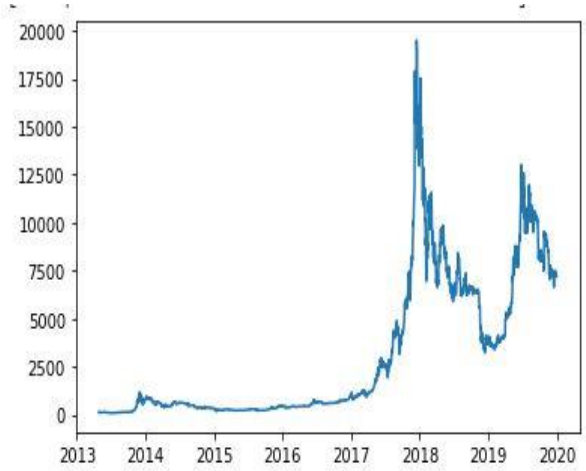


Fig 3: Plot of Time series

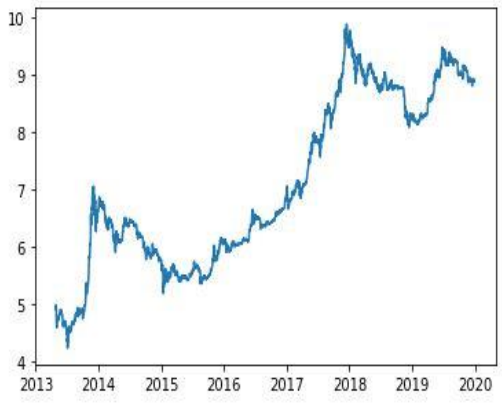


Fig 4: Plot of log transformed Data

At this stage we have made time series stationary, next part is to make a model on the time series after differencing using ARIMA. For implementing the ARIMA function we first need to find the values of parameters p, q and d. To determine the value of 'p' and 'q' we use two plots which are Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). In the plots of Fig. 5(a) & (b), the two dotted lines on either side of zero are the confidence intervals. These can be used to determine p and q values.

1. P – The lag value where the PACF chart crosses the upper confidence interval for the first time. If you notice clearly in this case p=2.
2. q – The lag value where the ACF chart crosses the upper confidence interval for the first time. In this case q= 18 since ACF shows significant lag for 18th day.

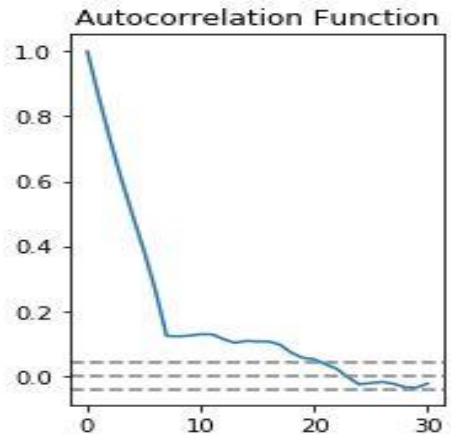


Fig 5(a): Plots of Auto Correlation Function

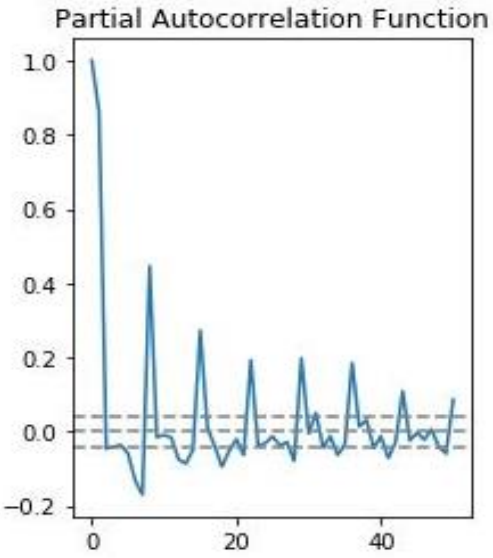


Fig 5(b): Plots of Partial Autocorrelation Function

IV. Proposed Model

To predict the BTC prices, we have modeled time series using ARIMA algorithm. Models with lower MSE are considered to be the ideal ones. First, we fit an ARIMA (2,1,0) model. This sets the lag value to 2 for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 0. Second, we try to fit an ARIMA (0,1,18) model. This sets the lag value to 0 for autoregression, uses a difference order of 1 to make the time series stationary, and uses a moving average model of 18. We tried many combinations of ARIMA for obtaining the lesser MSE value so that we can find the best fit for our model

```
model = ARIMA(logTimeSeriesTransformed, order=(8, 1, 0))

results = model.fit();
```

and finally it turned out to be 8, 1 and 0.

