Price Prediction of Bitcoin with ARIMA and RNN

Praneetha Thotakura
Dept. of computer science
Georgia State University
Atlanta, USA
pthotakura1@student.gsu.edu

Rishinya Chowdary Ravipati
Dept. of computer science
Georgia State University
Atlanta, USA
rravipati1@student.gsu.edu

Dr. Yanqing Zhang
Professor, *dept. of computer science*
Georgia State University
Atlanta, USA
yzhang@cs.gsu.edu

***Abstract*—Bitcoin is encountered as the most popular digital money which works without any help of central authorities or any government monitoring. It depends on the concepts of peer-to-peer software and cryptography are most often used. The main theme in inventing the Bitcoin as it is used as the means of exchanging the money over the internet. The purpose of this digital currency was to build an alternative payment system which was free of higher authority control that can be used in the similar way of using the old currencies. In this paper, we propose an appropriate model which can forecast the price of the Bitcoin by applying ARIMA and RNN methodologies. The aim is to predict the Bitcoin prices using the above techniques for the next day and arrange an approach to increase the profits for the investors. Moreover, we also tend to identify whether there is an oscillation between the prices of Bitcoin and the related news.**

Keywords—Bitcoin, ARIMA, RNN, prediction analysis, cryptocurrency, daily news, stock market.

I. INTRODUCTION

In the year 2008, an author named Satoshi Nakamoto said, "The initial problem with conventional currencies is all the trust which is required to make it work" [1]. He worked on cryptocurrency where he developed few techniques which enables people to buy, sell or even trade them easily and trade them. As per records, the popular cryptocurrency is Bitcoin, which had an explosive price. The concepts of Bitcoin, Cryptocurrency are nowadays attracting the people's mind in attaining lot of profits for the investors.[2] The raise of Bitcoin in the last few years grabs all people attention and even the social media highlights it. The stock market is being maintained by the Bitcoin investors where all of them trying to predict the entire stock market by using this concept of Bitcoin.

In this paper we mainly discuss different experiments in forecasting the price and, we try to figure out how the external factors like news affect its price. However, we will go through the methodologies like Autoregressive Integrated Moving Average (ARIMA) and Recursive Neural Networks (RNN) and few improvement methods to evaluate the performance of all the prices with respect to the news obtained.

II. PREVIOUS RESEARCH

Many experiments were organized in determining the fluctuation of the Bitcoin price prediction using various methodologies among which one is said to be Time Series Analysis. Also even using twitter sentiment data and stock prediction analysis the price of bitcoin can be stated and. For example, if we talk about the previous model which is based on twitter sentiment analysis the prediction model mainly calculates the sentiment of tweet in the span of 30 minutes along 4 shifts and therefore resulted the accuracy of 80%. Also, by the social media platform the tweets were almost invented a weak calculation [1]. At last, one of the authors named Kim discovered the movement in the price by using few techniques like Neural Networks and Linear SVM. He had worked over two months in identifying the fluctuations of the bitcoin price and the news related to the raise in the price of bitcoin. Among these techniques the SVM had given the 65.03% accuracy despite of neural networks [2].

III. CONCEPT

In this proposed model, we implemented the model using few techniques which are related to the Deep Learning concepts. Among those concepts, Autoregressive integrated moving average (ARIMA) is generally used to depict the future data points on a time scale. The major benefit of this model is that this might be used in circumstances when the data is non- stationary. Apart from this, Recurrent neural networks (RNN) along with LSTM and GRU is also used. The major benefit of RNN is that they recognize the data's sequential characteristics and use patterns to predict the next likely scenario. Now, we go through the overview of these concepts

A. ARIMA

An autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

An ARIMA model can be understood by outlining each of its components as follows:

Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.

Integrated (I): represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).

Moving average (MA): incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA Parameters can be described as follows:

- p: the number of lag observations in the model; also known as the lag order.
- d: the number of times that the raw observations are differenced; also known as the degree of differencing.
- q: the size of the moving average window; also known as the order of the moving average.

In a linear regression model, for example, the number and type of terms are included. A 0 value, which can be used as a

parameter would mean that component should not be used in the model. This way, the ARIMA model can be constructed to perform the function of an ARMA model, or even simple AR, I, or MA models.

