CS1951a Final Report

Thee Meensuk (tmeensuk), Petros Dawit (pdawit), Dikshyant Rai (drai), Kasemsan Kongsala (kkongsal)

Vision

Our “big idea” is that we want to create a predictive model of the Bitcoin price in US Dollars using other assets' historical prices and trading volumes. Given these past data, we want to be able to tell if we should buy or sell Bitcoin on any given day in order to maximize profits. The ultimate goal is to predict the exact prices in the future from past data.

Originally, we wanted to determine a universal equation that incorporates multiple variables to make predictions about Bitcoin’s future prices. Then we would do machine learning to determine certain unknown constants in the model (train it). However, as we investigated the topic more, we discovered many approaches that would likely lead to good predictions. Unfortunately, these approaches use very different variables that are irreconcilable by nature. Because we have to use different sets of data for each of these approaches, we found that it would help us gain more insights to try each approach separately, instead of trying to fabricate a complex universal financial model. Thus, we tried the approaches that most interested us and ended up testing 3 approaches to predict Bitcoin prices: the Mining Rate approach, the Macroeconomics approach, and the Trading Indicator approach.

Data

We mostly used Quandl’s API calls to retrieve our data. This datasource contains all the data we want to use for our computations. However, the data often contain noise, which is quite problematic to our models. Please see each sections about our data cleaning process, but, in general, we write our own python codes to makes the data fits into our models. We process

them using python's CSV library. All Bitcoin-related data are between January 2014 to March 2017. We picked this period because this is when Bitcoin is popular in general public, rather than in a small group. Below, please find the summary of the data used in each approach.

File	Mining Rate Approach	Trading Indicators Approach	Macroeconomics Approach
BCHARTS-BITSTAMPUSD.csv (Bitcoin Price in USD)	✓	✓	✓
BCHAIN-HRATE.csv (Total Bitcoin Hashrate)	✓	-	-
BAVERAGE-USD.csv (24h Average Bitcoin Price in USD)	✓	-	-
BCHAIN-TOTBC.csv (Bitcoin Market Capitalization)	✓	-	-
Bitcoin-cny-24hr-avg-4y.csv (24h Average Bitcoin Price in CNY)	-	-	✓
Bitcoin-eur-24hr-avg-4y.csv (24h Average Bitcoin Price in EUR)	-	-	✓
Bitcoin-usd-24hr-avg-4y.csv (24h Average Bitcoin Price in USD)	-	-	✓
Gold-price-4y.csv (Gold Price in USD, GBP, and EUR)	-	-	✓
us-treasurary-bond-10yr- 4yr.csv (Price of 10y US Treasurary Bond)	-	-	✓

Original Plan

After we decided on testing multiple Bitcoin's price prediction approaches, we agreed on 4 approaches. First, the Computational Power approach, later renamed the Mining Rate approach, utilizes the fact that the future amounts of Bitcoins can be very accurately determined from the publicly known hashing algorithm that generates new Bitcoins every second. The

hashing algorithm has a predetermined rate at which new Bitcoins are computed and generated. This model assumes that the expected price of Bitcoin is a function of computation power and market capitalization (total amount of $ circulating in the Bitcoin market).

In this case, we decided to use the $hashRate$ as the representation of computational power because it is a measurable, direct result of the computational power, and it is directly used to produced Bitcoin. For the $marketCap$, here, for simplicity, we used the market capitalization of Bitcoin against the USD (US Dollars). Thus, we had:

$$E(price) = f(hashRate, marketCap)$$

Next we looked at the Trading Indicator approach. We tried to use historical Bitcoin prices and volumes to predict the present and future. We started our guess as something like:

$$E(price) = f(p1, p7, p30, sd7, sd30, av7, av30, v7, v30)$$

where

$$p1 = 1 - day\ BTC\ Price\ Change$$
$$p7 = 7 - day\ BTC\ Price\ Change$$
$$sd7 = 7 - day\ Price's\ Standard\ Deviation$$
$$v7 = 7 - day\ Average\ Trading\ Volume$$
$$av7 = 7 - day\ Moving\ BTC\ PriceAverage$$
$$p30 = 30 - day\ BTC\ Price\ Change$$
$$sd30 = 30 - day\ Price's\ Standard\ Deviation$$
$$v30 = 30 - day\ Average\ Trading\ Volume$$
$$v30 = 30 - day\ Moving\ BTC\ PriceAverage$$

Third we considered the Macroeconomics approach. There are many financial assets whose prices are likely to correlate with the price of Bitcoin. We planned to use them in machine learning to find a model that can predict Bitcoin prices. Initially here, we were thinking that

currencies or currency-like assets would be most likely candidates to use in our model, including the Euro, Great Britain Pound, Chinese Yuan, Japanese Yen, Gold, Oil, and S&P500 stock market index.

Finally, we had the Black Market approach, where we would expect Bitcoin to appreciate along with the black market activities. We would track the black market activities in specific countries that use Bitcoin heavily in their dark net markets, such as India, Mexico, and China. We would find some indicators of the dark net activities and find their relationships with the Bitcoin price.

Real Execution

Mining Rate Approach

We believe that the total hashrate of the entire market of Bitcoin together with the total amount of Bitcoin in the market (market capitalization) have some influences on the price of Bitcoin. The reason behind this statement is that we view the production power (total computational power of the miners in a market) and market capitalization as the supply of the market, which should regulate the value of Bitcoin to make the market stay in equilibrium.

In order to learn more about this, we use different kinds of regressions to correlate the data from 50 days before today to predict the price of Bitcoin today. In sum, we gain some insights about the correlations of the hashrate, market capitalization and the price of Bitcoin.

In fact, this approach is quite different than the other approaches we used in this project: while other approaches try to precisely predict the price of Bitcoin at one point, the goal of this approach is to let us step back and see a big picture of the movement of the price and current trends. To do that, we also introduce the concept of value in addition to price, which will be discussed below.

Data Source and Cleaning

Since Quandl provides the API which allows us to pull the data in the csv format, we mostly use the data from,

https://www.quandl.com/collections/markets/Bitcoin-data

The data includes the price of Bitcoin, hashrate, market capitalization of a market. We use the data from January 2014 to March 2017, the period when Bitcoin become popular among the public. However, the data contains a lot of noise, especially the market price of Bitcoin, which has normal fluctuation as in any capital markets.

Since the price does capture the fluctuations of the price in the market as such fluctuations depend on interest rate risk, currency risk, operational risk, volatility risk, and other risks associated with the price that we might not know, we formalized our model by taking that fluctuations as a factor we cannot measure, captured by this equation,

$$realPrice_i = E(price)_i + \varepsilon_i$$

Where i is the day we are considering, $realPrice$ is the price of Bitcoin in one market against USD, and ε (epsilon) is the difference of the real price and the expected price. We do this because we think the expect price that we refer to is the real value of the Bitcoin, while the $realPrice$ (or, the market price) is the price that the market values because of other factors (like risks) that we mention above.

Please note the our functions are calculated daily. For example, ε for today's price might not be the same as ε of yesterday's price.

We illustration our concept of the relationship of $realPrice$, $E(price)$, and ε below.

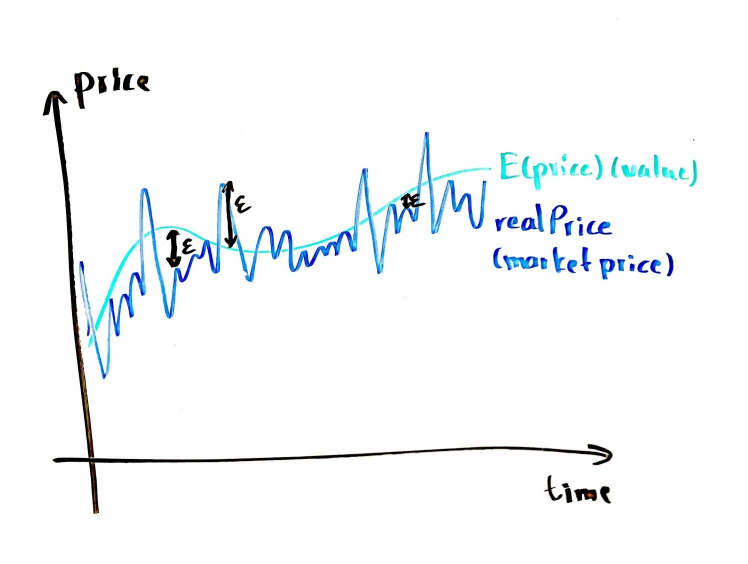


Figure 1: The relationship of $realPrice$, $E(price)$, and ε.

Now, we will work on how our prediction can reduce that ε we modeled. In other words, we need to smooth the curve so that we can eliminate the ε fluctuations. One way to do that is to use the Market Average, MA, for each day d, as defined below,

$$MA(length)_d = (\sum_{i=d-length}^{d} realPrice_i) / length$$

For example, today's value of $MA(50)$ is the average of the $realPrice$ from 50 days before today until today.

The next question we need to answer is that what argument we should supply to MA. Therefore, we illustrate $MA(10)$, $MA(50)$, and $MA(200)$ so that they might help us decide which one fits our model most.