After fitting the model by passing its parameters we have loaded the testing dataset from the drive again in the same way how we read training dataset. In training we have stored the BTC data collected for January 1st until January 7th, 2020. The Fig.6, shows the

Bitcoin closing prices which collected for seven days of January which is considered as test data.

```
Date
2020-01-01    7200.17
2020-01-02    6985.47
2020-01-03    7344.88
2020-01-04    7410.66
2020-01-05    7411.32
2020-01-06    7769.22
2020-01-07    8163.69
Name: Close, dtype: float64
```

Fig 6: Testing Data

Finally, we are going to predict the bitcoin closing price and display the Mean Squared error which is the evaluation metric for our predicted Model. We assign the timestamp of the dates to the data frame dates and predict the bitcoin price for seven days using forecast function.

V. Experimental Settings and Results

In order to evaluate the proposed model, we have used Mean Squared Error score as the evaluation metric. As shown in Fig.7, with the results we can see that the predictions are at most similar to what we have in our training dataset with Mean Squared Error value coming to 170962.195 which can be reduced by trying different order values of parameters which are passed to ARIMA model function while filling the model. We have also plotted the forecast shown in Fig.8. In the graph we see that from 1st of January until 3rd there is a rise in the price of Bitcoin and later it steeps down and moving further we can see unusual trend.

We have implemented a web service on ASP.net which will read the csv data from the google drive which was stored by the python code. We have created a webservice in order to make the data usable for Highcharts library to create an interactive graph for bitcoin prices. There are various steps in creating a webservice which will be described in following paragraphs.

```
2020-08-01    7208.545141
2020-08-02    7227.884176
2020-08-03    7241.733856
2020-08-04    7248.969412
2020-08-05    7245.413912
2020-08-06    7251.022627
2020-08-07    7266.436262
dtype: float64
Test MSE: 170962.195
```

Fig 7: The predicted values with Mean Squared error

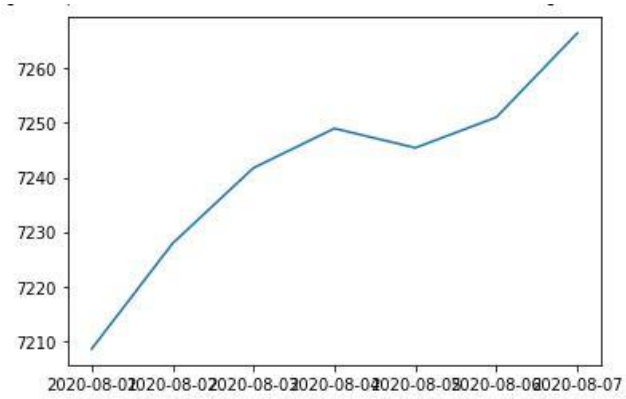


Fig 8: Plot of forecast

Webservice using ASP.net:

- Using Visual studio 2017 we create an ASP.Net web application.
- Then we create the resource classes for handling our GET, POST, PUT, and DELETE services.
- Finally, a model with key value pair is created.
- The result values show the key value pair which is data values of date and value of bitcoin price from google drive.

Web Service Implementation:

First, we implemented the Main controller which is a Web API class having a single Get() which returns the json response. Inside the Get() function we will call the helper class

```
UserCredential credential;
using (var stream =
new
FileStream(@"C:\BitcoinPrice\credentials.j
son", FileMode.Open, FileAccess.Read))
            {
// The file token.json stores the user's
access and refresh tokens, and is created
// automatically when the authorization
flow completes for the first time.
string credPath =
@"C:\BitcoinPrice\token.json";
                credential =
GoogleWebAuthorizationBroker.AuthorizeAsyn
c(

GoogleClientSecrets.Load(stream).Secrets,
                    Scopes,
"user",

CancellationToken.None,
new FileDataStore(credPath, true)).Result;

Console.WriteLine("Credential file saved
to: " + credPath);
            }

// Create Google Sheets API service.
var service = new SheetsService(new
BaseClientService.Initializer()
            {
                HttpClientInitializer =
credential,
                ApplicationName =
ApplicationName,
            });

// Define request parameters.
            String spreadsheetId =
"1QD_s42vDwmmWJsiDV2kF11buO22gliMkExCu-
QJxhp8";
            String range = "result!A1:B";

SpreadsheetsResource.ValuesResource.GetReq
uest request =

service.Spreadsheets.Values.Get(spreadshee
tId, range);

ValueRange response = request.Execute();
```

in which we implemented the code for reading excel data from the google drive.

