Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis

Stuart Colianni, Stephanie Rosales, and Michael Signorotti

1 Abstract

Past research has shown that real-time Twitter data can be used to predict market movement of securities and other financial instruments [1]. The goal of this paper is to prove whether Twitter data relating to cryptocurrencies can be utilized to develop advantageous crypto coin trading strategies. By way of supervised machine learning techniques, our team will outline several machine learning pipelines with the objective of identifying cryptocurrency market movement. The prominent alternative currency examined in this paper is Bitcoin (BTC). Our approach to cleaning data and applying supervised learning algorithms such as logistic regression, Naive Bayes, and support vector machines leads to a final hour-to-hour and day-to-day prediction accuracy exceeding 90%. In order to achieve this result, rigorous error analysis is employed in order to ensure that accurate inputs are utilized at each step of the model. This analysis yields a 25% accuracy increase on average.

2 Introduction

Cryptocurrency is an alternative medium of exchange consisting of numerous decentralized crypto coin types. The essence of each crypto coin is in its cryptographic foundation. Secure peer to peer transactions are enabled through cryptography in this secure and decentralized exchange network. Since its inception in 2009, the Bitcoin has become a digital commodity of interest as some believe the crypto coins' worth is comparable to that of traditional fiat currency.

Considering the exchange rates of cryptocurrencies are notorious for being volatile, our team strives to develop an effective trading strategy that can be applied to a variety of cryptocurrencies. Our method for determining the optimal time to trade involves correlating prices with one of today's most popular social media sources, Twitter. The advantages of using Twitter include having access to some of the earliest and fastest news updates in a concise format as well as being able to extract data from this social media platform with relative ease.

Our trading strategy applies supervised machine learning algorithms including support vector machines, logistic regression, and Naive Bayes to determine whether the price of a particular digital currency will increase or decrease within a predetermined time interval. The two approaches for training these classifiers involve using direct text, otherwise known as *tweets*, from Twitter users and using third party open-source sentiment analysis APIs to rate the positivity and negativity of words within each post. Both of these training methods prove to be effective in estimating the trajectory of cryptocurrency prices. In order to predict market movement to a particular granularity, a time series of tweets equal in length to the trading period is required one cycle beforehand. This time series of Twitter posts is used as an input to the classifiers.

3 Related Work

Applying machine learning to cryptocurrency is a relatively new field with limited research efforts. Using Bayesian regression, Shah et al. achieved an 89% return on investment over fifty days of buying and selling Bitcoins [8]. Another approach predicted the price change of Bitcoin using random forests with 98.7% accuracy [4]. These approaches fail to consider the feelings of individuals about Bitcoin, and therefore, fail to harness these potential features in their learning algorithms. Twitter sentiment analysis has been widely researched. Bollen et al. utilized the Profile of Mood States (POMS) to predict the movement of the Dow Jones Industrial Average with 87.6% accuracy. Go et. al focused only on classifying tweets and used several approaches to achieve an accuracy of 84.2% with Multinomial Naive Bayes, 79.2% with maximum entropy, and 82.9% using a support vector machine [2]. This paper will expand on the approaches of researches in the past and apply Twitter text classification and sentiment analysis to cryptocurrency markets.

4 Data

In order to create a training and testing data set for the learning algorithms, we utilize Tweepy - an open-source Python library for accessing the Twitter API [10]. The keyword,*bitcoin*, is searched in real time and tweets containing this token is placed into a text file. Additional data being collected for each post containing the keyword includes the user ID, a unique identifier which cannot be changed, and a time stamp. In addition, the prices of the cryptocurrency is collected every hour via the cryptonator.com API and placed into text files to create a

price history [6].

While tweets are collected in real time, excess white space is removed and the text is changed to lowercase. To clean the data, the following procedure is carried out. The first step is to remove all non-alphabetic characters. The second step is to remove duplicates. The reason for doing this is because of the prevalence of Twitter bots, many of which instantaneously disseminate tweets containing particular keywords. Not removing these tweets will cause the distribution of words in our training set to be skewed. Invalid English words which remain are identified and removed based on not having membership in the "words" corpus of the Natural Language Toolkit [5]. Stop words are subsequently removed from tweets based on membership in the "stopwords" corpus of the Natural Language Toolkit. We then create two data sets: one with stemming and one without. In the stemming set, all remaining words are stemmed using the Porter Stemming algorithm. Then, entries which no longer contain words are removed from the data set.

The duration of the data collection process was 463 hours spanning over twenty-one days. During this period, over 1 million tweets pertaining solely to Bitcoin were collected. The processed data set consists of over 350,000 individuals who posted at least once about Bitcoin. The distribution of the number of Bitcoin related tweets per user is exhibited in Figure 1. Please note that the x axis is in log scale due to the large variance in tweets per user.