In an autoregressive integrated moving average model, the data are differenced in order to make it stationary. A model that shows stationarity is one that shows there is constancy to the data over time. Most economic and market data show trends, so the purpose of differencing is to remove any trends or seasonal structures. Seasonality, or when data show regular and predictable patterns that repeat over a calendar year, could negatively affect the regression model. If a trend appears and stationarity is not evident, many of the computations throughout the process cannot be made with great efficacy.

Configuring an ARIMA model: There are basic steps to configure an ARIMA model.

Identification: You should determine if the data series is stationary or if there is significant seasonality that should be included in the model. Seasonality can be identified through an autocorrelation plot, a seasonal subseries plot or a spectral plot. The plots and summary statistics will help the data scientist understand the amount of differencing and size of lag that will be required.

The Autocorrelation Function (ACF) is used to determine the number of MA(q) terms in the model. It determines the correlation between the observations at the current point in time and all previous points in time. The Partial Autocorrelation Function (PACF) results determine the order of the model or the values for the MA portion of the model. The model order reflects how many times differencing must be used to transform a time series into a stationary series. The ACF and PACF plots are used to check residual time errors in the series.

Estimation: Estimate the parameters of the model using statistical software programs that perform approaches such as nonlinear least squares and maximum likelihood estimation.
*Validation***:** Check the results of model against assumptions. If there is a high level of seasonality in the data, the ARIMA model may not be the best option. A more sophisticated seasonal ARIMA model may be required. Validation is an iterative process, and the best models are built through a series of trial-and-error runs.

The process should be repeated until the desired level of fit is achieved on the test datasets. Two common errors found in the validation or diagnostic stage are overfitting and residual errors. Overfitting is a sign the model is more complex than necessary and has incorporated random noise from the dataset. Residual errors in forecasting can highlight bias in the model that could lead to inaccurate forecasts.

B. RECURRENT NEURAL NETWORKS

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feed forward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to unsegmented tasks.

Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.

The term "recurrent neural network" is used to refer to the class of networks with an infinite impulse response, whereas "convolutional neural network" refers to the class of finite impulse response. Both classes of networks exhibit temporal dynamic behavior.

A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units. This is also called *Feedback Neural Network (FNN).*

a. LSTM (Long short-term memory):

Long short-term memory (LSTM) is an artificial recurrent neural network architecture used in the field of Deep Learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well suited to classifying, processing and making predictions based on time series data since there can be lags of unknown duration between important events in a time series.

LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

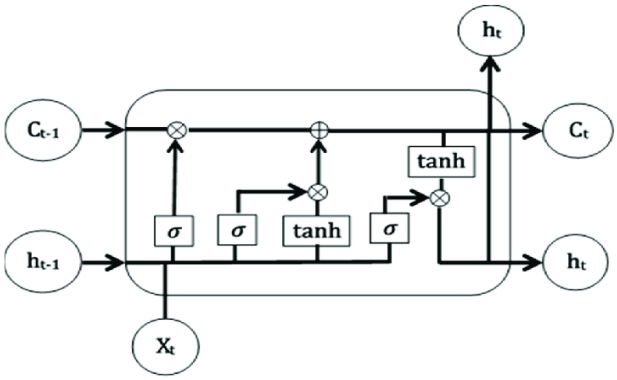

Fig.1 Block diagram of LSTM

i. Forget gate:

In a cell of the LSTM network, the first step is to step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for forget gate.

ii. Input gate:

Input gate is used to quantify the importance of the new information carried by the input.

iii. Output gate:

The output gate determines the value of the next hidden state. This state contains information on previous inputs.

b. GRU (Gated recurrent unit):

The GRU is like long short term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets.

Another Interesting thing about GRU is that, unlike LSTM, it does not have a separate cell state (Ct). It only has a hidden state (Ht). Due to the simpler architecture, GRUs are faster to train.

Architecture of Gated Recurrent Unit:

We have a GRU cell which more or less similar to an LSTM cell or RNN cell.

At each timestamp t, it takes an input Xt and the hidden state Ht-1 from the previous timestamp t-1. Later it outputs a new hidden state Ht which again passed to the next timestamp.

Now there are primarily two gates in a GRU as opposed to three gates in an LSTM cell. The first gate is the Reset gate and the other one is the update gate.

Reset Gate (Short term memory): The Reset Gate is responsible for the short-term memory of the network i.e the hidden state (Ht).

Similarly, we have an update gate. The only difference is of weight metrics i.e Uu and Wu.

