Prediction of cryptocurrency prices using LSTM and cryptocurrency financial data

Kaustubh Agnihotri (1001554290)
kaustubhsanjiv.agnihotri@mavs.uta.edu
University of Texas at Arlington

Abstract

Cryptocurrency trading has significantly increased in the past couple of years. It has created an expanding pool of money making people wanting to invest more and more, but with all the risk into account it is necessary to monitor the behavior of the crypto-currencies before investing. The approaches of data mining and machine learning can be incorporated into business intelligence (BI) systems to help users for decision support in many real-life applications. Considering this, in this paper we propose a machine learning approach to a system which can be used for the cryptocurrency price prediction. Specifically, we apply long short term memory (LSTM) blocks to build recurrent neural network (RNN) to handle the drastic changes in the price of any cryptocurrency as observed lately.

Keywords

Business Intelligence (BI), cryptocurrency, finance, price prediction, long short term memory (LSTM), recurrent neural network (RNN), multilayer perceptron (MLP), deep learning

1. Introduction

In the past couple of years, we have seen an incredible rise in the value of cryptocurrency. It is based on block chain technology which has gained a lot of attention during the recent years mainly because of the surge in the investments taking place in the cryptocurrency market all over the globe. This increase in the cryptocurrency trading has led to a significant rise in its prices and has become a hot topic of exploration around the world.

In this project we have used certain machine learning techniques to predict the prices of these crypto-currencies. Currently the system is able to accurately predict the closing price for the next few days for any cryptocurrency and has a future scope of predicting the prices and expected behaviors for a longer period with strategies to consider rare events. For this project we have used Long Short Term Memory (LSTM) model. LSTM is a deep learning model and is very effective while working with a time series data. LSTM units function as the building units for layers of Recurrent Neural Network (RNN). We also have normalized the data since it is spread over a wide range of prices for different cryptocurrencies. We have compared the predicted data and visualized it by plotting graphs.

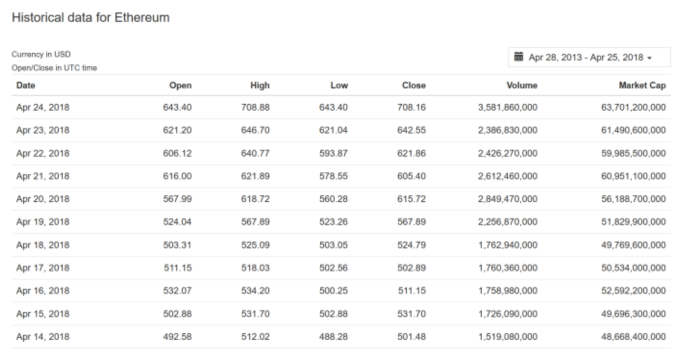
In order to have a good comparative study we have also implemented a Multilayer Perceptron (MLP) along with LSTM network. We have also generated random walks for the comparison purposes. Models and algorithms which are able to predict the cryptocurrency prices accurately can be of great incentive in the financial market due to the surge in the investments.

2. Dataset Description

It is very important to have recent data in order to predict the cryptocurrency prices as accurately as possible. Since the prices change every day, it is necessary that we update the dataset on a daily basis. Using a static dataset for the project was not a good option since it would be tedious to update such dataset.

In order to avoid this complexity, we have used the data provided by coinmarketcap.com. We are able to fetch the latest updated data from the web whenever we run the program. This has been implemented using the read_html functionality provided by pandas library.

Below is a snapshot (Figure 1) of the dataset along with description of each column (for Ethereum):