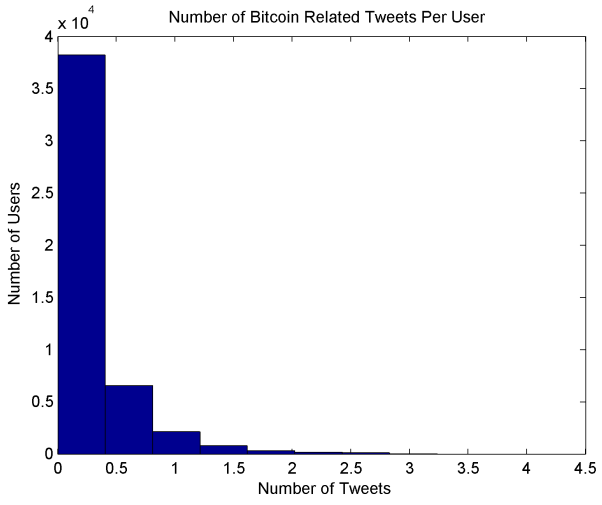

Fig. 1. The number of Bitcoin related tweets per user is displayed above. The x axis is in log scale. The data was collected from November 15, 2015 to December 4, 2015.

As displayed in Figure 1, the majority of individuals contributed only a few times throughout the data collection period. Although some users tweet far more than others, the individuals with the most posts only contribute to a fraction of the 1 million tweet data set. Half of all users who posted at least one tweet about Bitcoin only tweeted once about the digital currency.

In order to ensure accurate prediction results across all days, it is important to consider whether the number of data points each day is fairly consistent. Figure 2, which displays the number of tweets that mention Bitcoin per day, proves the posting frequency for this particular class of tweet is relatively uniform.

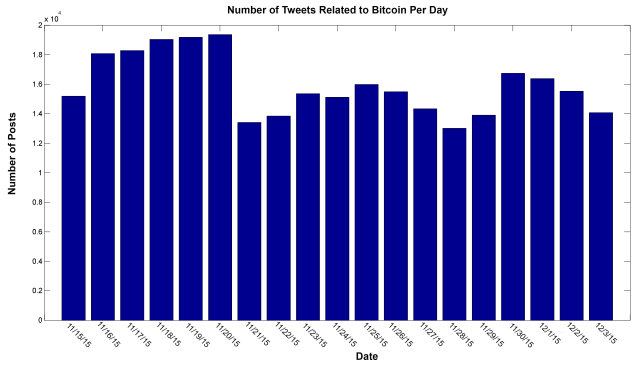

Fig. 2. The above chart shows the number of Bitcoin related tweets per day from November 15, 2015 to December 3, 2015. The dates included are limited to days with continuous data collection.

5 Method

In order to determine digital currency market movement with the Twitter data set, text classification and sentiment analysis algorithms are utilized. The goal of each algorithm is to predict whether the price of Bitcoin will increase or decrease over a set time frame. For the text classification approach, the implementations of Naive Bayes, logistic regression, and support vector machines in the Scikit Python library are utilized [7]. Training and testing on sentiment analysis data requires the same implementation of support vector machines and logistic regression. Although both types of algorithms are trained on the same data set, the fundamental approaches to formatting each model's feature vector is quite different.

5.1 Feature Vectors

For the examples below, suppose that we wish to reconstruct the processed version of the following tweet into a feature vector (without the removal of stop words).

Bitcoin has a bright future in the world's economy.

The features for text classification consist of a vector of all unique words in the data set lexicon. Since the vector encompasses all possible unique entries, it is sparse even for the longest of tweets. Suppose that there are n words in the learning algorithm's vocabulary. An example of a text classification feature vector for this model with each entry $x \in \{0, 1\}^n$ is as follows.

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ bright \\ \vdots \\ economy \\ \vdots \\ future \\ \vdots \\ 0 \\ \vdots \\ worlds \\ \vdots \\ 0 \end{bmatrix}$$

When a particular word is observed at least once, a binary value of one is recorded in the position of that word in the feature vector. When the total count of each word is represented in the same format of feature vector, the input is modeled as multinomial rather than Bernoulli. Therefore, the entries in a multinomial feature vector will take on values $x \in \{1, 2, ...k\}^n$.