In order to do that, we cleaned the data and compute our MA results with our Python code (CalculateMA.py). We illustrate them using excel. The illustration is shown below.

The red line is the $realPrice$, the green line is $MA(10)$, the blue line is $MA(50)$, and the purple line is $MA(200)$. Since all those three make sense, we think we can use any of them to use with our model. For this visualization, we will use $MA(50)$ because it does not mimic the $realPrice$

too much, and it is not too smooth. (In reality, we can pick any of them depending how far in the future we want to predict, but they might yield the results differently.)

Methodology and Results

We use $hashRate$ as the representation of the computational power and $marketCap$ being the the market capitalization of Bitcoin against the USD. We obtained the definition for the expected price as the function of $hashRate$ and $marketCap$ as follow,

$$E(price)_{i+k} = f(hashRate_i, marketCap_i)$$

Where i is the day we are considering, k is the number of days away from today that we are predicting in the future, $hashRate$ is the estimated number of giga hashes per second (billions of hashes per second) the Bitcoin network is performing, and $marketCap$ is the total number of Bitcoins in circulation the market price in USD.

This equation is our hypothesis. We believe that this hypothesis works because the expected price (the value) of day $i + k$ should be the result of the productivity of day i .

We, then, form our hypothesis that MA of day i is the expected price of day i:

$$MA(k)_i = E(price)_i$$

Next, we construct the model for $E(price)$. In this case, we believe that two variable linear regression might work. We, therefore, model that following,

$$MA(k)_i = f(hashRate_{i-k}, marketCap_{i-k})$$
$$MA(k)_i = \alpha \cdot hashRate_{i-k} + \beta \cdot marketCap_{i-k} + \gamma$$

To test this hypothesis, we create a csv file that contains 4 columns: date(i), $MA(50)_i$, $hashRate_{i-50}$, and $marketCap_{i-50}$, and stata to calculate the two variable regression. To do so, we ran commands that helps us bruteforce multiple model to fit our data as well as different configurations to apply our models

```
. // linear regression
. regress ma50 hash_rate total_cap
```

Source	SS	df	MS	Number of obs	=	1,169
				F(2, 1166)	=	1815.20
Model	36258131.2	2	18129065.6	Prob > F	=	0.0000
Residual	11645254.1	1,166	9987.35339	R-squared	=	0.7569
				Adj R-squared	=	0.7565
Total	47903385.3	1,168	41013.1723	Root MSE	=	99.937

ma50	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
hash_rate	.0005265	8.90e-06	59.15	0.000	.000509	.000544
total_cap	-.0002395	5.09e-06	-47.05	0.000	-.0002495	-.0002295
_cons	3530.089	67.18308	52.54	0.000	3398.276	3661.902

(Please see our imported data at “Data and Results.dta” using commands in “Mining Rate Approach.do”. In order to open the files, please use a computer with Stata 14 installed such as Brown University’s Windows or Mac computers)

Based on the regression analysis, we have, for

$$MA(50)_i = \alpha \cdot hashRate_{i-50} + \beta \cdot marketCap_{i-50} + \gamma$$

$$\alpha = 0.000526513738900334$$

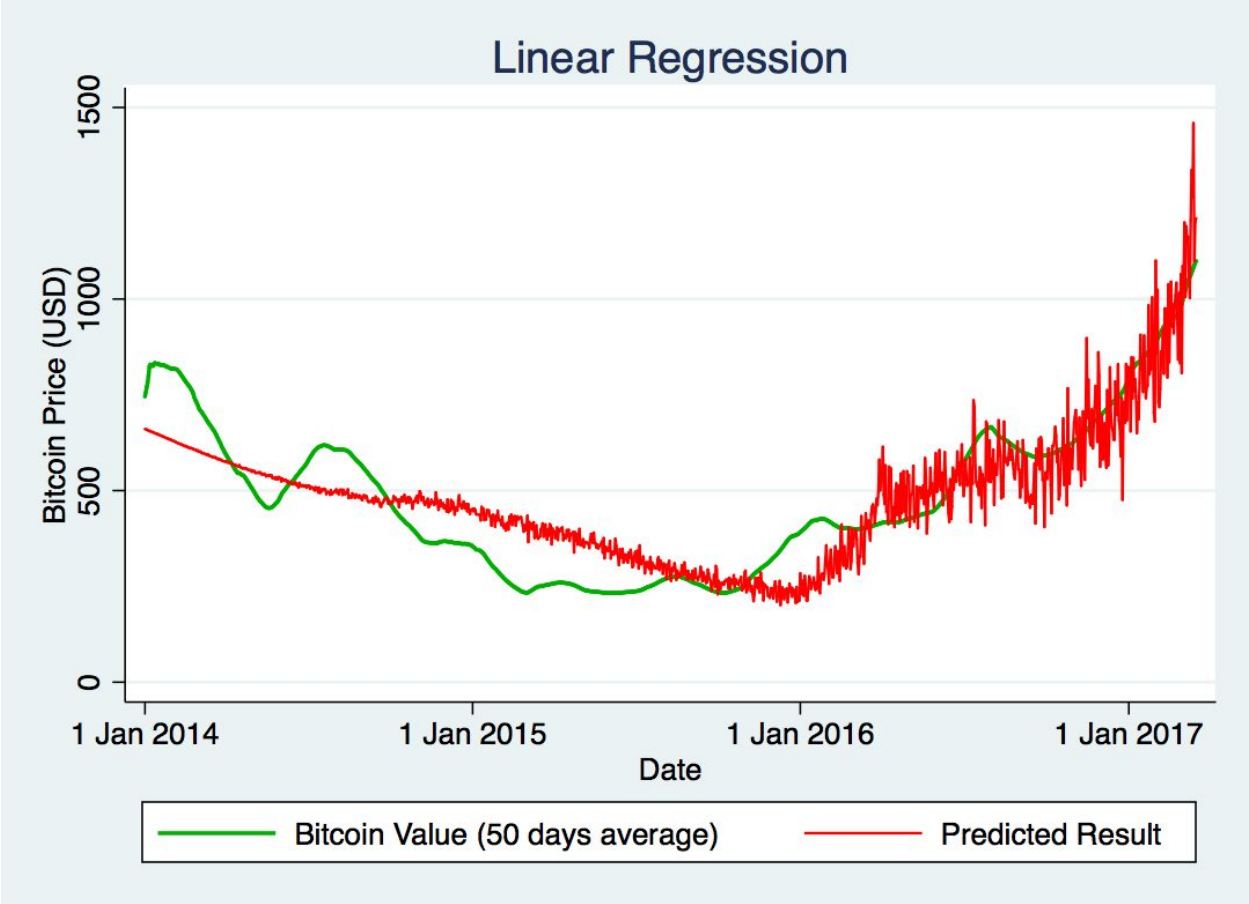
$$\beta = -0.00023952288238909$$

$$\gamma = 3530.08911436625$$

We use the equation to calculate the expected prices based on the hash rate and the market cap 50 days ago, we obtained the following csv file, with the first few rows shown below. This data is

1	date	ma(50)	hash_rate	total_cap	Expected Price
2	1/1/14	745.454163	4139.94817	11990400	660.293685
3	1/2/14	757.061522	4978.09719	11994475	659.7589262
4	1/3/14	770.89617	5847.42683	11999375	659.0429781
5	1/4/14	781.688949	4968.79793	12004200	657.42467
6	1/5/14	801.716088	4696.11999	12008300	656.2990575
7	1/6/14	824.200894	4938.50038	12012175	655.4985229
8	1/7/14	829.024166	4484.03716	12016250	654.283186
9	1/8/14	826.601623	5029.39303	12019950	653.6840887
10	1/9/14	825.344289	4908.20283	12024100	652.6262605
11	1/10/14	825.75343	5211.17832	12028150	651.8157135

Then, we plot our expected price (calculated from data 50 days before the day is is plotted) in red and $MA(50)$ in green.



Since the yellow line (the our estimation based on the data 50 days ago) can somewhat predict the green line (the real data on that day, according to our price/value definition), we believe our hypothesis works, but more data and analysis are needed to strengthen our hypothesis.