Historical data for Ethereum

Currency in USD
Open/Close in UTC time

Apr 28, 2013 - Apr 25, 2018

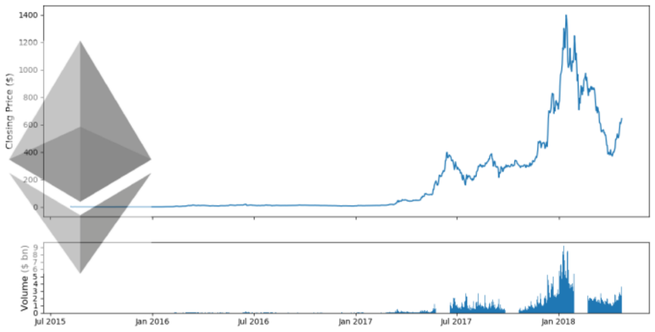
Date	Open	High	Low	Close	Volume	Market Cap
Apr 24, 2018	643.40	708.88	643.40	708.16	3,581,860,000	63,701,200,000
Apr 23, 2018	621.20	646.70	621.04	642.55	2,386,830,000	61,490,600,000
Apr 22, 2018	606.12	640.77	593.87	621.86	2,426,270,000	59,985,500,000
Apr 21, 2018	616.00	621.89	578.55	605.40	2,612,460,000	60,951,100,000
Apr 20, 2018	567.99	618.72	560.28	615.72	2,849,470,000	56,188,700,000
Apr 19, 2018	524.04	567.89	523.26	567.89	2,256,870,000	51,829,900,000
Apr 18, 2018	503.31	525.09	503.05	524.79	1,762,940,000	49,769,600,000
Apr 17, 2018	511.15	518.03	502.56	502.89	1,760,360,000	50,534,000,000
Apr 16, 2018	532.07	534.20	500.25	511.15	1,758,980,000	52,592,200,000
Apr 15, 2018	502.88	531.70	502.88	531.70	1,726,090,000	49,696,300,000
Apr 14, 2018	492.58	512.02	488.28	501.48	1,519,080,000	48,668,400,000

1 Cryptocurrency Dataset (Ethereum)

Column description:

a. Currency: Name of the cryptocurrency (Header)
b. Date: Date for which the record belongs
c. Open: Opening price for the day
d. High: Highest price reached during the day
e. Low: Lowest price reached during the day
f. Close: Closing price at the end of the day
g. Volume: Amount of trading done during the day
h. Market cap: Total value of all the shares at the end of the day

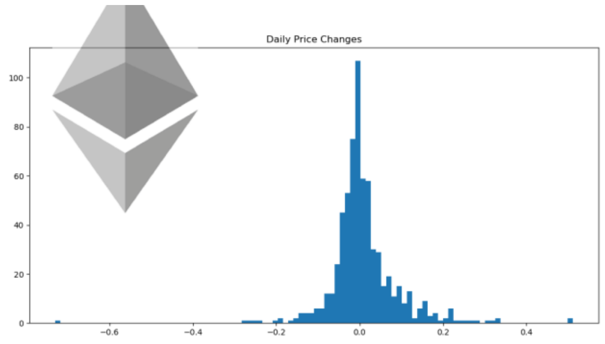
Figure 2 below shows us the trend of the closing price for Ethereum. We can clearly see that there was a sudden rise in the price over a short period.



2 Closing Price/Volume (Ethereum)

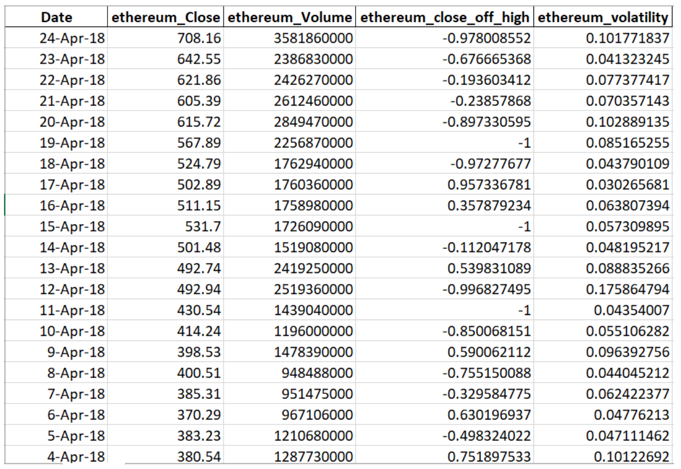
The data is normally distributed. We confirm this by plotting the distribution. This (Figure 3) plot clearly shows the bell curve of the distribution.

We have normalized the data since it is spread over a wide range of values for various columns. After normalizing we have extracted certain features which we use for training the model to predict the cryptocurrency prices.



3 Normal Distribution (Ethereum)

Figure 4 shows the dataset after feature extraction.



