Final Project Report
ORIE 4741

# Bitcoin Projection


Jay Tappen (fjt37) and QiWen Shu (qs76)
December 4th, 2017

Image: https://fortunedotcom.files.wordpress.com/2016/09/bitcoin.gif

# Introduction

Bitcoin is an international cryptocurrency and digital payment system. It was initialized in 2009 and its value has grown exponentially over the years, with a price of over 10,000 USD per Bitcoin at the time of this writing. It has been the most successful cryptocurrency to date in terms of market capitalization and public recognition.

It was designed around a secure protocol originally put forth under a pseudonym in this paper. The protocol revolves around the "blockchain", which is a data structure that can be thought of as similar to a linked list. Each node in the list is a set of transactions, called a "block". As Bitcoin transactions occur, they are placed in a new, tentative block. The process known as "mining" is taking this tentative block and adding it officially and permanently to the blockchain. This process has been made intentionally computationally difficult, such that mining a block takes a non-negligible amount of work and time (and therefore money). This computational difficulty contributes to making the blockchain tamper-proof. The exact difficulty can be changed on the fly, such that as more people start mining Bitcoin, it becomes harder in order to maintain mining speed control. This means that the mining difficulty can be used as an indication of how many people are mining. This blog post is a good introduction to how Bitcoin works if you are interested in more technical details, or you can visit the official site.

# Project Goal

Build a model that predicts the price of Bitcoin.

# Motivation

There are a few uses for an accurate model to predict the price of Bitcoin. The first is speculative investment. The price of Bitcoin was $3,000 when we started this project a few months ago. Today, it is over $10,000. Being able to predict this magnitude of growth could result in enormous capital gains. Additionally, people are now worried that Bitcoin is experiencing a bubble, and that the price will drop rapidly in the near future. Being able to predict that drop could save you from significant capital losses.

A second use for a Bitcoin model is for companies that accept Bitcoin as payment for online purchases. Knowing the current price is important for setting prices, but knowing the price in the near future is important as well so that you can update your website preemptively to avoid losses due to fluctuations in Bitcoin value. This is especially true given how volatile Bitcoin currently is and how much people are worried about a sudden drop.

# Dataset

## Real-valued Historical Data:

The dataset we started with is time series data with a total of 1581 samples with 24 features. Each sample represents one day.

There are two primary types of features in the dataset. One consists of historical data on price, market capitalization, and trading volume. Such data possesses a more direct relationship to the bitcoin price. We are going to use the historical price data and volume as the primary predictors of future price.

The other type of feature in the dataset is features that may reveal some of the more indirect relationships. These features include bitcoin count, mining difficulty, hashrate, transaction count, and the number of unique IP addresses. Mining difficulty and hashrate indicate how much mining activity there is. Transactions and number of unique IP addresses indicate how often Bitcoin is being used and by approximately how many people.

## Discarding Some Misleading Data:

For the first 3 or so years, Bitcoin was not being publicly traded like it is today. This means that the market price is only an approximation and that the price was nearly flat for the entire duration. We decided to drop the first 1000 days of data for this reason. While this may seem drastic, it leaves more than half of the data remaining and we are better served by training and test datasets that better reflect the environment in which we are trying to predict.

## Political Events:

We received feedback that media and legislative events can have a big impact on the price and that we were not taking this into account. We recognized this issue and went in search of a way to include this data. We found a list of important political dates in Bitcoin's history[1], and categorized each as either positive, negative, or neutral. Then we added a feature to our dataset that took the value 1 on days with a positive event, -1 on days with a negative event, and 0 on all other days. The dataset of events that we used had 93 events in total, which made it a very sparse feature.

[1] https://timelines.issarice.com/wiki/Timeline_of_Bitcoin

# Method Overview

The approach we took involved three parts.