Training and testing feature vectors for sentiment analysis models are fundamentally different. In order to generate feature vectors of this structure, preprocessed tweets are analyzed word-by-word in the text-processing.com API [9]. This API returns scores between zero and one for words' positivity, negativity, and neutrality. These scores are aggregated into a single vector similar to the one below.

$$x = \begin{bmatrix} 0.80 \\ 0.76 \\ .20 \end{bmatrix}$$

5.2 Naive Bayes

The Naive Bayes is a generative learning algorithm which is commonly applied to text classification and sentiment analysis machine learning problems. This approach to text classification utilizes the first format of feature vector where the appearance of a word is modeled by either the Bernoulli or multinomial distribution. In both versions of this algorithm, we assume the x_i variable given Y in the Naive Bayes mathematical program below to be conditionally independent of one another.

$$\underset{y_j}{\text{argmax}} \;\; P(Y = y_j) \prod_{i=1}^{m} P(x_i | Y = y_j)$$

In this formulation, y_j represents the classification of whether the Bitcoin price is increasing or decreasing over a predetermined time interval. The variable x_i is the feature vector for tweet i where a total of m tweets are collected. Since this is a generative learning algorithm, a model can be built for both positive and negative variation in the market. For each observation in the training set, the above product of probabilities is calculated assuming each market trend, and the results are compared. The classification resulting in the higher probability is assumed true and subsequently assigned to that particular post.

5.3 Logistic Regression

Discriminative learning algorithms such as logistic regression are also useful in the field of text classification and sentiment analysis. Unlike generative learning algorithms, this model examines two classes in the training set and determines the best separation. The logistic regression learning algorithm can be derived by maximizing the following likelihood function.

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

In this likelihood function, $x^(i)$ takes the form of either of the previously mentioned feature vectors. The index, i, maps the feature vector to one of the observations in the training set of size m. The exponent, $y^{(i)}$, represents the state of the market for feature vector i. The function h_θ is the sigmoid function below.

$$g(z) = \frac{1}{1 + \exp^{-z}}$$

In order to determine an update rule for the parameters, theta, the log likelihood function can be formulated. This function can then be differentiated in order to derive the stochastic gradient ascent formula below.

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

After reaching convergence, the parameters, theta, are utilized in the sigmoid function in order to classify the state of the digital currency market.

5.4 Support Vector Machines

Support vector machines are supervised learning algorithms that can perform nonlinear classifications by mapping data to higher dimensions through the use of the kernel trick. Support vector machines are an effective tool in sentiment analysis as proven by Go [2]. The L1 Norm Soft Margin model below can be trained with either the text classification feature vector or the sentiment analysis feature vector.

$$\begin{aligned} \min \;\; & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m} \xi_i \\ s.t. \;\; & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \qquad i = 1, ..., m \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, ..., m \end{aligned}$$

This support vector machine has the L1 norm soft margin formulation which includes a penalty term for points which are not linearly separable. This penalty term for this model is below.

$$C \sum_{i=1}^{m} \xi_i$$

The term, ξ_i, is a slack variable. The penalty equation acts as a trade off between having a large separation between terms and incorrectly classifying observations. The feature vector, $x^(i)$, is used with both the text classification and sentiment analysis forms outlined earlier in the section. The variable, $y^{(i)}$, is the observed class for a particular observation.

5.5 Training and Testing

Training and testing the models for the text classification and sentiment analysis problems involves nearly identical steps. We partitioned the training set in a 70-30 split where 70% of the data is reserved for training and 30% is marked for testing. When performing this random sampling, we assure there is an equal representation of tweets from both classes while maintaining an accurate timeseries representation. After training the model on this balanced training set, we record the rate at which samples are classified correctly with respect to the test set data points. We repeat this procedure ten times and aggregate the correct classification rates over all iterations.

6 Results

We tested classifiers using two data sets: one where the tweet had been stemmed using Porter stemming and one without. To assess the performance of different classifiers, we computed the accuracy of each. In every case, the accuracy of the classifier was on par or significantly better when operating on the unstemmed data.

In our first classification attempts, we treated the words in a tweet as elements of the feature vector for each classifier. Bernoulli Naive Bayes performed the best out of all text classification algorithms by achieving a day to day accuracy of 95.00% and an hour to hour accuracy of 76.23%.

Fig. 3. Accuracy Using Tweet as Feature Vector

	Bern. NB	Mult. NB	Log. Regression	SVM
Day-to-Day	95%	95%	85.5%	83%
Hour-to-Hour	76.23%	69.83%	74.44%	43.69%

In our second classification attempts, we used the text-processing.com API to calculate negativity, neutrality, and positivity scores for each tweet. The API also returns a positive, neutral, or negative label. These labels were used as the feature vectors for Naive Bayes (Bernoulli and Multinomial). The return scores were used as the feature vectors for the classifiers. Logistic Regression performed the best using this feature vector achieving a day to day accuracy of 86.00% and an hour to hour accuracy of 98.58%.

