Sentiment Analysis Of Twitter Data Related To Bitcoin Tweets And Comparison of Bitcoin Price Change with Sentiments

CSC869 Data Mining Term Project

Submitted By
Sabiha Hussain Barlaskar

917932501

05/22/2018

Contents

1. A brief Overview of Bitcoins

Bitcoin is a form of digital currency that is encrypted. The transactions occur without any intervention of some central authority. The transactions are encrypted and saved in a public distributed ledger called a Blockchain.

2. Problem Statement

Since using Bitcoin and cryptocurrencies are on the rise, I wanted to analyze the correlation between Bitcoin price change and social sentiments. Just like efficient AI algorithms like deep learning can be used to analyze stock prices, I wanted to analyze the pattern of bitcoins from a social media perspective. I chose this problem because it gave me the opportunity to explore two different sets of data as well introduced me to the field of Text mining and Natural Language processing.

The problem that I have worked on aims towards two goals:

a. Sentiment Analysis on tweets tweeted about Bitcoin
b. Is there a correlation between Twitter sentiment and bitcoin price change?

3. Literature Review of related work

Natural language processing is a challenging field requiring a lot of manual intervention for labelling the tweets/sentences/comments as positive and negative in order to train the data. In order to extract the features by a classifier, it needs to be fed a huge amount of data consisting of a combination of different sentences. Debjyoti Paul, Feifei Li, Murali Krishna Teja, Xin Yu, Richie Frost[1] have performed Sentiment analysis of US election using Stanford's Twitter data consisting of 1.6 million tweets to train their classifier. In 2009, Alec Go, Richa Bhayani[2], and Lei Huang from Stanford extracted the data based on emoticons and performed sentiment analysis on it using different classifiers.

4. Datasets:

Twitter datasets:
I used Tweepy which is a Twitter API, to extract the tweets using the following code:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth,wait_on_rate_limit=True)

# Open/Create a file to append data
csvFile = open('data/tweets_positive', 'a')
#Use csv Writer
csvWriter = csv.writer(csvFile)

for tweet in tweepy.Cursor(api.search,q="#bitcoin AND :)",
                           lang="en",
                           since="2018-05-15", until="2018-05-15").items():
    print (tweet.created_at)
    csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
```

I followed an emoticon based approach, where I used Tweets that were classified as positive/negative based on ☺ and ☹. I collected tweets having these emoticons and extracted the data related to the period May 08,

2018 – May 20, 2018. Initially, I collected data from March 1st – April 30th, but that data didn't contain emoticons.

Size of the data set	41049 data points	Negative-11406, Positive - 29643
Size of the set after removing duplicates	9400	Negative-1759, Positive - 7622
Attributes	Created_at, Tweet Text	
Missing data	19	

Table 1: Twitter data set description

Bitcoin dataset:

To extract Bitcoin data, Coindesk API can be used. But I downloaded from the CoinDesk website from 08 May – 20th May, 2018. https://www.coindesk.com/price/.

Size of the data set	312 data points
Attributes	Date, Closing Price

Table 2: Bitcoin data set description

5. Main Strategy

a. Cleaning and data preprocessing:

1. **Cleaning the data**: I cleaned the data to remove unnecessary characters like ":", "\", characters other than numbers and letters.
2. **Removing the duplicates**: There were around 31,649 duplicates in my training data. I removed them using Pandas.
3. **Removing the stop words:** I used NLTK's stop words removal strategy and removed the punctuation marks like "?", "", "," etc.
4. **Removed tweets containing "Like", "Follow", "Share":** There were a lot of tweets having these words. I added these words to the stop words since filtering out these words removed an entire tweet. Adding these to the stop words, just removed these words but not the entire sentence.
5. **Removed hex characters.**

b. Data Visualization:

i. Tweets by Location:

Figure 1: Tweets by location

ii. WordCloud Visualization:

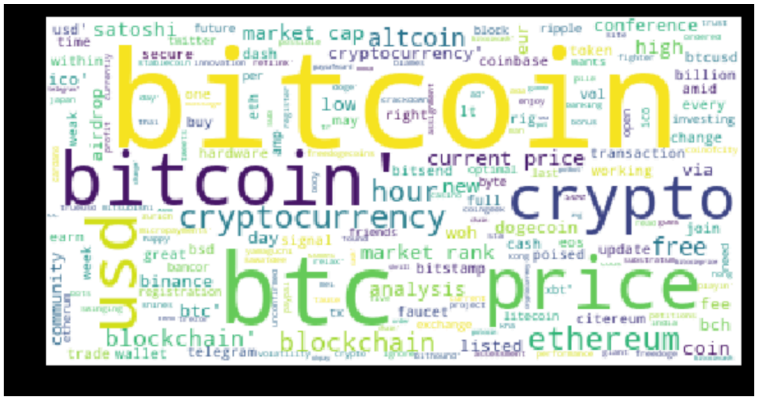

Figure 2: WordCloud Visualization to visualize the 80 most commonly used words

c. Labeling the training data using Emoticon based approach:

I assigned a label 0 for negative tweets and label 1 for positive tweets.

List of emoticons that I used for collecting the positive and negative data:

Positive	Negative
:)	: (
:-)	:- (
: D	:/

Table 3: Emoticon List

After collecting the tweets, I followed the cleaning of tweets where I removed stop words, punctuation marks and hexadecimal characters.

d. Training the data:

I used two algorithms for training:

1. Naïve Bayes
2. MaxEntropy

i. Naïve Bayes:

Creating the training data:
Forming bag of words:
The bag words are the frequency of occurrences of a particular word in the entire document of the training set. For creating the bag of words, I utilized Scikit learn's CountVectorizer to create a sparse frequency matrix. The count_vector.transform(data) function creates the matrix both for training and testing data. The result is as follows:

access	accessible	accommodation	accomodation	accomplish
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Figure 3: Sparse frequency matrix

In order to make the three steps of vectorizing, transforming and training easier, I used Scikit learn's Pipeline feature.
I created a Pipeline using CountVectorizer, TFIDFTransformer and Multinomial Naïve Bayes:

```
text_clf = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])
```

Sample Training data:

```
['billboards outside warren buffetts office genesismining bitcoin bi
tcoinawareness',
'much voting helped win two awards last night influencer year amp re
latable nice',
'friday autobayers sta making plans weekend friday fridayfeeling fri
daymotivation ico',
'registration open preregistration hurry',
'twitter bitcoin cryptocurrency etherum blockchain ico',
'call crypto king proud pakistan may allah bless brother aameen cryp
tocurrency bitcoin co',
'ico scam education siteprepared sec know crypto infrastructure gogr
een sesdeepstate ped']
```

Training the data
For training purposes, in order to avoid overfitting, I used Stratified K-fold cross validation. One of the challenges in my data, was that there were more positive tweets than negatives which formed a bias in my data. This is because I considered Neutral tweets as positive. People are excited about bitcoin and rarely tweet something negative about it. But even after performing stratified k-fold, I couldn't remove the bias.

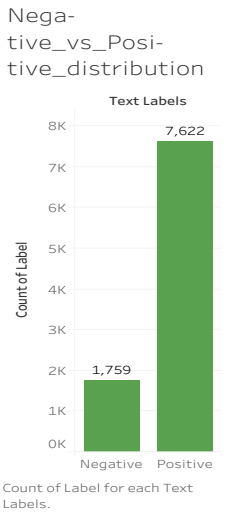

Figure 4: Distribution of negative and positive tweets

One way I followed is undersampling the training data by taking 2000 positive tweets out of 7622 tweets. I have discussed the results in the analysis section.

Testing the data:

For testing purposes, I extracted a set of 255 Tweets and manually assigned positive and negative label to those tweets. I used that set of data for testing my algorithm.

ii. Maximum Entropy:

Maxent classifier consists of feature based models. It is very similar to logistic regression and does not assume independence of features like Naïve Bayes. This allows use of bigrams as features, without worrying about repeated features.

Creating the training data:

For data to be fitted to the MaxEnt classifier, the data set needs to be converted to a tuple of the form:

```
training_set = ['bleeding': True,
  'country': True,
  'debt': True,
  'getting': True,
  'late': True,
  'red': True,
  'redfored': True,
  'thing': True,
  'worse': True},
  'Negative'),]
```

I used the following method to convert it into the form:

```
def list_to_dict(words_list):
      return dict([(word, True) for word in words_list])
```

Training the data:
During the training with MaxEnt I followed the same approach of undersampling as I followed in Naïve Bayes. I used the Iterative Scaling (IIS) algorithm of NLTK. Using this algorithm, as we increase the number of iterations, the accuracy increases. I used 50 iterations.

Testing the data:

For testing the data, I used the same data set as I used for testing the Naïve Bayes approach.

6. Analysis of Results and Evaluation Strategy:

I evaluated the results by observing the accuracy, f1-score, Precision and Recall

Multinomial Naïve Bayes with 2000 positive and 1759 negative tweets with 10-fold cross validation

F1	0.75
Precision	0.65
Recall	0.874
Accuracy	0.611

Table 4: Multinomial Naïve Bayes with 2000 positive and 1759 negative tweets with 10-fold cross validation

Multinomial Naïve Bayes with 2000 positive and 1759 negative tweets with 5-fold cross validation

F1	0.74
Precision	0.65
Recall	0.86
Accuracy	0.6

Table 5: Multinomial Naïve Bayes with 2000 positive and 1759 negative tweets with 5-fold cross validation

Multinomial Naïve Bayes with 7622 positive and 1759 negative tweets with 10-fold cross validation

F1	0.93
Precision	0.99
Recall	0.877
Accuracy	0.87

Table 6: Multinomial Naïve Bayes with 7622 positive and 1759 negative tweets with 10-fold cross validation

Multinomial Naïve Bayes with 7622 pos and 1759 negative tweets with 5-fold cross validation

F1	0.92
Precision	0.98
Recall	0.87
Accuracy	0.86

Table 7: Multinomial Naïve Bayes with 7622 positive and 1759 negative tweets with 5-fold cross validation

MaxEnt with 2000 pos and 1759 negative tweets with 10-fold cross validation

F1	0.84
Precision	0.87
Recall	0.82
Accuracy	0.725

Table 8: MaxEnt with 2000 positive and 1759 negative tweets with 10-fold cross validation

MaxEnt with 2000 pos and 1759 negative tweets with 5-fold cross validation

F1	0.83
Precision	0.87
Recall	0.79
Accuracy	0.71

Table 9: MaxEnt with 2000 positive and 1759 negative tweets with 5-fold cross validation

MaxEnt with 7622 pos and 1759 negative tweets with 10-fold cross validation

F1	0.92
Precision	0.96
Recall	0.87
Accuracy	0.847

Table 10: MaxEnt with 7622 positive and 1759 negative tweets with 10-fold cross validation

MaxEnt with 7622 pos and 1759 negative tweets with 5-fold cross validation

F1	0.91
Precision	0.87
Recall	0.87
Accuracy	0.843

Table 11: MaxEnt with 2000 positive and 1759 negative tweets with 10-fold cross validation