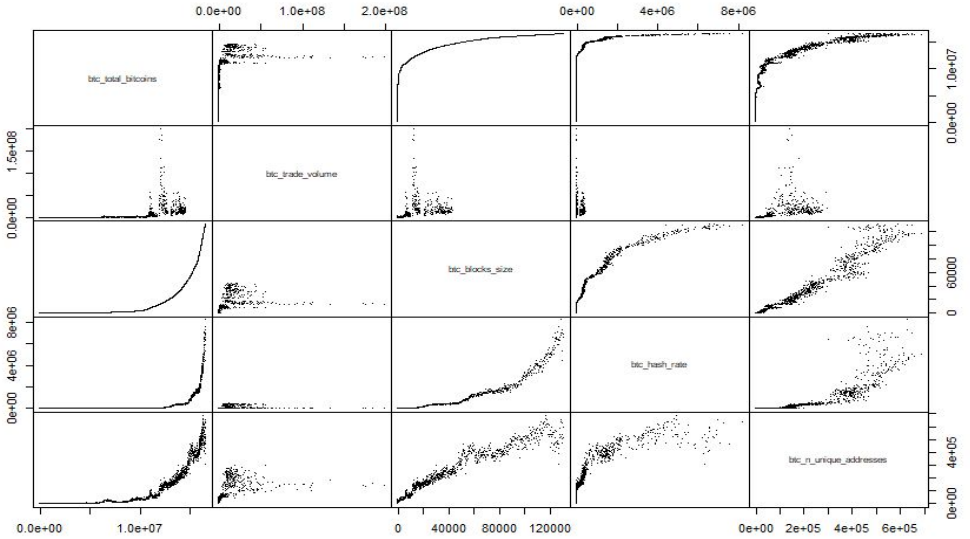
First, we examined the data. We performed exploratory data analysis, graphing various features with respect to the price, as well as with respect to other features. We got a sense for the shape of our data. We then went about the business of identifying which columns to include in our models and which columns might be redundant or unhelpful.

Once we had arrived at the set of features we thought would be most effective, we conducted some preliminary model analysis. This served the purpose of further narrowing down the feature set and providing us with some other guidelines for meta-parameters.

After these two steps, we were ready to construct our model to predict the price of bitcoin. This included training dozens of models to test different loss functions, regularizers, and data subsets.

In the end, this allowed us to arrive at the best possible model and draw conclusions.

# Feature Analysis




From the plot we can see that the number of total bitcoins, number of unique addresses, hash_rate, and block size are all closely related. The interrelationship between them from the above shown graph is observed to be positively related and potentially logarithmic, while future analysis would be needed to assess their exact relationship. Therefore, when we select the factors

for fitting our model, we will probably not select all of these four factors, instead, we will just pick one or two among them.

We also examined the relationship between the number of transactions and the other factors. Block Size and number of unique addresses seem to behave similarly, thus we should only pick one of them. The others all behave quite differently. Only the total number of bitcoins demonstrates a possible positive correlation with trading volume and hash rate is the most left skewed.

In the end, we chose the number of bitcoins, the number of transactions, the hash rate, and the number of unique addresses as our features. This came from intuition as well as correlation analysis. The number of bitcoins provides a measure of the age of the currency, and how many people have been involved over time; the number of transactions on a given day indicates how much the currency is actively being used; the hash rate indicates the amount of mining activity, which indicates the level of involvement of the technical community; the number of unique addresses indicates how many people are using the currency overall. We decided not to explain here why we discarded each of the other features, because there are so many of them, but in general it came down to each of the other features not adding any information that was not provided directly by these.

# Preliminary Model Analysis

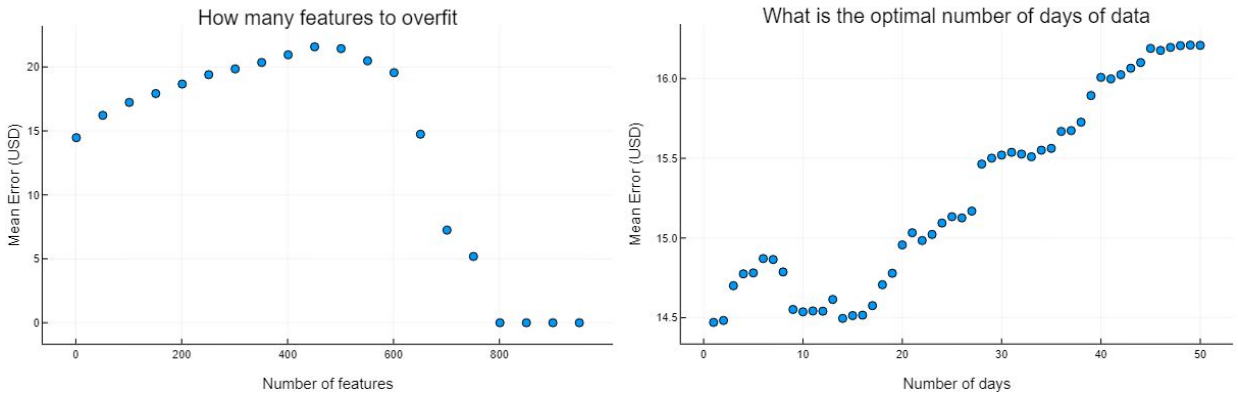
We created a variety of linear regression models and ran three primary tests. From these tests, we drew some conclusions that we used as we continued to develop our project. As an error metric for our tests we used the magnitude of the difference between the predicted price and the true price, averaged over all days. This error metric best represents the error measure that would most likely be used in applications of our model