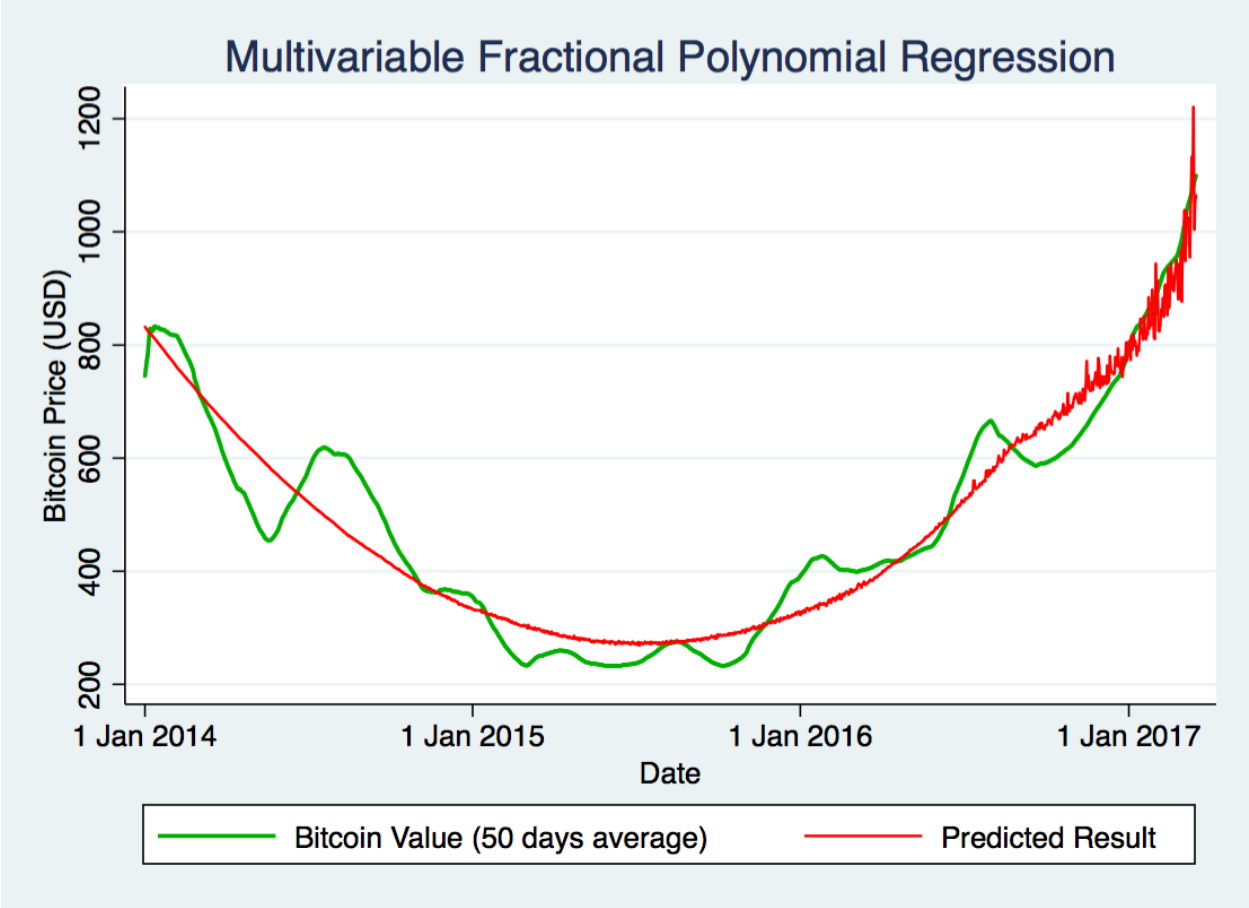
Thus, we extend this result by using stata to help us try different models by bruteforcing the degrees inputted and the types of the models. We found that Multivariable Fractional Polynomial Regression works well in our case. So we report the results as follow:

```
. mfp : regress ma50 hash_rate total_cap

Deviance for model with all terms untransformed = 14079.883, 1169 observations

Variable     Model (vs.)   Deviance  Dev diff.   P      Powers   (vs.)
-------------------------------------------------------------------------
hash_rate    lin.   FP2    14079.883   731.088  0.000+   1         .5 .5
             FP1           14079.883   731.088  0.000+   1
             Final           1.3e+04                    .5 .5

total_cap    lin.   FP2    13348.795   455.228  0.000+   1         3 3
             FP1           13321.739   428.173  0.000+   -2
             Final           1.3e+04                    3 3

-------------------------------------------------------------------------
End of Cycle 1: deviance =   12893.566
-------------------------------------------------------------------------

hash_rate    lin.   FP2    12857.581   121.329  0.000+   1         2 2
             FP1           12746.359    10.107  0.007+   3
             Final           1.3e+04                    2 2

total_cap    lin.   FP2    13830.485  1094.233  0.000+   1         3 3
             FP1           13578.967   842.715  0.000+   -2
             Final           1.3e+04                    3 3
```

ma50	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
Ihash__1	-38.12249	12.16596	-3.13	0.002	-61.99216	-14.25283
Ihash__2	55.70954	9.044839	6.16	0.000	37.96353	73.45555
Itota__1	-19.07393	.3894777	-48.97	0.000	-19.83808	-18.30977
Itota__2	6.391261	.1331542	48.00	0.000	6.130012	6.652511
_cons	256.7202	4.8756	52.65	0.000	247.1543	266.2862

Deviance:12736.252.

We use this generated model to plot a graph as follow,



Like the linear regression graph, the red line is generated from the data 50 days ago and the green line represents our definition of value of Bitcoin.

Conclusions and Discussions

Our models illustrates the big picture of the current trend in addition to other approaches in this project which precisely predicts the price of Bitcoin. We also discover that the hashrate is negatively correlated with the value of Bitcoin; the market capitalization, on the other hands, is positively correlated with the value of Bitcoin. This is quite contradict our intuition since the more the producers (higher the supply), the higher the price. We might explain that this is the case because the high demand for Bitcoin lately actually drives up the incentives to produce it more. We suggest that those who are interested in this approach reproduce our results using different datasets or define the value of Bitcoin differently.

Trading Indicators Approach

In the beginning, we were following our plan by doing a machine learning on the past Bitcoin prices. We used the closing prices of 7 prior days (whether the price went up or down) and fed it to the machine learning algorithm in Python (here we used kt-learn) to predict whether the price of the Bitcoin today will go up or down.

The data we collected came from Quandl and had information on open price, high price, low price, close price, volume (currency) and weighted price with ranges of dates from March 2017 until September 2011.

First, we pre-processed the data. We used excel to quickly create our csv file. Our csv file contained 10 columns. The first column was the date, and the second was the closing price for that date. The third column was called ‘Label’ which was determined by comparing the current closing price to the day prior. If today’s price was greater than that of the day prior, we would consider the label to be 1, otherwise 0. The next 7 columns after were ‘X Days Prior’ where X was a number ranging from 2 to 8. These columns were filled with 1s and 0s and described whether or not the price for those dates had gone up, represented by 1, or down, represented by 0 from X days prior (we were comparing the closing prices).

Below is an image of part of our csv file data.

Date	Close	Label	2 Days Prior	3 Days Prior	4 Days Prior	5 Days Prior	6 Days Prior	7 Days Prior	8 Days Prior
3/17/17	1087.88	0	0	0	0	0	0	0	0
3/16/17	1172.62	0	0	0	0	0	1	0	1
3/15/17	1257.32	1	1	1	1	1	1	1	1
3/14/17	1245.86	1	1	1	1	1	1	1	0
3/13/17	1242.46	1	1	1	1	1	1	0	0
3/12/17	1226.62	1	1	1	1	0	0	0	0
3/11/17	1176.55	1	0	1	0	0	0	0	0
3/10/17	1116.97	0	0	0	0	0	0	0	0
3/9/17	1190.89	1	0	0	0	0	0	0	0
3/8/17	1150.05	0	0	0	0	0	0	0	0
3/7/17	1233.05	0	0	0	0	0	1	1	1
3/6/17	1278.49	1	1	0	1	1	1	1	1
3/5/17	1273	1	0	1	1	1	1	1	1
3/4/17	1260	0	1	1	1	1	1	1	1
3/3/17	1285.33	1	1	1	1	1	1	1	1
3/2/17	1257.6	1	1	1	1	1	1	1	1
3/1/17	1226.39	1	1	1	1	1	1	1	1
2/28/17	1191.21	0	1	1	1	1	1	1	1
2/27/17	1194.64	1	1	1	1	1	1	1	1
2/26/17	1179.05	1	0	0	1	1	1	1	1
2/25/17	1152.2	0	0	1	1	1	1	1	1
2/24/17	1180.14	0	1	1	1	1	1	1	1
2/23/17	1188.11	1	1	1	1	1	1	1	1
2/22/17	1130.01	1	1	1	1	1	1	1	1
2/21/17	1124.62	1	1	1	1	1	1	1	1
2/20/17	1084	1	1	1	1	1	1	1	1
2/19/17	1051.8	0	0	1	1	1	1	1	1
2/18/17	1056.4	1	1	1	1	1	1	1	1
2/17/17	1055.46	1	1	1	1	1	1	1	1
2/16/17	1032.7	1	1	1	1	1	1	1	0
2/15/17	1011.53	1	1	1	0	1	1	0	0
2/14/17	1008.88	1	1	0	1	1	0	0	0
2/13/17	1000.79	1	0	1	1	0	0	0	0
2/12/17	1000.73	0	1	1	0	0	0	0	0
2/11/17	1012.4	1	1	0	0	0	1	0	0
2/10/17	996.08	1	0	0	0	0	0	0	0
2/9/17	986	0	0	0	0	0	0	0	1

When we did the machine learning, we used three models in sk-learn's modules: Naive Bayes, Logistic Regression, and Linear SVM. We found the exact same results for all three models. That is same values for training mean accuracy, cross validation score (mean and standard deviation) and predicted mean accuracy.