As we can see, by undersampling the data, the scores. Another reason for getting a good score can also be due to the bias in the data. A 10-fold cross validation yielded better results than 5-fold. Also, I got better results using Naïve Bayes compared to MaxEnt classifier. It can be seen that people tweet more positively about Bitcoins and most of the tweets are promotion based and neutral.

7. Comparison of sentiments with Bitcoins:

Due to the limitation of memory resources of my machine, I displayed the change in bitcoin prices along with the sentiment from 08th May to 20th of May, 2018. I extracted Twitter data of the same period without Emoticons and predicted their sentiments using MaxEnt classifier. I visualized the trend of bitcoin price change along with sentiment change. Initially, I noticed there was no correlation. Although we can see that the number of tweets dropped as the prices dropped, but there was no relation with the sentiment of the tweet. Therefore, I shifted the prices by 1, 2 and 3 hours just to see if the sentiments dropped in some hour, will the Bitcoin prices drop in the next 1 hour, 2 hours or 3 hours. That is, I wanted to check that if there is a drop in sentiments at 5pm, then will there be a drop in the price at 6 pm, 7 pm or 8 pm. But again, I didn't find any correlation. The results are shown below:

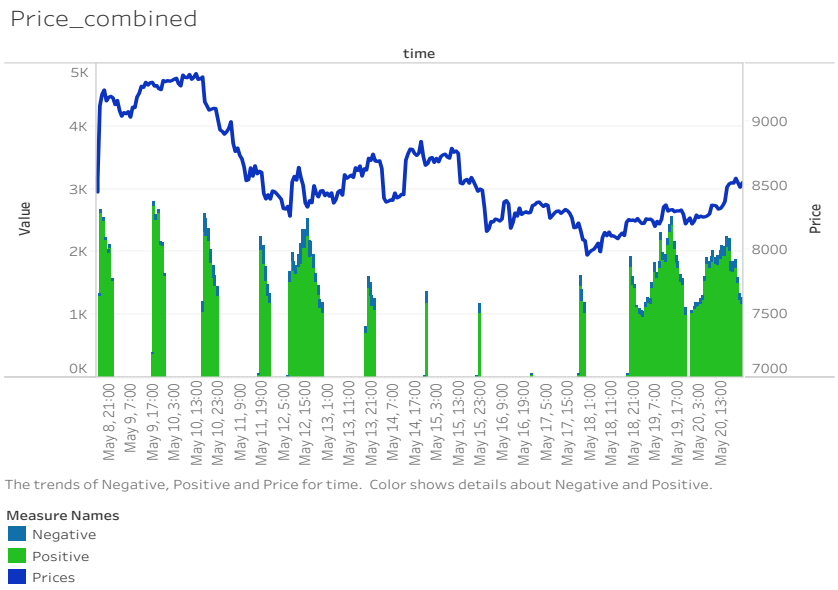

Figure 5: Comparison of bitcoin price trend with the sentiments

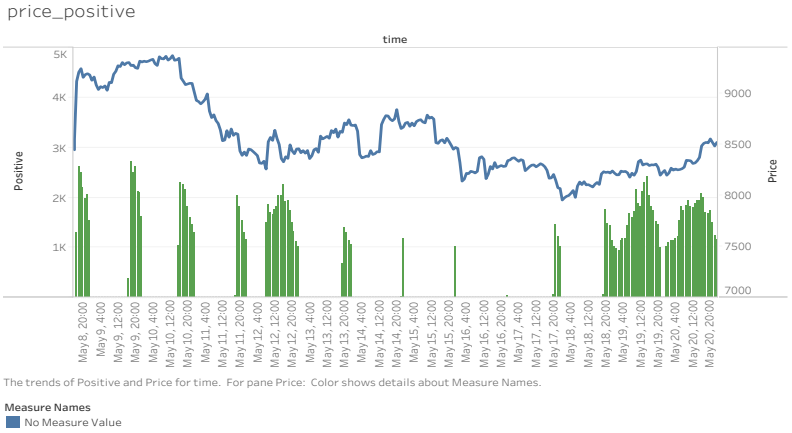

Figure 6: Comparison of bitcoin price trend with the positive sentiment

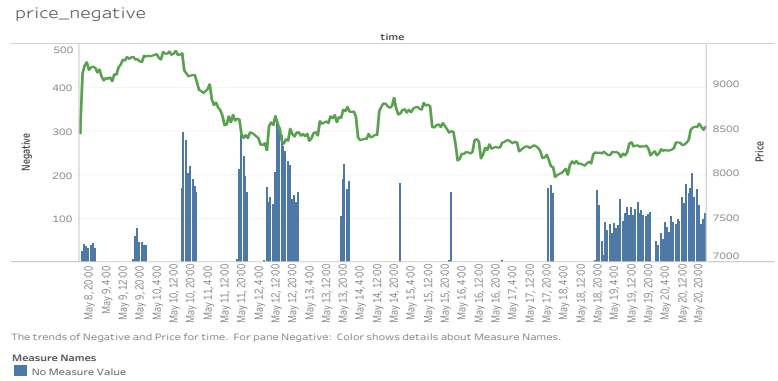

Figure 7: Comparison of bitcoin price trend with the negative sentiment

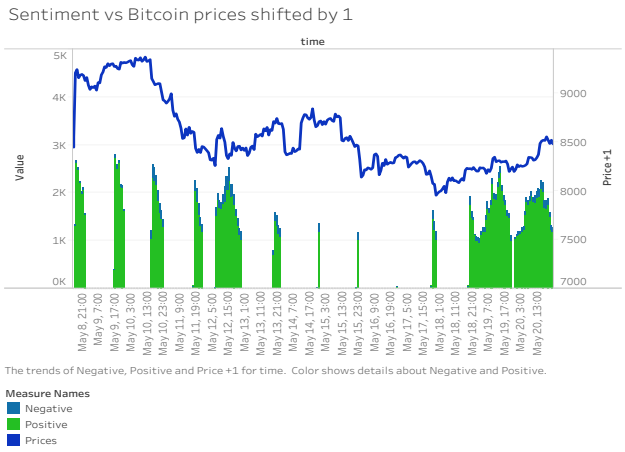

Figure 8: Comparison of sentiments with Bitcoin prices shifted by 1 hour

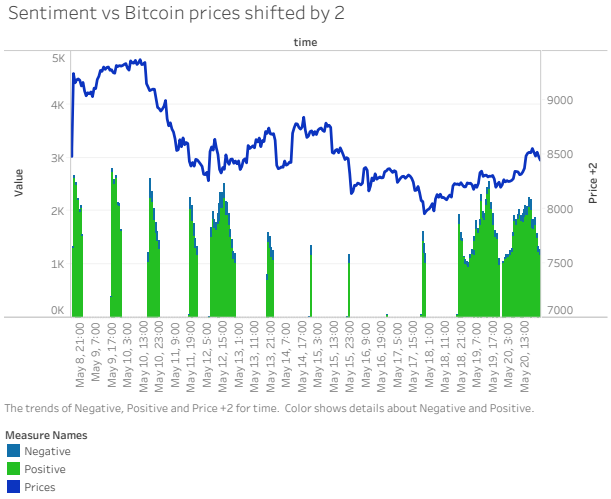

Figure 9: Comparison of sentiments with Bitcoin prices shifted by 2 hours

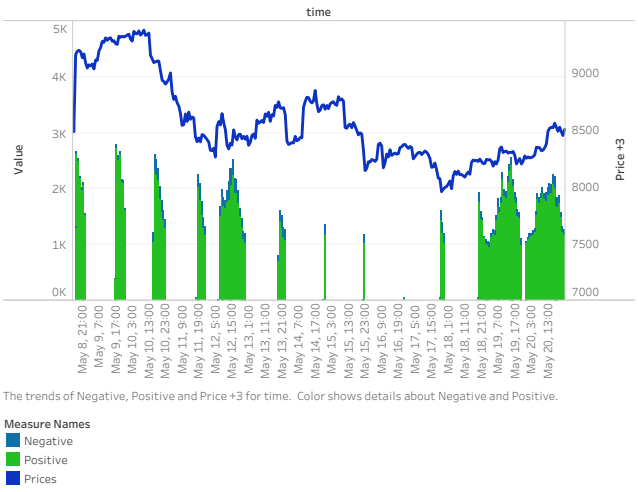

Figure 10: Comparison of sentiments with Bitcoin prices shifted by 2 hours

8. Pros and Cons of my strategy:

Pros:

1. My approach is not rule based and lexicographic, and therefore, it provides a better result compared to lexicographic tools like TextBlob and VADER.
2. Anyone interested in knowing the opinions of the masses about Bitcoins, can view the results and get an idea about the perception of people on the topic.
3. By utilizing two Machine learning algorithms to train the data with stratified cross validation, the data is not prone to overfitting. Getting fairly high scores on test data reflect the correctness of the model.