The first test we ran was a test to determine which features are the most important. After narrowing down our feature set, as explained above, we created a linear regression model that used yesterday's value for each feature to predict today's price. We could then determine the features that had the greatest impact by looking at the magnitudes of the weights in the weight vector. As expected, the price was by far the most significant indicator, but we also established that the hash rate and the number of unique addresses had relatively significant weights as well. This makes sense because these features reflect the number of miners and users, respectively.

The second test we ran was to determine how many features we could safely use without the model simply memorizing the data. For the purposes of the test, in order to use $n$ features we would just use historical price data for $n$ days. A plot of our results is on the left below. It is essentially a graph of training error. At 800 features, the model has completely overfit and there

is no training error, and thus no generalization. In our final models, we end up using about 70 features, so we should be safe from overfitting.

The third test we ran was to determine the optimal number of days of historical data to use. We tested this with a few different subsets of the features and got similar results each time. A plot of our results when we used price and trade volume as our features is on the right.




The plot indicates that the minimum error is achieved when we use around 14 days of historical data. This matches our plots for other subsets of the input features, which all indicated around 2 weeks of historical data to be optimal (the minimum at 2 days in this graph is not consistent when we use different feature sets). From observing plots of predictions made by the models, the reason the error increases when more days are used is that the model is overfitting to sections of the data that have steep price curves in order to minimize squared error in training, but in exchange they are fluctuating heavily on flat sections of the data. This translates to a higher average absolute error in price prediction by our error metric.

# Main Analysis

We organized our data into meaningful input and output data for our models, and then trained numerous models to compare the results achieved by different strategies. We set up a script that allowed us to easily train and compare models using different loss functions and regularizers.

## Data Format:

We arranged our selected features' data into an input matrix $X$ and output vector $y$. We knew that we wanted each output to be the price for a given day, such that $y_n$ is the price on day $n$. We also knew that each day's price would have to be predicted using only data available in advance, such that in row $n$ of $X$ we could only have data from days 1 through $n$ - 1. Specifically, each row of $X$ is composed of the last 2 weeks of each feature. So, in row $n$, we have 14 columns for each

feature: the value of each feature for days $n - 14$ through $n - 1$. This gives 70 columns in $X$ (the four features from the dataset, plus historical market price as a fifth feature, multiplied by 14 columns per feature). It also means that $y$ starts with the price for day 15 so that we have all of the data required to populate the first row of $X$.

## Training Method:

We selected the proximal gradient method as our primary technique for training our prediction models for bitcoin price. The stochastic gradient method was not necessary because our dataset was small enough to run normal proximal gradient in a reasonable amount of time.

## Loss Functions:

Since we had a script to easily test different loss functions, we decided to try most of the regression loss functions that we learned in class. We tested huber loss as well as both $\ell_1$ and $\ell_2$ loss. Bitcoin price is known to be volatile at times, so we thought that a robust loss function might be beneficial to help deal with outliers. On the other hand, we knew that on some days when the price changes dramatically it might be very important to predict that despite it being an outlier statistically. For this reason, we included $\ell_2$ loss in our tests to better fit to those extreme data points.

## Regularizers:

We tested the three primary regularizers that we learned about in class that were relevant for our data. The first was an $\ell_1$ regularizer (lasso), which encourages sparsity. Sparsity was likely to make sense because we have 70 input features, and many of them may be somewhat related. We also tried $\ell_2$ regularization (ridge regression) which encourages small coefficients. This is often done to make sure that none of the coefficients get too large and to help establish a unique solution in an underdetermined problem. This ended up not being an issue. The last regularizer we tried was the non-negative regularizer. This made sense because each piece of historical data had an intuitively positive relationship with the price output.