Naive Bayes Model:

Training Mean Accuracy	Cross Validation Score		Predicted Mean Accuracy
	Mean	Standard Deviation	
0.630681818182	0.630738208429	0.0487526200048	0.578947368421

Logistic Regression Model:

Training Mean Accuracy	Cross Validation Score		Predicted Mean Accuracy
	Mean	Standard Deviation	
0.630681818182	0.630738208429	0.0487526200048	0.578947368421

SVM Model:

Training Mean Accuracy	Cross Validation Score		Predicted Mean Accuracy
	Mean	Standard Deviation	
0.630681818182	0.630738208429	0.0487526200048	0.578947368421

At first glance these same results seemed to indicate an error in our machine learning. However, after careful investigation, it was not impossible that we got the same results for all three models. After all, these models shared the same target, which was to most accurately predict the labels (buy/sell signals) from past price movements. We checked the results given from these models, and found that all models predicted the same buy/sell signal given the same feature set. Having checked that there was no error in the code, we concluded that our model was too simplistic in that...

(1.) We turned the change in price from detailed numbers into just 0's and 1's, thus destroying the nuances among the dataset. Certain combinations of middle-ground prices could have led to some differences in prediction, but by converting the information into just 0's and 1's, we eliminated the chance for the middle-ground data to differentiate themselves, resulting in all middle-ground data giving the same prediction.

(2.) We used a feature set that was too small, making the categorizations for our models too rough. We only took 7 past days for the feature set to train our model. This could have been fine if we were considering inherently different features. However, in this case the features were just prices. The only difference between them was the date that the prices were recorded. Furthermore, the range of time that we considered was only up to 8 days prior, meaning there was little discernable difference between those days. Bitcoin prices tend to move in trends and not a random walk. This means that if the price movements 2, 3, 4, 5, 6, 7, and 8 days prior to a given day had more 1's than 0's (more days when prices increased than decreased), the model would just predict a "buy" signal, and if more 0's than 1's, a "sell". There were only 7 days that we used in the feature set, so a 4-3 increase-decrease would simply signal a buy (label 1), and vice versa. In other words, if there were 4, 5, 6, or 7 days when the price had increased, it was "obvious" to just predict a buy. An extreme example would be doing machine learning on a single feature: whether or not a person has ever jumped off a 100-ft building (1 if yes, 0 if no). Then even when we use different models to predict whether that person is still alive after jumping, it should be "obvious" to just predict <jump == dead> and <no jump == alive>. Similarly, because the 3 models that we used (Naive Bayes, Logistic Regression, and SVM) were targeting the same label prediction, and there wasn't much difference among the models given only the 7-day feature set, it was "obvious" to just predict "buy" if there were 4, 5, 6, or 7 days when the prices had increased. Later on, when we tested with higher numbers of days, the identicality among these models broke down, and we could see that the models actually gave different predictions.

Having learned from these mistakes, we redesigned our Trading Indicator approach to be more complex, thus allowing for nuances and incorporating more numbers of days. This time, we wanted to apply machine learning with Naive Bayes, Logistic Regression, and Support Vector Machines (SVM) to learn and train our data across 7 technical trading indicators that professional traders frequently use. We calculated from the data to predict whether or not to buy or sell a stock the next day. We would calculate these indicators individually and then merge the indicators with their dates as a key and clean the data from there. We would check if each date had all 7 indicators. Some dates wouldn't because some indicators were calculated over time and the original data does have some gaps in the dates. The original data can be found in the 'data' folder with the name 'BCHARTS-BITSTAMPUSD.csv'

Moving on, the 7 indicators we chose to look at were Simple Moving Average (SMA), Volume Weighted Average Price (VWAP), Rate of Change of Price for 20 days (RCOP), Stochastic Indicator for 5 days (STOCH), a Momentum Indicator (MOM), Percentage Price Oscillator (PPO) and a Moving Average Convergence Divergence (MACD). The final csv with these values can be found at 'trading_trend_v2.csv'.

We had to do calculations a second time as we didn't get good machine learning results the first time and we figured we made a mistake in our original calculations. Some of these indicators predicted all buys or all sells, and this wasn't useful information. Our second implementation with the correct calculations doesn't have this. We also normalized our values. You can find the code for the calculations in the 'trading_trend_v2.py' file. As you can see down below, we have a description of each indicator and the formulas we used to calculate the values for training our model.

Simple Moving Average (SMA):

The simple moving average calculates the average price over a time period. Our time period was 14 days.

Formula: SMA = sum of price over n period / n period

Volume Weighted Average Price (VWAP):

The volume weighted average price is an indicator, like the simple moving average, that computes a weighted average of the price over a given time period. The weights are the the volume transacted in the given time period.

Formula: vwap = Σ(price(i)*quantity(i) / Σ quantity(i)

Where 0 < i < n

Rate of Change of Price for 14 days (RCOP):

The rcop calculates the rate of change of the current price over a previous price. Our previous price was 14 days ago.

Formula: ROCP (14 days) = Current Price/ Price 14 days ago

Stochastic Indicator for 5 days (STOCH):

A stochastic indicator is the one that normalizes price in percentage between 0 and 100. We used that Fast %k stochastic indicator which you can see the formula below. Our time period was 14 days.

Formula:

Fast %k = ((close-low)/(high-low)) * 100

close = closing price of most recent transaction

low = lowest price in given time period

high = highest price in a given time period

Momentum Indicator (MOM)

The momentum indicator calculated the current price minus a price n days ago. Our time period was 14 days.

Formula: mom = current price - price 14 days ago

Percentage Price Oscillator (PPO)

This indicator calculates the difference between a fast moving average and a slow moving average. Our fast moving average was the time period halved while the slow moving average was the time period. The time period was 14 days.

Formula: PPO = (FastMA - SlowMa)/SlowMa * 100

Moving Average Convergence Indicator (MACD)

For this indicator, we compared the 12, 26 and 9 day exponential moving averages. Below is our conversion chart for denoting the values:

Compare MACD = <12-day EMA> vs <26-day EMA> vs <9-day EMA>

12-day EMA > 26-day EMA < 9-day EMA = +2

12-day EMA > 26-day EMA > 9-day EMA = +1

12-day EMA < 26-day EMA < 9-day EMA = -1

12-day EMA < 26-day EMA > 9-day EMA = -2

Accuracy Scores

Using 7 indicators, we test 7 + 1 cases for their accuracies for 3 models. Each of the first 7 cases uses one of the trading indicators listed above. The last case is when we do machine learning with all the 7 features combined into one feature set. We are using 3 models: Naive Bays (NB), Logistic Regression (Log), and Support Vector Machine (SVM). Thus, there are 24 accuracy scores in total.

Accuracy scores SVM algorithm for different Features							
sma	vwap	rcop (20 days)	stoch (5 days)	mom	ppo	macd	All Combined
46.23%	46.44%	58.86%	56.21%	58.25%	58.68%	56.42%	48.27%

Accuracy scores NB algorithm for different Features							
sma	vwap	rcop (20 days)	stoch (5 days)	mom	ppo	macd	All Combined
58.04%	57.80%	58.85%	58.85%	58.85%	58.85%	56.41%	55.19%

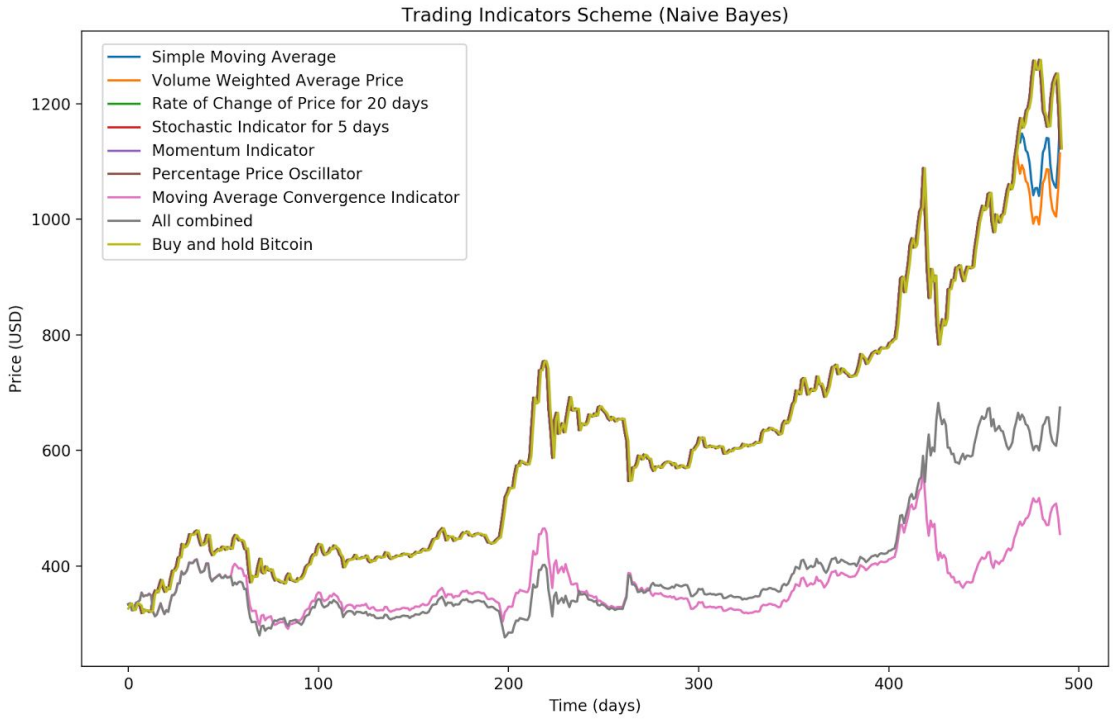
Accuracy scores LOG algorithm for different Features							
sma	vwap	rcop (20 days)	stoch (5 days)	mom	ppo	macd	All Combined
46.02%	46.02%	58.85%	56.62%	58.85%	58.85%	56.41%	48.47%