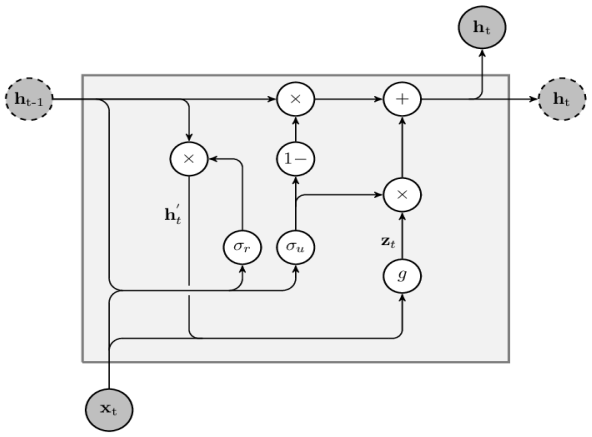

Fig.2 Block diagram of GRU

IV. PROPOSED MODEL

A. Dataset and Preprocessing:

The dataset is taken from coinbase website which contains minute to minute update of stock prices. After the dataset is loaded, the timestamp values are not humanly readable. So, they are converted into a normal date first. After this timestamp column is dropped and replaced by date column. There are few NAN values, and we don't want to drop those values. Since it is a time series data, we need to be extremely careful how we handle these values. There are two options: Fill the missing values with previous values (forward fill) Fill the missing values with future values (backward fill). We chose Forward fill. After this, the dataset is split into training and testing datasets based on the ARIMA and RNN models.

```
coinbase.shape
(1574250, 9)
```

Fig.3 Shape of the dataset

The dataset contains a total of 1574250 rows and 9 columns. Also, let us have a look of first five rows in a dataset to examine the columns.

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1417411980	300.0	300.0	300.0	300.0	0.01	3.0	300.0
1	1417412040	300.0	300.0	300.0	300.0	0.01	3.0	300.0
2	1417412100	300.0	300.0	300.0	300.0	0.01	3.0	300.0
3	1417412160	300.0	300.0	300.0	300.0	0.01	3.0	300.0
4	1417412220	300.0	300.0	300.0	300.0	0.01	3.0	300.0

Fig.4 Head of the dataset

As discussed, once the timestamp values are removed and replaced by date, the dataset looks as follow.

	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price	Date
0	300.0	300.0	300.0	300.0	0.01	3.0	300.0	2014-12-01 05:33:00
1	300.0	300.0	300.0	300.0	0.01	3.0	300.0	2014-12-01 05:34:00
2	300.0	300.0	300.0	300.0	0.01	3.0	300.0	2014-12-01 05:35:00
3	300.0	300.0	300.0	300.0	0.01	3.0	300.0	2014-12-01 05:36:00
4	300.0	300.0	300.0	300.0	0.01	3.0	300.0	2014-12-01 05:37:00

Fig.5 Timestamp replaced by Date

After this initial processing, the NaN values are replaced by values using Forward fill and all the values in the dataset are correlated.

The dataset looks as follows the correlation which is done by using coinbase.corr().

	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
Open	1.000000	0.999997	0.999997	0.999996	0.204421	0.497802	0.999999
High	0.999997	1.000000	0.999994	0.999998	0.204978	0.498775	0.999998
Low	0.999997	0.999994	1.000000	0.999997	0.203783	0.496770	0.999998
Close	0.999996	0.999998	0.999997	1.000000	0.204399	0.497814	0.999999
Volume_(BTC)	0.204421	0.204978	0.203783	0.204399	1.000000	0.575376	0.204366
Volume_(Currency)	0.497802	0.498775	0.496770	0.497814	0.575376	1.000000	0.497755
Weighted_Price	0.999999	0.999998	0.999998	0.999999	0.204366	0.497755	1.000000

Fig.6 Correlated Dataset

B. ARIMA Implementation:

The steps in the ARIMA Implementation are as follows:

Step 1: Converting Dataset into workable schema:

The dataset used in our research is indexed by timestamps (minute by minute for 7 years), this leads to very large number of transactions.

The dataset is converted into Day-to-Day indexed dataset, Monthly indexed dataset, quarterly indexed dataset, and yearly dataset by taking Statistical Average of the Time-stamped value for the desired period.

After analyzing this newly created dataset, we concluded that monthly-indexed dataset is perfect for our research as it doesn't lose any granular pattern and doesn't over emphasize any irrelevant variation in our dataset.

Step 2: Stationarize the Dataset:

Now, in order to produce good results from our Statistical ARIMA model, we need to stationarize our dataset. This data pre-processing work can be done by performing various transforms on our dataset.