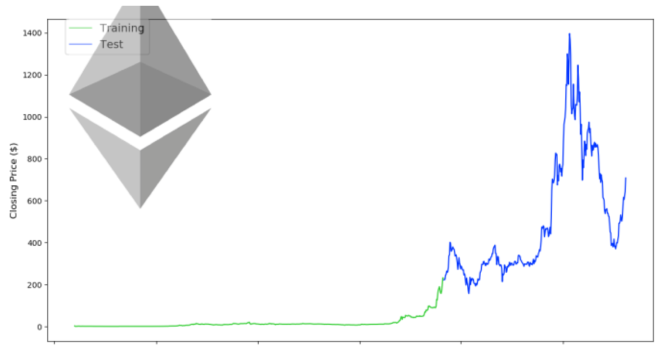
Date	ethereum_Close	ethereum_Volume	ethereum_close_off_high	ethereum_volatility
24-Apr-18	708.16	3581860000	-0.978008552	0.101771837
23-Apr-18	642.55	2386830000	-0.676665368	0.041323245
22-Apr-18	621.86	2426270000	-0.193603412	0.077377417
21-Apr-18	605.39	2612460000	-0.23857868	0.070357143
20-Apr-18	615.72	2849470000	-0.897330595	0.102889135
19-Apr-18	567.89	2256870000	-1	0.085165255
18-Apr-18	524.79	1762940000	-0.97277677	0.043790109
17-Apr-18	502.89	1760360000	0.957336781	0.030265681
16-Apr-18	511.15	1758980000	0.357879234	0.063807394
15-Apr-18	531.7	1726090000	-1	0.057309895
14-Apr-18	501.48	1519080000	-0.112047178	0.048195217
13-Apr-18	492.74	2419250000	0.539831089	0.088835266
12-Apr-18	492.94	2519360000	-0.996827495	0.175864794
11-Apr-18	430.54	1439040000	-1	0.04354007
10-Apr-18	414.24	1196000000	-0.850068151	0.055106282
9-Apr-18	398.53	1478390000	0.590062112	0.096392756
8-Apr-18	400.51	948488000	-0.755150088	0.044045212
7-Apr-18	385.31	951475000	-0.329584775	0.062422377
6-Apr-18	370.29	967106000	0.630196937	0.04776213
5-Apr-18	383.23	1210680000	-0.498324022	0.047111462
4-Apr-18	380.54	1287730000	0.751897533	0.10122692

4 Dataset After Feature Extraction (Ethereum)

Column description:

a. Date: Date for which the record belongs
b. ethereum_Close: Closing price at the end of the day
c. ethereum_Volume: Amount of trading done during the day
d. ethereum_close_off_high: Difference between the closing price and highest price of a particular day. Values -1 and 1 indicate that the closing price was equal to daily low and daily high respectively
e. ethereum_volatility: Volatility is calculated as the difference between the daily high and daily low divided by the opening price for a particular day

These are the basic features of any trading stock that can be used to monitor the movement in its price. We divided the data into training and testing datasets with a 2:1 ratio. Figure 5 shows the division of the data.



5 Dataset Division (Ethereum)

We have tried to divide the data up to a point just before the sudden rise in the price so as to see if our model is able to adapt to the sudden surge in the price.

3. Methods Used

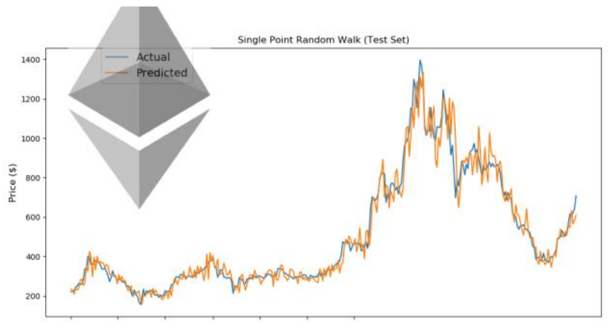
Random Walk

Random walk is a stochastic or random process which describes a path which consists of succession of random steps on some mathematical space such as the closing price in this case. Random walk most often refers to a special category of Markov chain or Markov processes, but many time dependent processes are referred to as random walks, with a modifier indicating their specific properties. [4]

Here we have used the random walk so as to notice how close can we get to the price changes by using just random number variation. For this we have extended the basic lag model wherein we set tomorrow's price equal to today's price and multiplied it with a random number. In mathematical terms we can state it as follows:

$$PredPrice_t = ActualPrice_{t-1} * \epsilon, \epsilon \sim N(\mu, \sigma)$$

Here we determine the μ and σ from the training dataset and apply the random walk model for prediction. Figure 6 shows the single point random walk that we get using the above method.