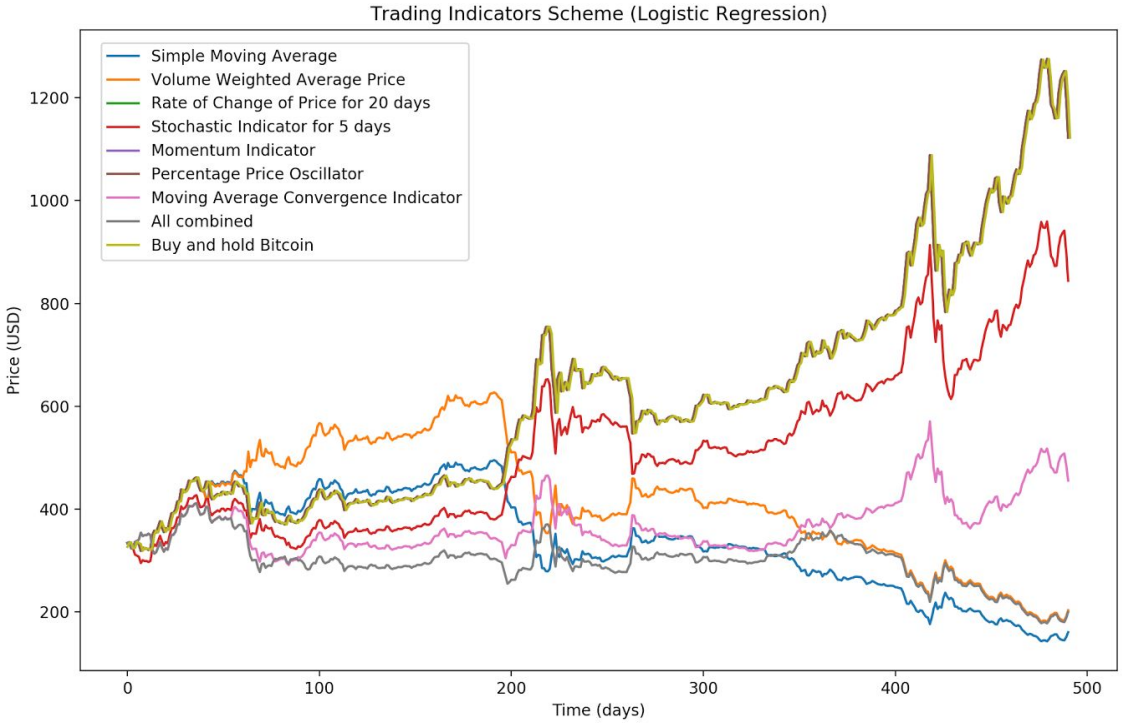
Additional Visualization

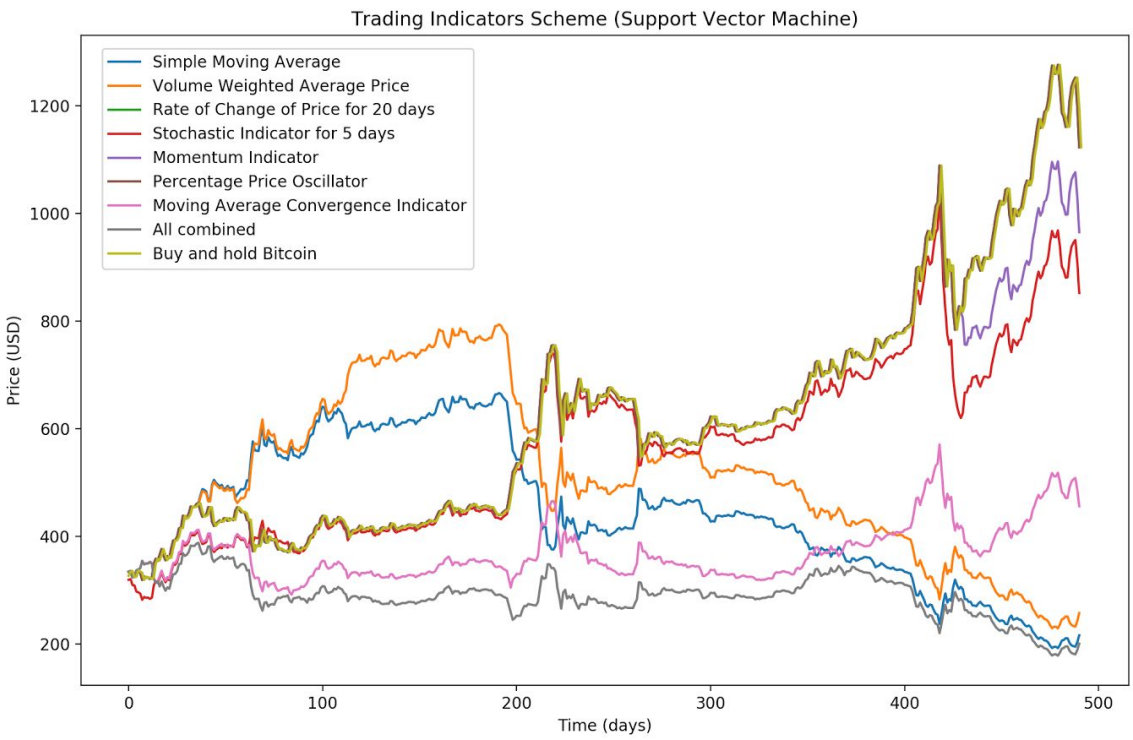
To aid in understanding how each of our trained models perform, we have created 3 graph visualizations, with each graph for each model. In every graph, we start out with 7+1 = 8 cases, as mentioned above (Each of the first 7 cases uses one of the trading indicators listed above.

The last case is when we do machine learning with all the 7 features combined into one feature set.). These 8 cases get 1 Bitcoin each in the beginning of the testing period.

Then we let them buy or sell according to the signals generated from our model, and then compare with the actual price movements in the past. If the model predicts a buy and the price increases, we gain money equal to that increase. If the model predicts a buy and the price decreases, we lose money equal to that decrease. If the model predicts a sell and the price increases, we lose money equal to that increase. If the model predicts a sell and the price decreases, we gain money equal to that decrease. We keep performing these buy/sell tests every day, until the last day of the test period. Then we compare how each of these indicators does. In addition to the 8 predictive indicators, we also add a benchmark, which is when we have one Bitcoin in the beginning of the period and does nothing (no buy or sell trades) until the end. This is equivalent to an indicator signaling “buy” every day. Finally, we compare the gains from these indicators with the benchmark where we does nothing other than just holding one Bitcoin until the end.


Trading Indicators Scheme (Naive Bayes)
Simple Moving Average
Volume Weighted Average Price
Rate of Change of Price for 20 days
Stochastic Indicator for 5 days
Momentum Indicator
Percentage Price Oscillator
Moving Average Convergence Indicator
All combined
Buy and hold Bitcoin
Price (USD)
Time (days)


Trading Indicators Scheme (Logistic Regression)
Simple Moving Average
Volume Weighted Average Price
Rate of Change of Price for 20 days
Stochastic Indicator for 5 days
Momentum Indicator
Percentage Price Oscillator
Moving Average Convergence Indicator
All combined
Buy and hold Bitcoin
Price (USD)
Time (days)



Results and Analysis

Some of the accuracy scores for the Trading Indicators approach exceed 50%, which is the threshold we want to pass. Having accuracy scores over 50% means that some of our trading indicators were able to predict better than a coin toss whether or not the Bitcoin price would go up or down on a given day, considering that the Bitcoin price has an equal chance to go up as well as to go down. However, if we look at the accuracy table, there are many 58.85% or 58.86%. These are the accuracy scores of certain indicators predicting “buy” most of the time. These indicators are momentum indicators that will try to follow the trend and suggest the buy/sell signal according to the trend. Since Bitcoin has been trending upward in the recent years, it is not surprising that these momentum indicators will suggest “buy” signals to match the appreciating Bitcoin price. Thus, if we ask a middle school kid to give trading signals like this,

and he says “buy” all the time, we would still have the 58.85% or 58.86% accuracy scores as well because in the recent years Bitcoin has been going up more than down.

Unfortunately, 58.85 and 58.86% are both the accuracy scores from just following the trend and the highest accuracy scores in our results. This means our best indicators are going to be performing as well as simply having one Bitcoin in the beginning of the test period and holding it until the end. Thus, we have been unable to beat the market (which is equivalent to just buying and holding Bitcoin). This conclusion is confirmed by our 3 graphs where all of them have the Buy-Hold strategy at the top of the net worth ranking by the end of the period.