Cons:

1. Since, I have not been able to utilize a huge amount of data due to limited resources, the results are not satisfactory.
2. The data is classified using an emoticon based approach. However, not all emoticons reflect the sentiment of a particular tweet. People may use a ☺ emoticon for a negative tweet.
3. The current approach doesn't handle neutral tweets.

9. Future direction:

In the future, I would try to utilize more data and observe the correlation in detail by minutes rather than hours. Not every tweet about bitcoin is significant. So, extracting data by filtering users who have more than five followers can be an approach that will lead to the improvement of the current approach. Also, the current approach calculates the sentiment of a tweet irrespective of context. For example, a tweet like, "This is the last month when Bitcoin price will be less than $10k", can be a positive or negative tweet according to the context based on whether a person is trying to invest on Bitcoin. In such cases, if the semantics of the tweet can be extracted that will help in classifying with a context based method. I haven't considered bigrams as a feature. As a future work, I would like to use bigrams as a feature while training.

10. Conclusion:

The project focused on retrieving the sentiments of the tweets related to Bitcoin. I used the emoticon based approach to retrieve the positive and negative tweets using Tweepy. I predicted the sentiment of tweets collected over a period of 10 days and visualized the Bitcoin price change with sentiment. However, I noticed no correlation between the sentiment and Bitcoin price change but I did notice some correlation between the number of tweets and bitcoin price change. I shifted the bitcoin price by 1, 2 and 3 hours but still didn't notice much correlation. I used two machine learning algorithms for the prediction and analyzed the results.

References:

1. Debjyoti Paul, Feifei Li, Murali Krishna Teja, Xin Yu, Richie Frost, "Compass: Spatio Temporal Sentiment Analysis of US ElectionWhat Twitter Says!"
2. Alec Go, Richa Bhayani, and Lei Huang. 2009, "Twitter sentiment classification using distant supervision. CS224N Project, Stanford 1, 12 (2009)"
3. http://blog.chapagain.com.np/machine-learning-sentiment-analysis-text-classification-using-python-nltk/
4. http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
5. Evita Stenqvist Jacob Lönnö, "Predicting Bitcoin price fluctuation with Twitter sentiment analysis"