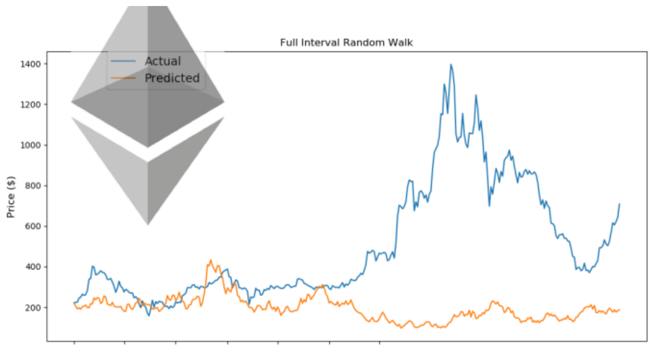


6 Single Point Random Walk (Ethereum)

Although it seems like the model is able to predict the prices quite accurately, however models that make predictions only one point into the future are often misleadingly accurate as errors aren't carried over to subsequent predictions. Now we tried to measure its accuracy over multipoint predictions so that the errors from previous predictions aren't reset but are rather compounded by further predictions. This can be represented as follows:

$$PredPrice_t = ActualPrice_{t-1} * \epsilon, \epsilon \sim N(\mu, \sigma) \; \& \; PredPrice_0 = Price_0$$

As seen from figure 7 the model could not accurately predict the prices anymore.



7 Multipoint Random Walk (Ethereum)

Even though single point random walk was able to predict the prices accurately, the model did not have any solid grounds. Hence, we now move on to the magic of artificial intelligence.

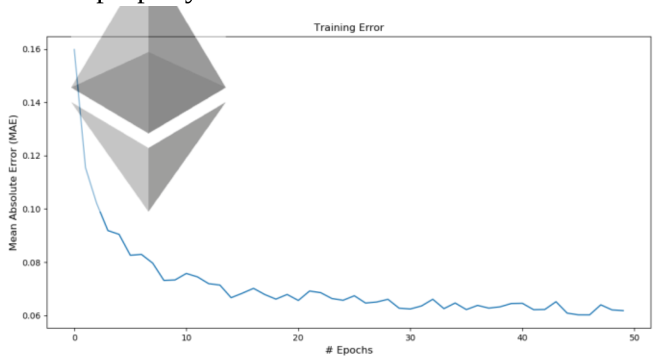
Long Short Term Memory (LSTM)

Humans don't start thinking from the scratch at every moment [3]. For ex. while reading this paper, each word we understand is based on the understanding of the previous word. Our thoughts have persistence. Traditional neural networks lack this ability and seems to be a shortcoming. Recurrent neural networks address this issue. In simple terms, they are networks with loops within them to allow the persistence of information.

LSTMs allow to produce very special kind of neural network which works for much more effectively for a wide variety of tasks. With RNNs we face the problem of long term dependencies. It is quite possible that the gap between the relevant information and the time when it is actually needed becomes very large. As the gap grows, RNNs become unable to learn to connect the information. Even though theoretically RNNs are absolutely capable of handling these gaps, they are unable to do so in practice.

LSTMs are a special kind of RNN, capable of handling and learning long term dependencies [3]. The default behavior of LSTM networks is to handle the long term dependencies. They have a similar chain like structure as RNN, but the repeating module has a different structure using four layers of neural network and interacting in a way to remember long term information.