Anyway, while we have not been able to beat the market throughout the test period, some of our indicators actually outperformed it for an amount of time. In the Logistic Regression and Support Vector Machine models, the Simple Moving Average and Volume Weighted Average Price both gave us higher net worths than the Buy-Hold benchmark from day 0 until approximately day 200. After that point, these two indicators plummet down and never recover again. Even though they do not consistently beat the market, we discovered that it is possible to find indicators that might do well over a certain period of time. In this case, the Simple Moving Average and Volume Weighted Average Price do well from day 0 to day 200 and outperform the market. We also notice that these two indicators are especially good at making money during the time when Bitcoin price has no trend and is fluctuating within a small range. After Bitcoin starts an upward trend, these 2 indicators lose their predictive power. This means that if we use these indicators during the appropriate times, such as using Simple Moving Average when there is no trend, and using others when the trend is upward, we might be able to beat the market.

In fact, we discovered that a more sophisticated algorithm with layers of conditions would have been able to generate a better prediction of the Bitcoin price. We could have a trend indicator checking for the price momentum of Bitcoin, and if there is no momentum, we can use

one set of indicators (like Simple Moving Average). If there is high momentum, we can use another set of indicators. Even more complicated layers could be added to the conditions after we check for momentum. This way, we would be able to pick the appropriate indicators to make good predictions.

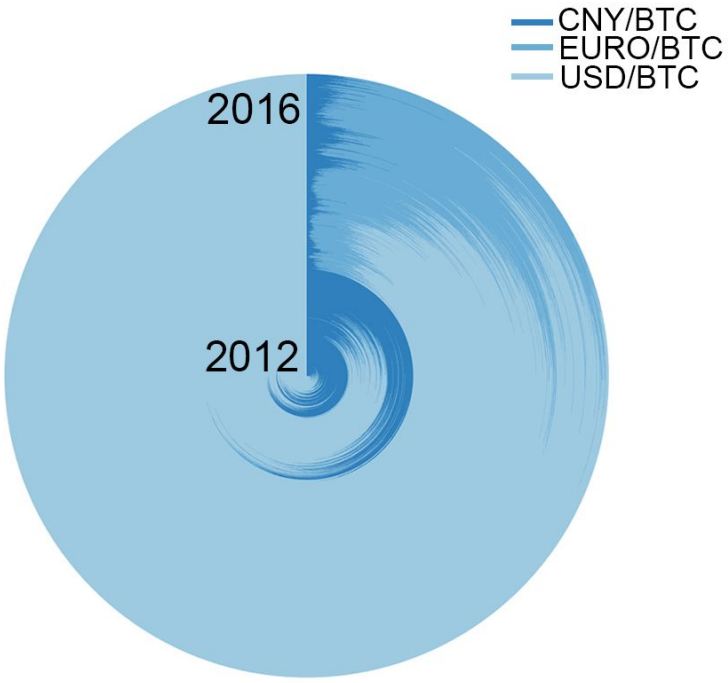
Macroeconomics Approach

For this model to predict Bitcoin prices we used machine learning. We used classifiers such as Naive Bayes, Logistic Regression and Support Vector Machines. In order to feed such classifiers we had to come up with some good sets of features. To accomplish this task, there are two things we need to consider: the time span of past data and the types of assets we want to incorporate in our model.

Since we are trying to use the past data to predict the future prices of Bitcoin, we want to consider how far we should go back in the past. Because Bitcoin is essentially a financial asset, too far back in the past will have little impact on the present or future, and using those data will count against us. On the other hand, only a few days prior to the date of prediction might not provide us with enough data to make good predictions. Thus, we expect the optimal time frame to be in the middle, not too short or too long. In order to find out how long, we tried different time frames and compared the results. We found out that the optimal learning period should be somewhere between 10 and 80 days for our purposes.

The types of assets to be used as features are important as well. We have brainstormed and came up with 4 major assets whose prices will likely correlate with Bitcoin's. The assets we chose were Chinese Yuan (CNY), Euro (EUR), US Treasury Interest Rates and Gold. We want to use these assets as the main bases for our features to be used in our machine learning model. However, we will also transform some of these features to make the models more realistic, which we hope will help increase the accuracy of our prediction.

Visualization for Currency Choice



It is not out of arbitrary discrete that we chose the Chinese Yuan and Euro as the only 2 currencies for our Macroeconomics model. We picked those 2 currencies out of many others because in addition to their large economic sizes, the Chinese Yuan and Euro have relatively high trading volumes in the Bitcoin market, meaning they should have high economic relationship with the value of Bitcoin as well. We demonstrate their trading volumes through the visualization above.

The circle above represents the proportion of trading volumes for Chinese Yuan, Euro, and US Dollars in the Bitcoin market. Our data starts from year 2012 to year 2016. In 2012, represented by the most inner part of the circle, we can see a very high proportion of the Chinese Yuan volume. At certain periods, there are even more Bitcoins traded in Chinese Yuan than in US Dollars. However, due to the Chinese government's crackdown on Bitcoin, there have been sharp decreases in the Chinese portion of Bitcoin's trading volume. However, as you can see, after the crackdown and the sudden decrease in Chinese volume, the Chinese proportion has been gradually increasing until the end of our dataset (year 2016, represented by the outermost layer of the circle).

The Euro was insignificant in the beginning of our dataset. In 2012, the Euro was almost non-existent. However, because of the open market and an increasing popularity of Bitcoin, the

Euro volume has been rapidly increasing. At the end of our dataset, in 2016, the Euro accounted for almost 1 out of 4 Bitcoin transactions in the world.

Raw Data

The data extracted from https://www.quandl.com/ give us the historical Bitcoin prices in US Dollar (USD), Chinese Yuan (CNY), and Euro (EUR), as well as the interest rates for 10-year US Treasury Bonds and Gold Prices in USD. For the Bitcoin prices, the data goes daily from September 2013 to October 2016, daily. There are 365 days in a year, so we have about 1400 data points.

	A	B	C	D	E	F
1	Date	24h Average	Ask	Bid	Last	Total Volume
2	10/31/2016	702.09	702.57	701.99	702.56	33358.89
3	10/30/2016	708.94	699	698.54	698.76	30953.5
4	10/29/2016	708.56	718.19	717.54	717.71	41680.39
5	10/28/2016	689.81	691.3	690.58	691.07	32503.1
6	10/27/2016	684.89	685.64	685.54	685.58	39446.96
7	10/26/2016	667.12	675.01	674.39	674.91	47584.95
8	10/25/2016	659.05	663.36	663.12	663.28	32655.7
9	10/24/2016	654.36	654.05	653.63	653.93	27473.1
10	10/23/2016	657.27	657.6	656.94	657.2	17879.04
11	10/22/2016	646.39	658.32	657.5	658.1	34537.66
12	10/21/2016	636.76	635.53	634.9	634.93	26549.17
13	10/20/2016	634.72	634.36	633.79	634.01	19569.68
14	10/19/2016	637.51	632.66	632.13	632.52	33026.46
15	10/18/2016	643.75	640.68	640.37	640.56	22898.31
16	10/17/2016	643.35	642.25	641.28	642.09	24763.11
17	10/16/2016	648.06	647.5	646.98	647.05	11160.09
18	10/15/2016	646.81	645.51	645	645.48	10915.63
19	10/14/2016	642.45	642.71	642.33	642.55	24246.38
20	10/13/2016	641.6	640.12	639.85	640.09	23130.15
21	10/12/2016	642.49	640.04	639.81	639.81	23440.73
22	10/11/2016	633.27	641.99	641.76	641.82	46050.51
23	10/10/2016	619.99	620.03	619.64	620.04	22035.54
24	10/9/2016	624.33	620.02	619.52	619.85	11578.78
25	10/8/2016	623.15	624.5	624.38	624.46	12804.13

Label and Feature Transformation

From the raw data, we want to craft a set of features and label to train with machine learning. For the label, we will just want to classify whether or not the price of Bitcoin on a given day is higher than the day before. We give 1 for higher and 0 for same or lower.

To get features, we will consider each asset individually, and one additional case where we use all the 4 assets to do the machine learning. First, for the Chinese Yuan, CNY, the raw data has the price of one Bitcoin in terms of CNY. However, this feature contains two sets of information both in itself: the exchange rate between CNY and USD, or CNY/USD Exchange rate, and the value of Bitcoin in USD. In other words, <CNY/BTC> = <CNY/USD> * <USD/BTC>. Thus, if the value of Bitcoin appreciates (which it did), while the exchange rate fluctuates within a small range, the resulting CNY/BTC will go up wildly as the <USD/BTC> part goes up.

However, we want to find features to predict the label. Our labels, +1 if the Bitcoin price goes up, and 0 if the price goes down, should not depend on whether or not the absolute price of Bitcoin is high or low. If that were to be the case, since the price of Bitcoin has appreciated sharply in the recent years, the more recent prices would all indicate one direction of price change, and almost all prices in the past would indicate the other direction. That means our model would be predicting decreases in Bitcoin prices every day in the long past up until a point in time when the model switches to predicting increases in prices every day until today.

