Bitcoin Stock Price Prediction

Bharat Vaidhyanathan
College of Computer & Information Science
Northeastern University, NUID: 001280769
Boston, MA 02115
`vaidhyanathan.b@husky.neu.edu`

Manthan Thakker
College of Computer & Information Science
Northeastern University, NUID: 001229259
Boston, MA 02115
`thakker.m@husky.neu.edu`

Abstract

Cryptocurrency, especially Bitcoin, is one of the most volatile markets today and has gained a lot of attention from investors across the globe. Cryptocurrency, being a novel technique for transaction system, has led to a lot of confusion among the investors and any rumors or news on social media has been claimed to significantly affect the prices of cryptocurrencies. The goal of this study is to predict prices for Bitcoin using Machine Learning Techniques for the next day and prepare a strategy to maximize gains for investors. We also aim to find out if there is a co-relation between fluctuating Bitcoin prices and related news.

1 Introduction

Stock market is one of the most volatile data available in terms of Machine learning datasets. Researchers have been long trying to predict the stock market and any breakthrough in this field would result in, literally, the people being able to mint money. Cryptocurrencies, to be specific, has gained a lot of traction in the recent years from investors accross the globe. Cryptocurrency being a novel technique for transaction system has led to alot of confusion among the investors and any rumors or news on social media has been claimed to significantly affect the prices of cryptocurrencies. Bitcoin is one of the oldest and biggest cryptocurreny being traded as of now, in terms of the volume being traded. It is so big that even now, with the advent of thousands of new cryptocurrencies, Bitcoin has a market share of more than 55% as compared to other cryptocurrencies, being followed by Ethereum at 8.57%. This says a lot about why Bitcoin might be a really interesting and important stock to predict. Also, Bitcoin prices fluctuate heavily. Over the past 2 years, Bitcoin has seen its highest price around \$20000 and its lowest price around \$900. It is very sporadic and this is one of the most important reasons which attracted us to analyse and predict its price. Our rest of the paper would discuss our various attempts to predict its price and us trying to reason out how external factors like news affect its price.

Contributions Throughout the paper we have enumerated and explained the challenges of crypto-currency price predictions. This project has been a woderful learning experience for both of us.

Manthan

- Bitcoin price dataset and its cleaning
- Bitcoin price dataset transformation and imputing
- Feature correlation, importance and more analysis
- Implementation of the ARIMA model
- Implementation of the Recurrent Neural Network with LSTM

Bharat

- Related News Collection from Coindesk
- Further cleansing of the news and vectorizing using NLP techniques
- Implementation of the Random Forest Model
- Parameter tuning for Random Forest
- Implementation and tuning of lookback (How many days before data to look)
- Analysis of the effect of related news on the predictions

2 Related Work

ARIMA Time Series data mining can be identified as Forecasting or a Classification task. In Classification, we predict classify if the prices are going to increase by a certain percentage or decrease by a certain percentage relative to previous values. In Forecasting, we predict the future values given the past data. Time-series forecasting has been performed predominantly using statistical-based methods, for example, the linear autoregressive (AR) models because of their flexibility to model many stationary processes. These include the well-known ARMA (autoregressive moving average) model (Fan and Yao, 2003) and its extensions by Weron and Misiorek (2008) for short-term time series forecasting. The ARMA model assumes a linear relationship between the lagged variables and produces only a coarse approximation to real-world complex systems and generally fails to accurately predict the evolution of nonlinear and non-stationary processes. ARMA model performance frequently degrades considerably whenever time trends and seasonality features are present in the highly fluctuating time series data. Methods such as the linear autoregressive integrated moving average (ARIMA) (Fan and Yao, 2003), which are based on the evolution of the increments are used at times to remove/reduce first-order (moment) non-stationarity. However, differencing generally amplifies the highfrequency noise in the time series, and great effort is thus required to determine the order of an ARIMA model. Also, ARIMA models are largely limited to capturing the first-order non-stationarity in a time series data. Engle (1982) introduced the ARCH (autoregressive conditional heterodasticity) model to capture the second-order (moment) non-stationarity; i.e., time-varying conditional variance or volatility. However, most of these methods tend to be limited for nonlinear and stationary time series forecasting by the local linearity assumption implicit with an AR-type structure.