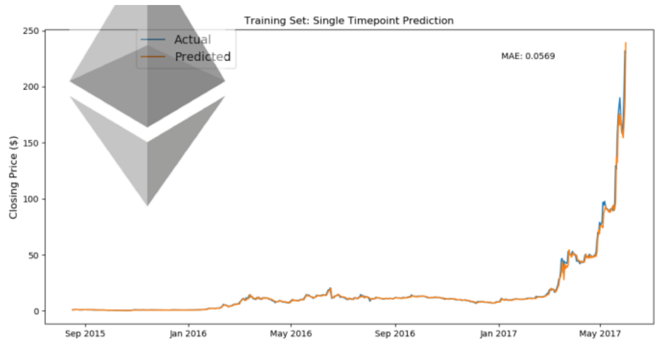
We use this ability of LSTM to make our model adapt to the sudden changes in the cryptocurrency prices. The model is able to normalize the predictions to the window size that we set. We have used the LSTM package from the keras module in python. While building the model we have used the Mean Absolute Error as the loss function. We also set the return sequences and stateful parameters to true so as to build the RNN even more effectively. As we train our model the error goes on reducing as seen in figure 8. This reduction in the error certifies that the model is being trained properly.



8 *Mean Absolute Error (Ethereum)*

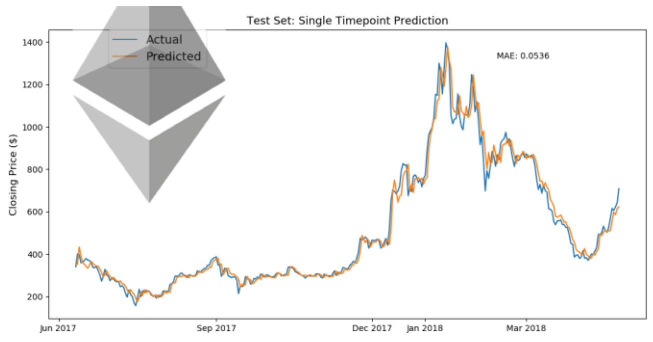
While training the model we have also used the auto validation feature of keras there by ensuring that we don't face the over fitting problem. After training the model we predicted the prices on the training as well as testing data. For the training dataset the predictions had a mean absolute error of 0.0569 for the Ethereum dataset. We can see from figure 9 that our predictions were quite accurate over a large time span.

For the predictions on the test set we got a mean absolute error of 0.0536 for Ethereum dataset. This clearly suggests that the trained model performed very well on the test set and predicted the prices with very high accuracy.



9 *Training Set Prediction (Ethereum)*

Figure 9 shows the test prediction graph. We were able to obtain an accuracy of 95.033 for the test set predictions for Ethereum dataset.



10 *Test Set Prediction (Ethereum)*

Multilayer Perceptron (MLP)

In order to compare the results obtained from the LSTM model with another reasonable model we have implemented a multilayer perceptron using the keras module in python.

Multilayer perceptron is a class of feed forward artificial network. It consists of three or more layers of nodes [5]. Each layer except for the input layer consists of nodes which are neurons using a non-linear activation function. The multiple layers and non linear function are the differentiating factors distinguishing between linear perceptron and multilayer perceptron. A multilayer perceptron is able to distinguish between data that is not linearly separable.

Multilayer perceptron refers to a number of perceptrons that are organized into layers. [6] We have used rectified linear unit (ReLU) as the activation function for building the model. The ReLU function is f(x) = max(0, x). This is applied element-wise to the output of some other function. In multilayer perceptron rectifier units replace all other activation functions except perhaps the readout layer. ReLUs improve the neural networks by speeding up the training. The gradient computation is very simple and there are no mathematical operations like exponentials, multiplication or division. We have used Mean Absolute Error as the loss function so that we can compare the performance with LSTM model.

4. Results

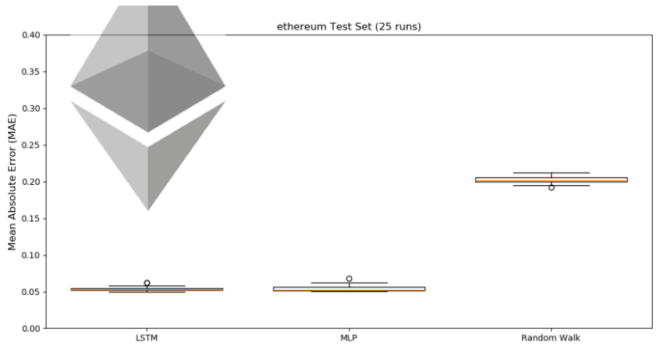
LSTM Model

The LSTM model which consisted of recurrent neural network constructed by using LSTM blocks gave us a very low mean absolute error and a very high accuracy. We had large data set for Ethereum, but we also tested the model with the Bitcoin-cash dataset which had very less number of records and found that the model performed satisfactorily. Below is table with the mean absolute error and accuracy for Ethereum and Bitcoin-cash.