After fetching data from the google drive, we will create a new object of response class in which we will save the data fetched form the google drive by iterating a for loop for each value. The excel sheet has predicted and actual values which will be save in the respective "Predicted" and "Actual" variables and then the result will be return to the Get() function in the Main Controller class.

```
IList<IList<Object>> values2 =
response2.Values;
if (values2 != null&&
values2.Count > 0)
{
result.Actual = new
List<KeyValuePair<string,
string>>();
foreach (var row in values2)
{
result.Actual.Add(new
KeyValuePair<string,
string>(row[0].ToString(),
row[4].ToString()));
}
}

result.Actual =
result.Actual.OrderBy(x =>
x.Key).ToList();

return result;
```

Price Prediction using High Charts

We have used Angular 7 to design our high charts. To use angular, we must install nodejs, followed by installation of Angular CLI. Using Js, the angular application development process becomes effortless. Node allows us to spin up a lightweight web server to host our application locally in our system.

Angular:

Angular is a platform and framework that enables us to build single-page client applications using HTML and TypeScript. All typescript libraries incorporate core and optimal functionalities that can be directly imported into our apps.

Like any other framework, Angular's architecture depends on a few fundamental concepts. The basic building blocks of its architecture are *NgModules*, which collect related code into functional sets; an Angular app can be defined by a set of NgModules. An app must have at least one *root module* to enable bootstrapping, and accordingly have one or multiple *feature modules*.

- Components are used to define *views*, which are parts of screen elements that Angular can choose among and alter according to the application logic and data.
- Components use *services*, which provide specific functionality to views. These services are added to the components as *dependencies* in order to increase code modularity, reusability and efficiency.

Modules:

NgModules are containers for the application's code block. It is used to bind the components and services used to create an Angular application into groups. NgModule defines the scope of components, service providers and other code files. Functionalities can be imported from and exported to other NgModules and can also be utilized by using multiple NgModules.

Every Angular app need to have at least one NgModule class i.e., the *root module*, which is named App Module and resides in a file named app.module.ts. The application is launched by *bootstrapping* the module. All the components and services defined in our application should be declared in this root module file.

Component:

A component is responsible for controlling the *views*. The component's application logic is defined inside a class. This class communicates with the view through an API of properties and methods. For our application, we created a component to define our graph and its characteristics. We defined methods to fetch response from our web-service and plot the results on a graph. We set our defined methods to be invoked on initialization. The code snippet is as below,

Graph Implementation:

```
export class GraphComponent implements OnInit {
subscription: Subscription;
public options: any = {
chart: {
      type: 'line',
      height: 500
        },
     title: {
          text: 'Bitcoin Price Preditcion Graph'
        },
        credits: {
         enabled: false
        },
        tooltip: {
         formatter: function () {
           return 'x: ' + this.x + '  y: ' + this.y;
         }
        },
        xAxis:
         {
         categories: []
         },
        series: [
         {
           name: 'Predicted Price',
           data: []
         },
         {
           name: 'Actual Price',
           data: []
         }
        ]
       }
```

Fetch Data from URL:

```
    getApiResponse(url) {
       return this.http.get(url, {})
```

```
        .toPromise().then(res => {
          return res;
        });
    }
```

#reads the result from our web-service and plots our graph.

```
    getData() {
    this.getApiResponse("http://localhost:62476/a
    pi/Main").then(
        (data:any) => {
          const yAxisPred = [];
          const yAxisActu = [];
          const xAxisArr = [];
          data.Actual.forEach(row => {
            const temp_row = [
              Number(row.Value)
            ];
            if(xAxisArr.find(ob => ob === row.Key)
    === undefined){
              xAxisArr.push(row.Key)
            }
            yAxisActu.push(temp_row);
          });
          data.Predicted.forEach(row => {
            const temp_row = [
              Number(row.Value)
            ];
    yAxisPred.push(temp_row);
    });
    this.options.xAxis['categories'] = xAxisArr;
    this.options.series[0]['data'] = yAxisPred;
    this.options.series[1]['data'] = yAxisActu;
    Highcharts.chart('container', this.options);
    },
    error => {
    console.log('Something went wrong.');
    })
    }
    ngOnInit() {
    this.getData();
    }
    }.toPromise();
 return res;
 }
 }
```