Stanford Over the last decade, Artificial Neural Networks have outperformed various statistical and probabilistic models especially for classifications tasks. One such attempt by students at Stanford tried to predict the price direction for the next 3 days based on past data. [x] performs comparative study of Logisitic Regression, Naive Bayes and Deep Neural Networks and finds that Artificial Neural networks are able to classify the directions with an accuracy of 72 percent. They also confirm that spikes in bitcoin prices are also co-related to sentiments of people in social media forum about bitcoin.

Kaggle In the past year, Recurrent Neural Networks have gained allot of popularity to predict sequence to sequence time series data to forecast stock prices. Adding to that, the literature is rich with all kinds of RNN as its widely used in Natual Language Processing. Various Kaggle Competitions and online forums report better accuracy for stock market predictions using RNN, but there have been rare or no such formal investigation on predicting Spikes and as well as the time series data of bitcoins which will serve as our goal of this project.

3 Method and Model

To solve the problem using Machine learning, we first tried to categorize the problem and tried to find previous solutions on how they solved it. We quickly learned that, since, the problem involves prices which are changing with time, this could be modelled as a Series prediction problem. In parallel, we also tried to solve the problem as a normal machine learning problem with the features being the previous prices and the output being the price predicted for that day. We explain what models we hvae used and how we configure them to predict the Bitcoin prices.

3.1 ARIMA

Autoregressive integrated moving average (ARIMA) is a statistical regression model, which can be utilized in time series forecasting applications, such as finance. ARIMA makes predictions while considering the lagged values of a time series, while accommodating for non-stationarity. The model, which is one of the most popular linear models for time series forecasting, originates from the autoregressive (AR) and moving average (MA) models, as well as their combination, also known as ARMA. For making predictions with the ARIMA model, we had to follow a step by step procedure to be able to feed the data to the ARIMA model. This first involved Visualizing the time series data: It is essential to analyze the trends prior to building any kind of time series model. The details we are interested in pertains to any kind of trend, seasonality or random behaviour in the series.

To understand ARIMA, lets start with the math involved in it.
Let $y_1, ..., y_t$ define a univariate time series, with each $y_i \epsilon R$.
An AR model of order p, denoted as $AR(p)$, the current (predicted) value of a time series is expressed as follows:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon t \tag{1}$$

where c is a constant, $\phi_1, ..., \phi_p$ are the parameters of the model and ϵ_t is a variable producing white noise. Similarly, a moving average process of order q, denoted as MA(q), expresses the time series as a linear combination of its current and q previous values. More concretely, MA(q) is defined as follows:

$$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} \tag{2}$$

where is the mean of y, 1, . . . , q are the model parameters and i corresponds to a random variable producing white noise at time point i.
Using the AR and MA processes parameterized by their corresponding orders, p and q, respectively, an ARMA(p,q) process can be defined. Although the ARMA model can generally only be applied to stationary time series, non-stationary time series can be converted to stationary using the dth differentiation process. More generally, the process of differencing is the transformation of a time series to a stationary process by eliminating trends and seasonality, hence stabilizing the mean. This is achieved by simply by computing the difference between consecutive observations. Consequently, an ARIMA model with parameters p,d,q, denoted as ARIMA(p,d,q), performs d differentiations to the original time series values, i.e., yt is converted to y.

The full ARIMA(p,d,q) is defined as follows:

$$y_t^{(d)} = \phi_1 y_{t-1}^{(d)} + \phi_2 y_{t-2}^{(d)} + ... + \phi_p y_{t-p}^{(d)} + \epsilon_t + c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \mu \tag{3}$$

3.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:
For b = $1, ..., B$:
Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
Train a classification or regression tree fb on Xb, Yb. After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x' or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

3.3 Recurrent Neural Net

RNNs provide a generalization of the feed forward network model for dealing with sequential data, with the addition of an ongoing internal state ht serving as memory buffer for processing sequences. In this paper, we employ a seq2seq RNN. More concretely, let $\{x1, ..., xn\}$ define the set of n sequence inputs of a seq2seq RNNs and $\{y1, ..., yn\}$ be the set of outputs. For this study the internal state of the network is processed by Gated Recurrent Units (GRU)