To eliminate the <USD/BTC> part that represents the absolute value of Bitcoin (in USD), we can divide <CNY/BTC> on each day by <USD/BTC> of the same day. This leaves us with only <CNY/USD>, which could be a good predictor of whether or not the price of Bitcoin in USD would go up or down on a particular day. To think about economic realities, the CNY/USD represents the exchange rate between USD and CNY. If CNY/USD decreases, it means $1 is equivalent to less CNY, or that the Dollar has depreciated. If the Dollar depreciates, $1 should be able to buy fewer Bitcoins. Conversely, 1 BTC will be equivalent to more Dollars. That means higher USD/BTC. Given this logic, it makes sense that we would be able to predict whether or not USD/BTC will increase or decrease just by looking at CNY/USD, calculated from <USD/BTC> and <CNY/BTC>.

Now that we have extracted the CNY/USD out from the raw data, we also want to include the concept of “time” into our feature. Currently there is no “time” in our CNY/USD data, but we want to predict the future from the past. Thus, we include time into our feature by

creating more CNY/USD at different times. As a result, we will have CNY/USD 1 day before today, 2 days before, 3days before, and so on. Because we are running machine learning with different time frames, the number of features will vary with the length of our time frame. For example, if we are to do the 40-day time frame case, there would be 40 features representing each of the 40 previous days of CNY/USD exchange rates. However, the corresponding label remains the USD/BTC price.

The next case is to predict the same set of labels using EUR/USD on previous days. The process is the same as in the CNY/USD case except that we change CNY to EUR.

Next is to use the 10-year US Treasury Yield on previous days. The Treasury Yield is straightforward, so we don't need to do extra feature transformation here.

Next, we use the Gold price in USD/Ounce. We picked Gold because Gold has always been the alternative currency for thousands of years. Until recently, when people lost confidence in major currencies like the US Dollars, Gold was seen as the "Safe Heaven" for investors and central banks alike to park their money. Bitcoin is often compared to Gold as the direct competitor for the alternative currency status. As Bitcoin gained more popularity and people began using it for transactions, Gold movement should be a very good indicator for Bitcoin's price to go up or down.

Finally, we combine every feature into one batch of ML. That is, in the first 4 parts we predict labels from only one of CNY/USD, EUR/USD, 10-year Treasury Interest Rate, and Gold. If we are considering the 30-day learning period, we will have 30 features, each corresponding to each of the previous 30 days. In this final machine learning attempt, if we consider the 30-day period, we will have 120 features, broken down into 30 CNY/USD, 30 EUR/USD, 30 10-year Treasury Interest Rates, and 30 Golds.

Feature Scaling

We also want to scale the features down to carry similar weights. The set of features that we put in our machine learning model may be at different scales. There are features such as interest rates which is at the range of 0 to 100 and there are other features like EUR/USD which is in the scale of 0 to 1. It would be quite difficult to perform regression analysis on such data and harder

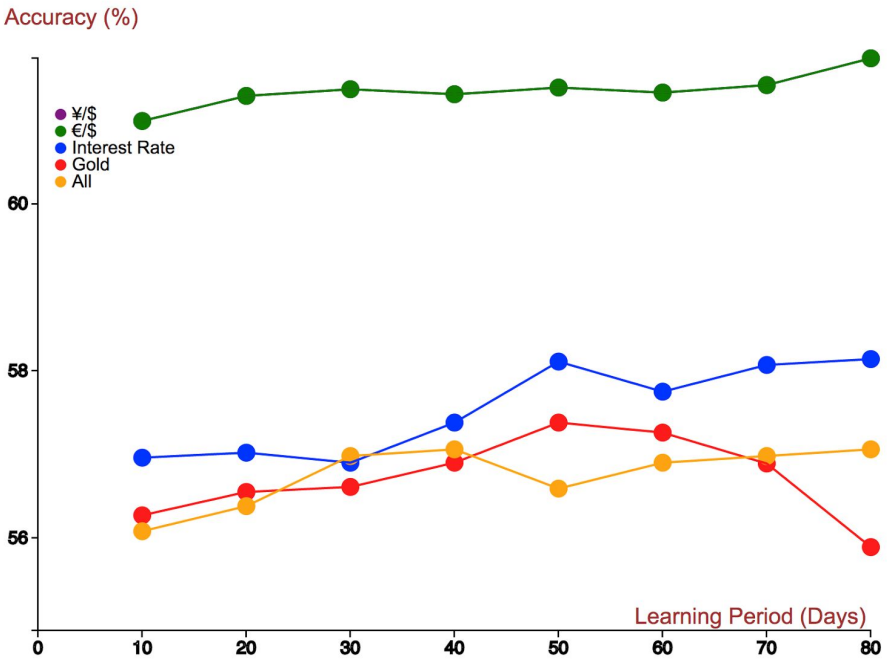
for a gradient descent algorithm to reach the global minimum. So, in order to tackle such inconsistencies in feature sets, we decided to scale all of the features using min-max normalization scheme. For each feature set we would subtract the features value with minimum value and divide by that value with range which is (maximum - minimum). In this way every features such as CNY/USD, EUR/USD, Interest Rate, and Gold will be in a range of 0 to 1 and our classifier will be more efficient.

Model's Accuracy Scores

We did machine learning with three different models: Naive Bayes, Logistic Regression, and SVM (Support Vector Machine).

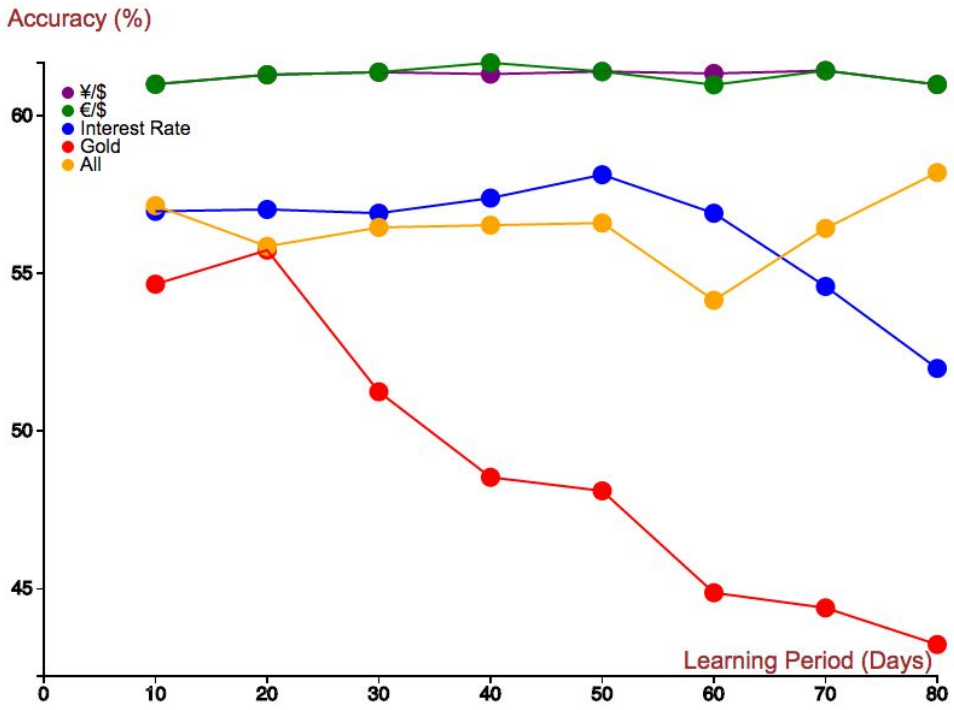
Using the aforementioned features, we trained our models with different sets of features described above, using 75% of the Bitcoin data (75% oldest data) we collected. Then we tested the trained models against the 25% most recent data and arrived at the following accuracy results.

Naive Bayes



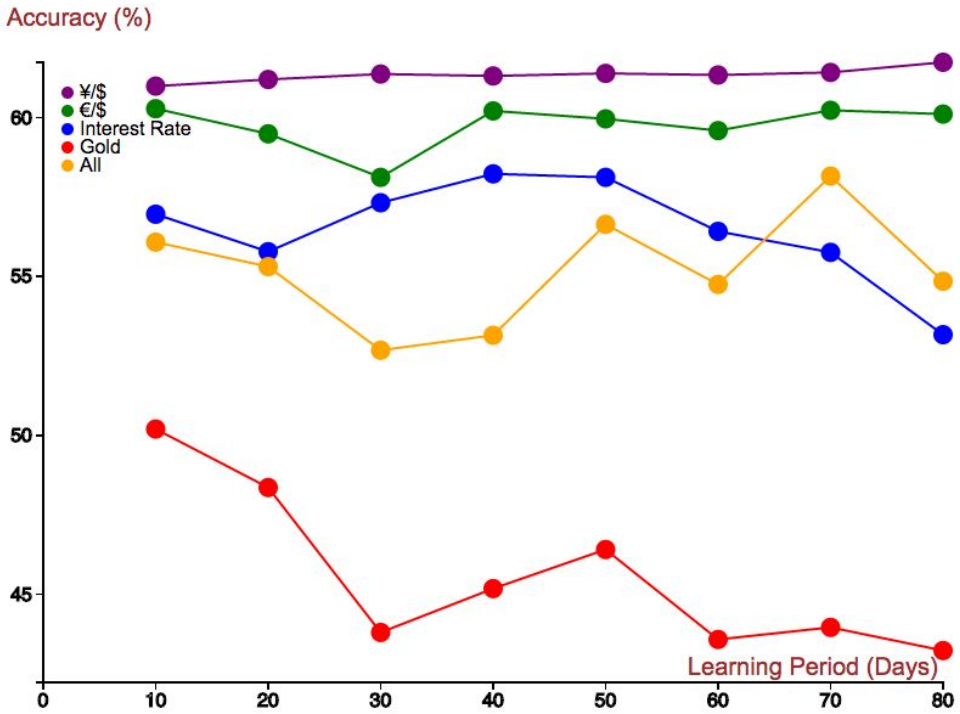
Feature Set	CNY/USD Alone	EUR/USD Alone	10-year US Treasury Yield (%) Alone	Gold ($/Ounce) Alone	All of them combined
10 Days	60.99	60.99	56.96	56.27	56.08
20 Days	61.29	61.29	57.02	56.55	56.38
30 Days	61.37	61.37	56.90	56.61	56.98
40 Days	61.31	61.31	57.38	56.90	57.06
50 Days	61.39	61.39	58.11	57.38	56.59
60 Days	61.33	61.33	57.75	57.26	56.90
70 Days	61.42	61.42	58.07	56.89	56.98
80 Days	61.74	61.74	58.14	55.89	57.06