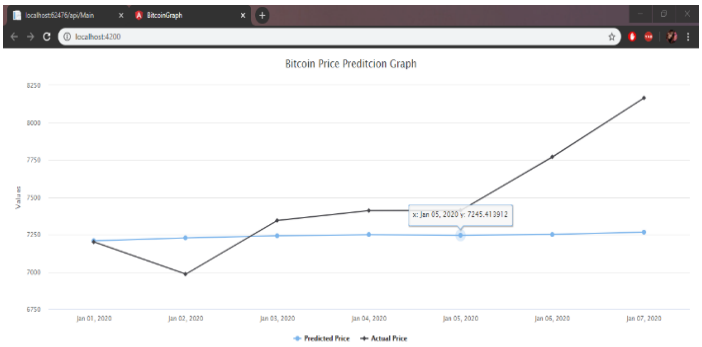
Services:

A service can be defined as a class with a precise and a well-defined purpose. Angular uses services to increase modularity and reusability as they generally have global scope. For our application, we used HttpClient service. Usually, front-end applications interact with backend services over the HTTP protocol. Latest browsers support two different APIs for making such HTTP requests: the XML Http request interface and the fetch() API.

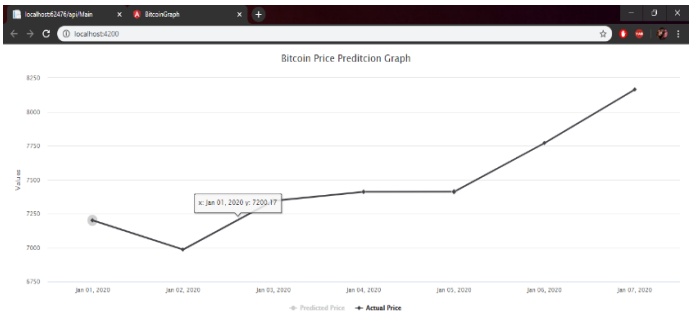
Once our component is ready, we ensure that all its necessary references and tags are created in our Angular application's root module. We access the app.module.ts file present at the root folder and add the HttpClientModule references there.

On successful execution it prompts open our browser and we can see the graphs for both predicted and actual price values of bitcoin.

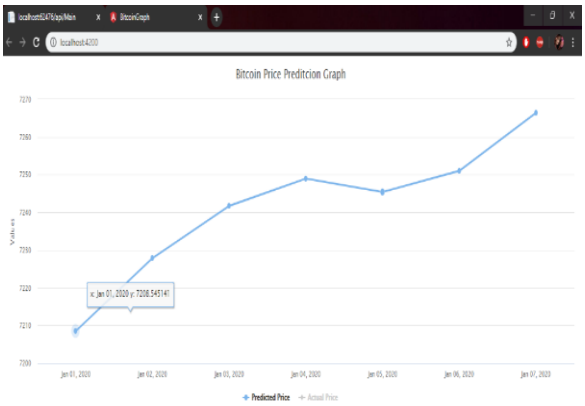
Output:



a) Predicted & Actual values of BTC prices



b) Actual values of BTC prices



c) Predicted values of BTC prices

VI. Conclusion

In this research work, we have investigated bitcoin closing price prediction by using an ARIMA model. Towards this end, at first, we have preprocessed time series data to make it stationary, and then, have searched over feasible (p, q, d) parameters for finding the ARIMA model which minimizes the MSE (Mean Squared Error) of prediction. The results we get indicates that the bitcoin price prediction using the value "closing price" history could result in large MSE values since bitcoin's price is vulnerable to high jumps and fall-downs. On the other hand, the results also confirm that the ARIMA model could be still used for price prediction in sub-periods of the timespan, which is by dividing the timespan to several timespans over which, dataset has a unique trend. Furthermore, we have investigated the effect of the different parameter p, q and d value on the achieved MSE in price prediction. We have elaborated this work by creating web service and high chart for better resolution of graph obtained. In future the research work could be further done by implementing ARIMA with CNN or LSTM mixed approaches to predict Bitcoin Price.

References:

[1] Wikipedia. (2018) The world's billionaires. [Online]. Available: https://en.wikipedia.org

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

[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 FinTech-KDD, 2018.

[7] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," Neurocomputing, vol. 10, no. 3, pp. 215–236, 1996.

[8] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns," in Neural Networks, 1988., IEEE International Conference on. IEEE, 1988, pp. 451–458.

[9] M. Bri`ere, K. Oosterlinck, and A. Szafarz, "Virtual currency, tangible return: Portfolio diversification with bitcoins," Tangible Return: Portfolio Diversification with Bitcoins (September 12, 2013), 2013.

[10] C. Chatfield and M. Yar, "Holt-winters forecasting: some practical issues," The Statistician, pp. 129–140, 1988.

[11] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," Andrej Karpathy blog, 2015.

[12] B. Scott, "Bitcoin academic paper database," suitpossum blog, 2016.

[13] https://coinmarketcap.com/