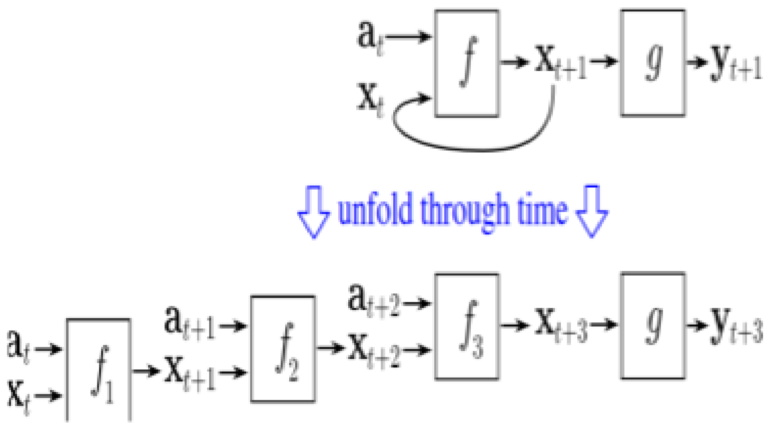

Figure 1: An internal representation of RNN

4 Experiments

4.1 Data

We use coinbase dataset from the publicly availble coinbase site which is by far the most reliable crypto exchange in United States. Our horizon forecast is days and therfore we aggregate the hourly data by day.

NAN Values: Common technique in time series data to handle NAN values is to forward fill the data in case of missing values. We need to be sure to not use the backward fill startegy in this case as we don't want to predict the past given the future.

Before we can jump into modeling, it's best to get an idea of the structure and ranges by making a few exploratory plots. This will also allows us to look for outliers or missing values that need to be corrected. Some of the interesting trends:

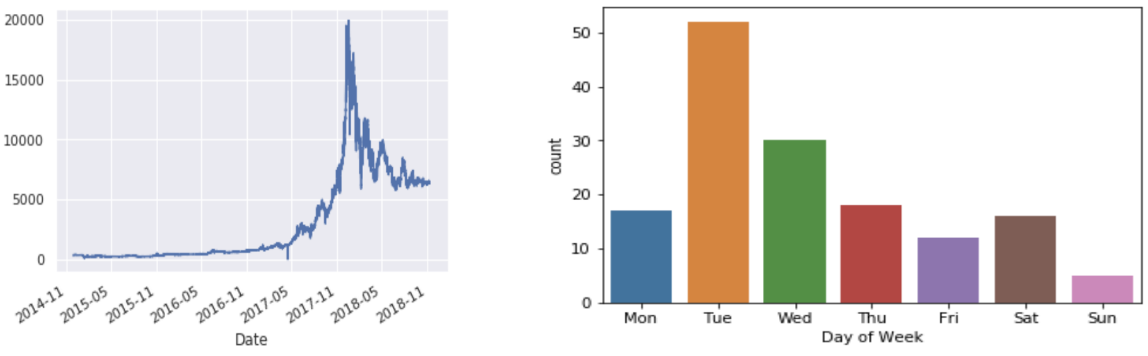

Figure 2: Left: Bitcoin price dataset visualized, Right : Bitcoin Volume traded per week of the day

Price Bitcoin prices are very volatile. We aggregate the data from the coin

Observation 1 As you can see from Figure 2 right, the graph the highest count of bitcoin transactions is on Tuesdays and lowest on Sunday which kind of expected but this confirms that there is some seasonality in data.