Logistic Regression



Feature Set	CNY/USD Alone	EUR/USD Alone	10-year US Treasury Yield (%) Alone	Gold ($/Ounce) Alone	All of them combined
10 Days	60.99	60.99	56.96	54.65	57.14
20 Days	61.29	61.29	57.02	55.73	55.85
30 Days	61.37	61.37	56.90	51.23	56.45
40 Days	61.31	61.67	57.38	48.53	56.52
50 Days	61.39	61.39	58.11	48.10	56.59
60 Days	61.33	60.96	56.89	44.87	54.14
70 Days	61.42	61.42	54.58	44.39	56.42
80 Days	61.74	60.98	51.98	43.23	58.19

Support Vector Machines (SVM)



Feature Set	CNY/USD Alone	EUR/USD Alone	10-year US Treasury Yield (%) Alone	Gold ($) Alone	All of them combined
10 Days	60.99	60.28	56.96	50.20	56.08
20 Days	61.29	59.49	55.78	48.36	55.31
30 Days	61.37	58.12	57.32	43.80	52.68
40 Days	61.31	60.21	58.22	45.18	53.15
50 Days	61.39	59.96	58.11	46.41	56.64
60 Days	61.33	59.59	56.42	43.58	54.75
70 Days	61.42	60.29	55.76	43.96	58.15
80 Days	61.74	60.11	53.17	43.23	54.85

Results and Analysis

Using coin tosses to predict the Bitcoin price performance would have resulted in approximately 50% accuracy. Thus, any score above 50% is better than a coin toss. We don't expect any score higher than 70% because any model with such high accuracy and simplicity should have been discovered by someone else long before, and that person must have exhausted most profit opportunities to the point that there is little economic benefit left for us. If any score higher than 70% shows up, we expect to surpass Bill Gates' net worth within 10 years.

Our results show a reasonable range of accuracy scores from 43% to 62%. The worst-performing model is the Naive Bayes, which makes sense because the Naive Bayes tries to measure the posterior mass probability rather than assuming correlations among variables, thus deviating from the reality where the data is continuous rather than discrete. This forces the model to make assumptions that the data is discrete rather than continuous. This is further reflected in the Naive Bayes' accuracy score graph shown above, where the accuracy scores for CNY/USD and EUR/USD are the same (thus one line is completely hidden behind the other). After we saw the result from Naive Bayes model, we realized that the continuous nature of our data should make it unsuitable for the Naive Bayes model which is good for discrete data. However, because we somehow achieved the accuracy scores above 60% when using CNY/USD and EUR/USD, we included the Naive Bayes result to confirm the consistent predicting power of CNY/USD and EUR/USD in guessing the buy/sell signals.

The Logistic Regression (Log) and Support Vector Machine (SVM) also boast very high accuracy scores (60%) for CNY/USD and EUR/USD. However, using Gold price as a feature seems to lower the accuracy, at least in the case of Logistic Regression and Support Vector Machine. In cases where we train the models too long (such as training for 60, 70, and 80 days in SVM), using Gold as a feature makes the accuracy score drop below 50%, which is quite upsetting.

Whatever model we use, we see a general trend in the accuracy scores. The typical pattern is that the accuracy score goes up when we consider a longer time window, up until a point when the accuracy drops as we increase the length. Individual feature sets perform differently with the CNY/USD and EUR/USD being generally superior to the other two feature sets (10-year US Treasury Interest Rate and Gold). The CNY/USD and EUR/USD exchange rates can maintain quite consistent accuracies throughout different learning periods. From 10 days to 80 days, the CNY/USD and EUR/USD both seem to be around 60% accuracies for Naive Bayes, Logistic Regression, and SVM. Meanwhile, for Log and SVM, the 10-year US

Treasury Interest Rates peak at 40-50 days before dropping down, and the Gold peaks at 10-20 days before plummeting. When we combine every feature, the accuracy scores seem to be between the higher ones (CNY/USD and EUR/USD) and lower ones (US Treasury and Gold). The Naive Bayes produces mixed results, with Gold peaking at 50 days, Treasury not peaking below 80 days, and All Features Combined having unusually low accuracies.

Black Market Approach

For the Black Market Approach, we discovered that the data in the Darknet are encrypted and very hard to get meaningful results from them. Thus, we decided to discontinue using this approach.

Final Conclusion and Discussion

We tried 3 main approaches to predict the future prices of Bitcoin based on past data: the Mining Rate approach, the Trading Indicators approach, and the Macroeconomics approach. In our Mining Rate approach, we learned that the hashrate is negatively correlated with the value of Bitcoin. On the other hand, the market capitalization is positively correlated with the value of Bitcoin. Moreover, we were able to predict the future price trend of Bitcoin based on essentially these two variables. Regardless of the high speculation in the market, the price trend that we were able to outline from the hash rate and market capitalization nicely predicted the actual prices of Bitcoin. This confirms that even though there could be speculation, irrationality, and greed, the Bitcoin market is still governed by the law of economics: the demand and supply that eventually lead to the final price.

The Trading Indicator approach yielded disappointing results because we were not able to outperform the Buy-and-Hold strategy (market) using any of the 24 cases of indicators that we had chosen. However, we discovered that certain indicators were able to outperform the market for a short period of time when the marketing was going sideway instead of trending up. This discovery has led to a promising conclusion that if we have a set of conditions that can first categorize the state of the Bitcoin market (such as whether or not the market is trending up or going sideway), and then for each categorization use the appropriate indicators for the market state, we might be able to beat the market. For future analysis, we are proposing that one

performs 2 types of machine learning. The first machine learning can be used to categorize the state of the market, and the second type of machine learning is to find the best combination of indicators for each state. This way, we might be able to outperform the market using trading indicators. It is also noteworthy that since we were simply using textbook trading indicators for the features in our model, it could also be concluded that because too many people knew about these simple trading indicators, most economic profits had already been competed away, and it made sense that we did not beat the market from such indicators. However, if we are able to invent multiple layers of indicators, put together through machine learning, we might be successful in inventing new ways to predict the future based on past data.

The Macroeconomics approach gives reasonably high accuracy scores in predicting the future Bitcoin price using the past data from other financial assets. This makes sense because there is clearly some economic relationship between Bitcoin and each of these assets. For the CNY/USD exchange rate, since the Chinese Yuan is the currency as well as Bitcoin, they are both similarly competing for the investors' money. Suppose that CNY/USD exchange rate decreases, it makes sense for people holding US Dollars to buy more Chinese goods because the same amount of USD leads to more CNY and more Chinese goods. In this case, there will be less money going to Bitcoin and thus lower Bitcoin price in US Dollars. For the Euro, it is also the same. For the 10-year US Treasury, or any interest rate in general, if the interest rate goes up, people will want to hold more US Dollars because they get more interests, thus lowering the demand for Bitcoin decreasing the Bitcoin price. Gold is also the same. If Gold price increases, it makes sense to buy Bitcoins that are cheaper. If Gold price decreases, it might be better to sell Bitcoin and buy Gold, lowering the price of Bitcoin. All in all, these financial assets have certain relationships with Bitcoin, and knowing some data about them can help us predict the price of Bitcoin.

We conclude that the difference in successes among the 3 approaches was caused by the amount and quality of data in each approach. Judging from our results, among the 3 approaches that we tried, the more information that we put in, the more accurate our prediction became. The worst-performing approach was the Trading Indicators approach where we only considered past Bitcoin prices. However, the other two approaches gave very good results. This is because the Mining Rate incorporated the hash rate information, which further added to the information about just past Bitcoin prices alone. In addition, the Macroeconomics approach also added information about the other financial assets' prices. Because these 2 approaches used

additional data beyond just plain past Bitcoin prices, they performed better than the Trading Indicators approach.