Moreover, in order to validate whether our dataset is stationary or not, we have used Dickey–Fuller test.

Dickey-Fuller test: In statistics, the Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive time series model. The alternative hypothesis is different depending on which version of the test is used but is usually stationarity or trend-stationarity.

In order to stationarize the dataset, we have used a variety of transforms. We can use many transforms like Cox-Box, Seasonal differentiation, and Regular Differentiation for this purpose.

In our research, we applied them in the sequence mentioned above and we were able to successfully stationarize our dataset with a Dicky Fuller test value of 0.003063.

Step 3: Training ARIMA Model:

The statistical ARIMA model accepts 3 parameters which are defined below:

Lag Order (p): Number of lag observations included in the model.

Degree Of Differencing (d): Number of times that the raw observations are differenced.

Order Of Moving Average (q): Size of the moving average window.

In order to get the best model for our research, we trained our model for various values of (p, d, q) and then chosen the model according to AIC value.

The Akaike information criterion (AIC) is a mathematical method for evaluating how well a model fits the data it was generated from. In statistics, AIC is used to compare different possible models and determine which one is the best fit for the data. AIC is calculated from:

- the number of independent variables used to build the model.
- the maximum likelihood estimates of the model (how well the model reproduces the data).

The best-fit model according to AIC is the one that explains the greatest amount of variation using the fewest possible independent variables

Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. So, if two models explain the same amount of variation, the one with fewer parameters will have a lower AIC score and will be the better-fit model.

Thus, AIC rewards goodness of fit, but it also includes a penalty that is an increasing function of the number of estimated parameters.

Step 4: Prediction of Bitcoin:

The value predicted by our model will be in another value-range because we have done 3 transformations on it.

Now, in order to get the value in the required range we need to inverse those transforms and the value outputted by the inverse transforms is the predict value of the model.

C. Results of ARIMA Model:

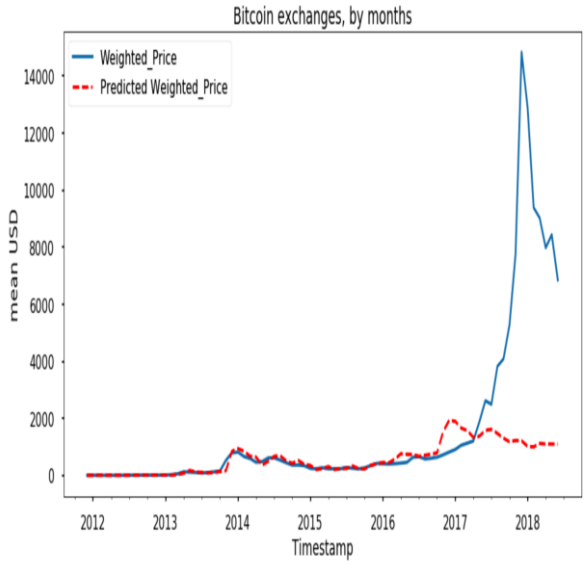

Fig.7 Prediction of Bitcoins using ARIMA

From the above image, we can conclude that our model predicted correctly to some extent. In order to bridge the little gap, RNN model can be used.

D. RNN Implementation:
The steps followed in the implementation of RNN are as follows:

Step 1: The dataset is loaded and fixed random seed for reproducibility.
Random seed: A random seed is used to ensure that results are reproducible. In other words, using this parameter makes sure that anyone who re-runs your code will get the exact same outputs.

Step 2: The dataset is then normalized using the Minmax Scalar.
Normalizing the Dataset: The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

Step 3: The dataset is the split into training and testing datasets.

Step 4: The input is then reshaped into samples, time steps and features.
One of the most crucial aspects of modern statistics and machine learning is resampling, which is the process of repeatedly drawing samples (subsets) from a training set and refitting a particular model on each sample in order to gain info such as variability of fit.

Step 5: Then the LSTM network is created and fitted into using keras.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

Step 6: The predictions are now made and inverted for best results.

Step 7: The root mean square is then calculated where 2161.60 RMSE is the train score and 2118.96 RMSE is the test score.
RMSE is a good measure of accuracy, but only to compare prediction errors of different models or model configurations for a particular variable and not between variables, as it is scale dependent.

Step 8: The results are then plotted.

E. RESULTS OF RNN:

When we observe the below graphs, we can see that the actual prices and predicted prices are almost the same which is not the case with ARIMA.