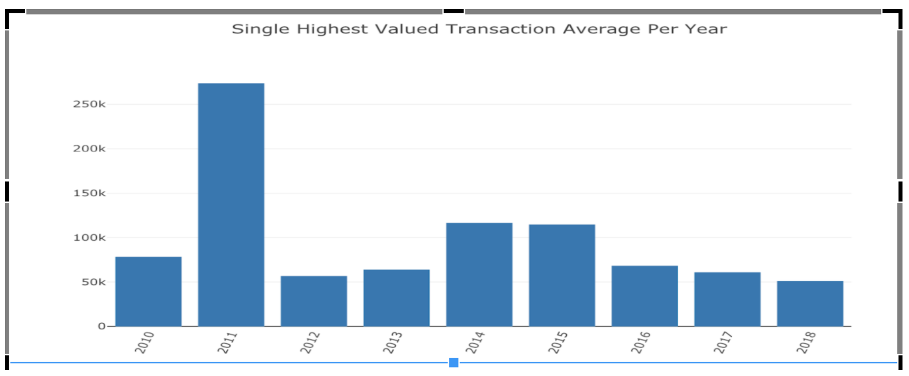

Figure 3: Dataset diversity: Bitcoin Volume traded per yer

As you can see from the above graph the highest valued transaction per year was actually during 2011 and not 2017 where the bitcoin gold rush where the prices shooted up from 7000 US dollar to 19000 to US dollars.

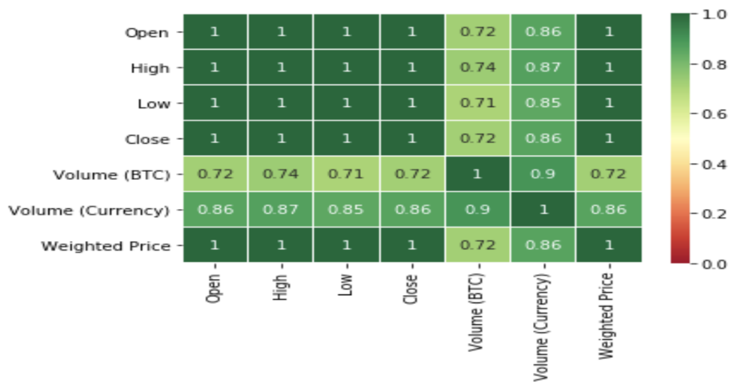

Figure 4: Feature correlation plot

The above graph represents the co-relation matrix between each features. As expected most of the features are highly dependent since they have been calculated for each day and are aggregated from same prices. The intresting fact is volume of bitcoin has a high corelation with volume of all cryptocurrencies which is expected as bitcoin is the most popular currency but this may vary with time.

Stationary: To Dealing with Seasonality and trend, we take the first order difference so that the price data is stationary and function a function of time

PartialAutoCorrelation: The important task would be to find the number of days on which today's price is dependent on, or formally finding the window size of past predictions to consider to predict present prices. We find past 7 days to be a good estimate to predict the present values.

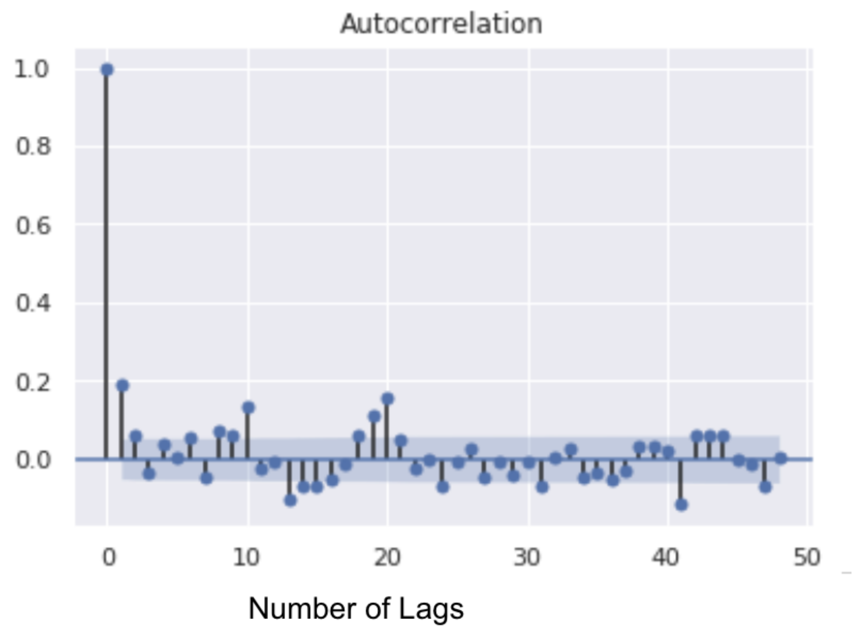

Figure 5: Feature exploration: Autocorrelation plot

4.2 News Data

Now, to investigate the claim of bitcoin related news and tweets affecting the bitcoin prices, we needed to perform some natural language processing on the input data. We started off with the input texts and converted them to word embeddings. Word embeddings are basically vectorial representation for words. We used the Doc2Vec utility (based off of Word2Vec) from the gensim library for generating these word vectors.