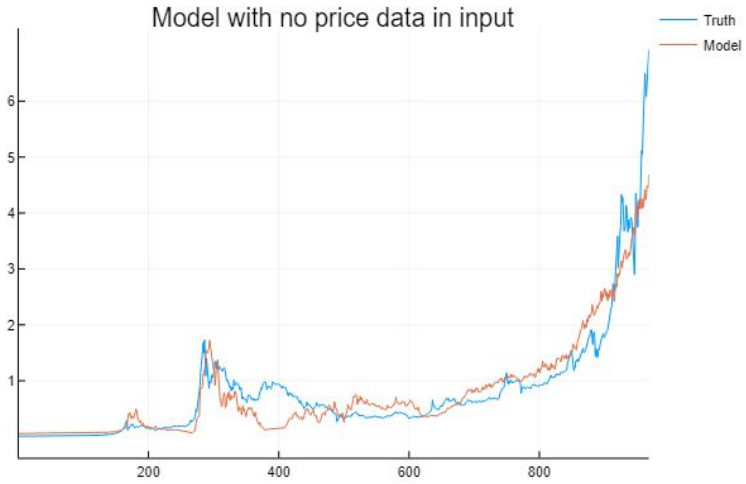
## Model Comparison:

We trained a model using every possible pair of the listed loss functions and regularizers (and some combinations of regularizers). For each pair we also tested each regularizer with a range of $\lambda$ values (except for the non-negative regularizer, because it does not require a weighting). To compare all of these models, we used cross validation with $k = 5$. In our cross validation, as the error metric for model evaluation, we used the mean absolute error in the predicted price for each

day in the test set. This seems to be the most practical error function to use based on how our model might be used in real world applications.

### Analysis Without Historical Price Data:

Based on our main analysis, we found that historical price plays the biggest role in all of the models, with a weight that is on average more than a full order of magnitude larger than the next largest weight. We wanted to verify that the other features contained useful information that they could contribute to our model. To determine this, we trained a model that used the four selected features but did not use historical price. This model performed surprisingly well. On the right is a graph of its predictions compared to the true price. Its approximate accuracy indicates that there is useful information in the other features that is improving our model.




### Analysis with Data on Political Events:

We tested all of the models a second time with the political events data included, and it had very little impact on the accuracy of the models. In fact, some of the models became less accurate. This may have been due to the sparsity of the events, which made it difficult for the model to learn. However, important political events that have a large impact on the price of Bitcoin are, by nature, infrequent, so it might be the case that it is simply difficult to meaningfully include them in a model. It might also be that some of the events had a larger impact than others, so the model had a hard time generalizing the effect of an event. In that case, we may be able to make an improvement by separating the events out into a couple of features based on the magnitude of the event.

# Results & Conclusion

After comparing all of the models as described above, we arrived empirically at the conclusion that the best performing model was the model that was trained with huber loss and non-negative regularization. This makes sense because huber loss is robust to outliers and optimizes for the

absolute loss that we used to evaluate the quality of our models, and each of our features has an intuitively positive relationship with the market price.

This model allowed us to predict the next day's price with an average error of $23.99 USD. This is 5% better than generic linear regression on the same features. However, that average error is heavily weighted by the most volatile sections of the data, where the error is high for both models. For series of days that are less volatile, our model performs much better than generic linear regression. Quantitatively, this can be seen in the fact that our model predicts within $1 of the true price on 173 days, compared to only 37 days for standard linear regression.]

Additionally, our model has relatively low percent error. It has an average percent error of 4.4%, compared to the generic linear regression model which has an average percent error of 14.1% when predicting the next day's price.

In conclusion, we have developed a model that is accurate during periods of minimal to normal volatility but that is not robust to the current extremes seen in the cryptocurrency markets. Therefore, we would not recommend using our model for real world investing at this time. Additionally, given the completeness with which we tested each of the relevant regression strategies from class, we believe that the volatile periods of Bitcoin's market price cannot be modeled using only tools from class.

# Discussion & Future Improvement

Our original plan was to develop a general algorithm for all kinds of cyber currency through analyzing bitcoin. However, access to the necessary data for other cryptocurrencies was limited and some of the features that our model relies on do not exist for other currencies. One possible improvement we could try in the future is to generalize our algorithm to model currency price from features that are more readily available for more cryptocurrencies.

Moreover, we determined that the model strategies from class will not be able to model the data with an acceptable accuracy, so we could do more research on other types of models. Potentially, we could try a classification model that simply predicts whether the price will go up or down.

In addition, as discussed above, including political event data did not improve our model and we may be able to deal with this by separating out the more influential events from the less influential events.