1 *MAE & Accuracy (LSTM)*

Currency	Mean Absolute Error (MAE)	Accuracy (%)
Ethereum	0.0569	95.03
Bitcoin-cash	0.0691	93.99

Now since the model is dependent on random seed, we implemented another MLP model which also used random seed and did 25 test runs on the models using 25 different random numbers. We also did the same with the random walk model which had predicted the prices pretty accurately if not as much as the LSTM model. We were able to determine the consistency of the models by comparing the mean absolute errors produced by them. Figure 11 shows us the comparison between the mean absolute errors produced by these models.



11 *MAE Comparisons (Ethereum)*

Table below shows all the ranges of the mean absolute errors for Ethereum for LSTM, MLP and Random Walk models. We can clearly see that the LSTM model has a very low range of mean absolute error. Whereas the MLP model and the random walk model had significant range of error. This suggests that the two models lack consistency i.e. we will be getting results, significantly varying every time we train the model. As we know that the prices are updated daily and hence the model will also be needed to be trained often if not daily (though daily preferred). In this case a high consistency and high accuracy is always preferred.

2 *MAE Comparisons (Ethereum)*

Model	Avg MAE	MAE Range
LSTM	0.0569	0.0143
MLP	0.0576	0.0220
Random Walk	0.2156	0.0227

From the table we see that the average mean absolute error for 25 different LSTM and MLP models is very close. However, the range of mean absolute error for the 25 test runs is significantly high for MLP as compared to LSTM models. The random walk model has a range pretty close to MLP, but it definitely lacks accuracy as the average mean absolute error is too high. Thus, the LSTM model is able to consistently perform better than the other two models.

5. Conclusion

We successfully predicted the prices for future with the help of the three methods that we used. From the results that we obtained we were able to conclude that the LSTM model performed better than the simple MLP model and the Random Walk model while predicting prices for a 25 test run with different random seeds. We were able to achieve high accuracy even though there were huge drifts in the prices of the cryptocurrency.

The use of RNN with LSTM as the building blocks allowed us to reduce the error even when the data trend shifted drastically as time progressed. This was also the factor responsible in neutralizing the effect that random seed has on the predictions. The small window size of LSTM, normalizing every prediction and the ability to handle long term dependencies was responsible for making the model accurate and consistent.

Future Scope

The model as of now focuses on using the mean absolute error as loss function for the training purposes. We can improve the performance by including mean squared error as well which is expected to make the model adapt better to peaks and troughs thereby improving the accuracy of the model.

Using more data features along with sentiment analysis might make the model efficient enough to predict the sudden rise/fall int the price. However, it is not absolutely certain that sentiment analysis will be able to foresee such events, but we can use this as a factor to complement what our model predicts.

6. References

The reference papers do not have a direct relationship with the project. However, the base concept of predicting the stock prices using deep learning is at the core of all the papers. The methodologies used are quite different hence comparing them directly was not possible, but the accuracy and consistency shown by the RNN based LSTM model came out to be better than any other methods.

[1] Carson Kai-Sang Leung, Richard Kyle MacKinnon, Yang Wang, *A Machine Learning Approach for Stock Price Prediction*

[2] Dr. V. Z. Attar, Kaustubh Khare, Omkar Darekar, Prafull Gupta, *Short Term Stock Price Prediction Using Deep Learning, 2017 2nd IEEE International Conference*

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] https://en.wikipedia.org/wiki/Random_walk

[5] https://en.wikipedia.org/wiki/Multilayer_perceptron

[6] https://stackoverflow.com/questions/24282121/why-use-tanh-for-activation-function-of-mlp?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

7. Appendix

1. I have worked by myself on the project