We then feed these sentence vectors to the models. To get the previous x day's news, we use the same kind of shifting as with the bitcoin price data to get the previous x day's news. A lot of days in our dataset had no news and this too was an important negative signal for the model, which it picked up on.

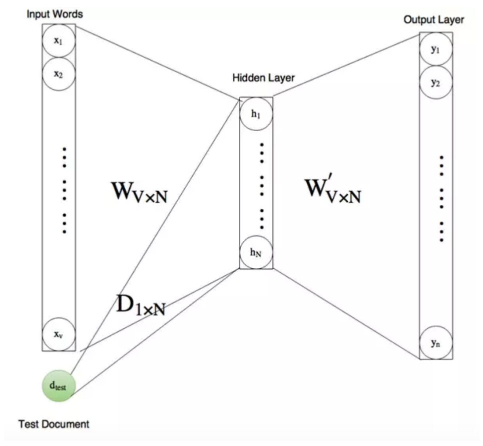

Figure 6: A representation of the internal Doc2Vec neural net

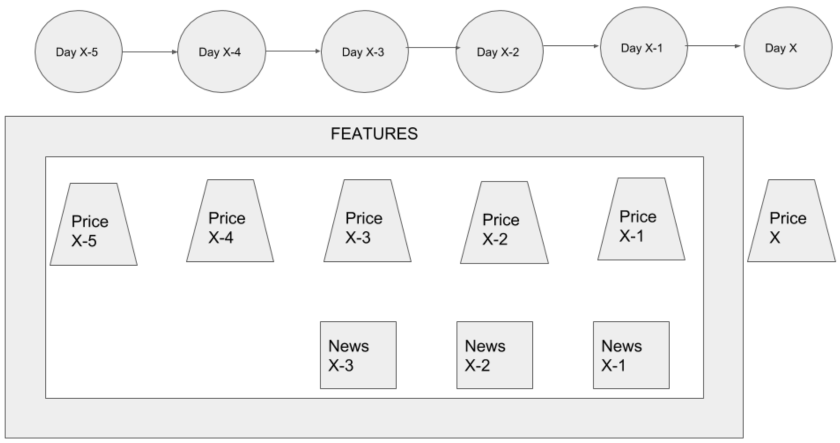

Figure 7: Feature design for the models

4.3 Experiments and Evaluations

ARIMA ARIMA performed similar to other models when the window length was 3 and forecasted a single day's value (present). This was expected given how ARIMA works which is intuitively autoregression for the past 3 values. However, the model failed significantly when it had to predict forecasts for the next 7 days.

Random Forest Random Forest performed performed in a pretty standard way when the window length was 3 and forecasted a single day's value (present). The RMSE with Random forest was decent and we saw that increasing the max depth up to 16 gave us the best results. Increasing the max depth upto 50 gave us the best results when considering the news data. We saw that Random forests started to over fit when we decreased the number of trees, and this was expected.

RNN The best architecture of RNN we found to be predict rmse to be decent in all cases was a deep RNN with 7 LSTM. We used mean squared error as loss function and for optimizer, we used adam optimizer and learning rate of 0.01 as opposed to 0.007 of default value. We also tried training using GRU and got very similary rmse results. However, with GRU, the training time reduced with little accuracy loss. We found training with 1000 epochs was sufficient enough. Adding to that, we found the predictions to be most accurate for immediate next day and relatively lesser with each day but better than ARIMA model.

Some Int resting observations: We found that GRU were able to adapt to newer sequences faster whereas LSTM were still weighing previous patterns.

Model	RMSE
ARIMA	\$395.24
RF	\$406.23
RF with News	\$849.61
RNN	\$271.62
RNN with News	\$539.85

Figure 8: RMSE comparisons for different models

References

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

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

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

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

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

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

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

[8] Medium. (2018). *A simple explanation of document embeddings generated using Doc2Vec.* - Amar Budhiraja