We observed a lot of difference in the actual and predicted prices by the last. This is not the case with RNN. This is because of the LSTM network present in the RNN model.

The LSTM model is considered superior to ARIMA at a cost of increased complexity.

E.Results of RNN Implementation:

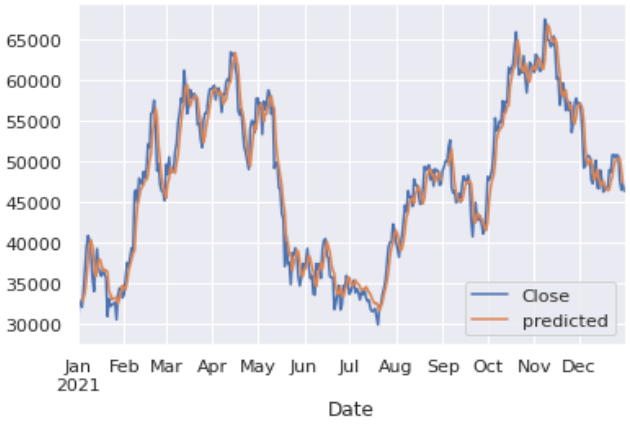

Fig.8 Prediction of Bitcoins using RNN for Jan 2021

Fig.9 Prediction of Bitcoins using RNN for Apr 2021

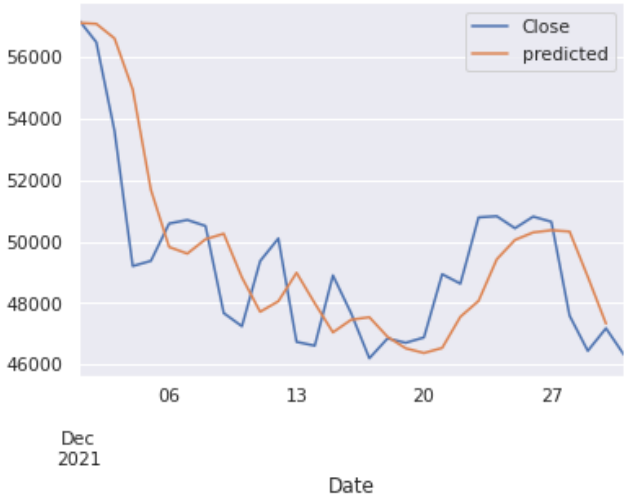

Fig.10 Prediction of Bitcoins using RNN for Dec 2021

V. FUTURE WORK

We can include other models to compare the results, such as SARIMA, Facebook Prophet, among others.We can also try GRU cells instead of LSTM for the RNN. We can include more features that influence BTC price. E.g.: some other stock exchange price, Twitter data, Whale data, altcoins data, active BTC addresses, etc. We can increase the number of datapoints by reducing the time intervals. In this research, we used daily prices, but we could try with hourly prices for example.

VI. CONCLUSION

Although both ARIMA and LSTM could perform well in predicting Bitcoin price, the LSTM would take extra amount of time to train the neural network model for about 42 minute an epoch via 4 core CPU or 1 minute 12 seconds via 2 core GPU. However, after training, the LSTM could make prediction more efficiently, and the precision rate is also higher. In general case, taking less previous data to make prediction in LSTM could lead to better result. ARIMA is quite efficient in making prediction in short span of time; but as the time grows, the precision rate would decrease.

REFERENCES

[1] Bitcoin Price Forecasting using Time Series Analysis, International Conference on Neural Networks in the year 2016.

[2] Connor Lamon & Eric Nielsen & Eric Redondo, Cryptocurrency Price Prediction Using News and Social Media Sentiment.

[3] Jonathan Rebane & Isak Karlsson & Stojan Denic & Panagiotis Papapetrou, Seq2Seq RNNs and ARIMA models for Cryptocurrency Prediction: A Comparative Study.

[4] Brandon Ly & Divendra Timaul & Aleksandr Lukanan, Applying Deep Learning to Better Predict Cryptocurrency Trends.

[5] Leopoldo Catania & Stefano Grassi & Francesco Ravazzolo, Predicting the Volatility of Cryptocurrency Time–Series.

[6] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)

[7] Medium. (2018). Introduction to Word Vectors – Jayesh Bapu Ahire

[8] Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents.

[9] Medium. (2018). A simple explanation of document embeddings generated using Doc2Vec. - Amar Budh