A confusion matrix along with the precision, recall, and F-Score were calculated for the most accurate hour to hour classifier (Logistic Regression using sentiment scores) and the most accurate day to day classifier (Bernoulli Naive Bayes and Multinomial Naive Bayes using the tweet as a feature vector).

Fig. 4. Accuracy Using Sentiment as Feature Vector

	Bern. NB	Mult. NB	Log. Regression	SVM
Day-to-Day	55.0%	55.0%	86.0%	53.5%
Hour-to-Hour	52.73%	52.42%	98.58%	47.03%

Fig. 5. Confusion Matrix - Bernoulli NB Day-to-Day

	Positive	Negative
True	11, (100%)	8, (72.72%)
False	0, (0%)	1, (9.09%)

Fig. 6. Confusion Matrix - Logistic Regression Hour-to-Hour

	Positive	Negative
True	254.1, (100%)	225.8, (88.86%)
False	1.8, (0.71%)	5.3, (2.09%)

The precision matrix for day-to-day results has a precision of 1, a recall of 0.92, an accuracy of 0.95 and an F-Score of 0.96. The negative predicted value calculated to 0.89, and the true negative rate calculated to 1. The precision matrix for hour-to-hour results has a precision of 0.99, a recall of 0.98, an accuracy of 0.986 and an F-Score of 0.99. The negative predicted value calculated to 0.98, and the true negative rate calculated to 0.99.

7 Error Analysis

Our initial accuracy for predicting the hour to hour sign change of Bitcoin using the Bernoulli Naive Bayes classifier was 59.0%. In order to determine the possible error locations within the machine learning pipeline, we thoroughly inspected inputs at each step. During one of the data processing steps, we noticed many near duplicate posts that differed based on numerical ID values within the tweets' text. In order to remove these automated postings, we adjusted the data cleaning procedure by removing all non-alphabetic characters before removing duplicate tweets. In order to ensure these automated postings were removed, we computed the Levenshtein distance from each tweet to every other tweet. The Levenshtein distance is the edit distance between two strings. This value can be leveraged to prove that any pair of tweets is dissimilar by at least some threshold value. After cleaning, the new data set was roughly 50% smaller, but yielded a significantly better classification accuracy of 76.23%.

8 Future Work

In order to further improve the accuracy of the learning algorithms, additional research can be performed in the area of error analysis. An improvement that can be explored for text classification algorithms involves accounting for

negation as outlined by Jurafsky et al [3]. According to Jurafsky, an efficient method of accounting for negation in text analysis is to prepend the prefix, *not_*, after a negated word. An additional modification that can be made to the training set is to ensure that the training set has an equal number of words associated with each classification. Although our sets are relatively equal (8061 words in the price decrease set and 8852 in the price increase set), creating a training set that is completely unskewed could result in lower classification error. In addition, we can formulate a set of words where each element has a high correlation with cryptocurrency market movement and use this as a basis for training the learning algorithms. This adjustment will result in sparser feature vectors for text classification and possibly more accurate predictions.

REFERENCES

[1] Bollen, Johan, and Huina Mao. *Twitter Mood Predicts the Stock Market*. http://arxiv.org/pdf/1010.3003\&. 14 Oct. 2010. Web. 12 Nov. 2015.

[2] Go, Alec, Lei Huang, and Richa Bhayani. *Twitter sentiment analysis*. Entropy 17 (2009).

[3] Jurafsky, Daniel. *Classification: Naive Bayes, Logistics Regression, Sentiment*8, 2015. Web. 12 Dec. 2015.

[4] Madan, Isaac, Saluja, Shaurya, and Aojia Zhao, *Automated Bitcoin Trading via Machine Learning Algorithms*, Department of Computer Science, Stanford University.

[5] *Natural Language Toolkit*. Natural Language Toolkit NLTK 3.0 Documentation. 2015. Web. 11 Dec. 2015

[6] *Online Bitcoin Wallet*. Cryptonator. 2014. Web. 11 Dec. 2015.

[7] *Scikit-learn*. : Machine Learning in Python 0.17 Documentation. Web. 11 Dec. 2015.

[8] Shah, Devavrat and Kang Zhang *Bayesian Regression and Bitcoin*. http://arxiv.org/pdf/1410.1231v1.pdf. 6 Oct. 2014. Web. 12 Nov. 2015.

[9] *Text-processing.com*. API Documentation for Text-processing.com Text-processing.com API 1.0 Documentation. Web. 11 Dec. 2015.

[10] *Tweepy*. Tweepy. Web. 11 Dec. 2015.