Bitcoin Price Forecasting via Twitter Sentiment Analysis

by James Peters-Gill[1]
University of Oxford

***Abstract*— This short paper explores the application of supervised Machine Learning algorithms in the forecasting of cryptocurrency price. Whilst the methodology is versatile and could be applied to a wide range of alternative currencies and financial derivatives, the currency considered in this project is Bitcoin (BTC) due its prominence at the time of publication (Summer 2018).**

I. INTRODUCTION

Cryptocurrency differs from traditional fiat currency in that it uses blockchain technology to regulate the generation of its units and verify the transfer of funds, thus operating independently of central banks.[1]

A blockchain is a digital ledger which is decentralised and available to the public. It is organised into a sequence of blocks where each block consists of a list of transactions together with a unique number at its footer called a *proof of work*. Each listed transaction is deemed valid only if the sender has authenticated it by means of a digital signature. Likewise, the entire block's validity is dependent on it having a correct *proof of work*.

A *proof of work* is a number which requires extensive computing power to obtain, yet can be validated with ease. Blockchain technology uses cryptographic hash functions (such as *SHA256*) as its mechanism for *proof of work*.[2] These are functions which are infeasible to compute in the reverse direction. The requirement for a block to be authenticated is that its hash begins with a specified number of zeros and thus a suitable number must be placed into the footer to achieve this. Obtaining such a number is impossible without extensive *trial and error* and consequently requires considerable computational work.

Anyone in the world can be a block creator. It involves listening for transactions being broadcast, collecting them into a block and performing computational work in a bid to find a successful *proof of work*. To reward a block creator for their labour, they may include a transaction in the ledger in which they receive a specified quantity of currency known as the block reward. As a result, the aggregate size of the digital economy increases with each block.

The number of zeros necessary for an accepted hash is periodically calibrated so as to maintain a constant rate of block production despite the increasing population of miners and access to computing power. The block reward is also halved periodically. As a result, the size of the crypto coin economy may be modelled as a geometric series and hence is naturally limited in size.

Though there are now many thousands of them, *Bitcoin* was the first implemented example of such a currency. Since its inception in January 2009, numerous techniques have been utilised to forecast the value of the coin with varying success. It seems feasible that social media would provide a valuable insight into current demand for the crypto coin and thus provide a useful tool in forecasting its market price.

As of 2018, Twitter was one of the world's most popular social networking platforms with 330 million active monthly users and 500 million tweets sent per day.[3] It is therefore unsurprising that tweets can be used to analyse public sentiment with remarkable success, especially when investigating a topic so widely discussed online as cryptocurrency. Unlike regular market commodities, crypto coins have no intrinsic value and thus their valuation is determined almost entirely by public sentiment. This paper investigates the extent to which twitter data can be utilised to train supervised machine learning models to predict the market valuation of *bitcoin*.

II. DATA

A. Training the Sentiment Analysis Tool

A corpus of 100,000 tweets was used, each of which had been classified as having either *positive* or *negative* sentiment.

B. Training and Validating the BTC Model

BTC price data was obtained from *kaggle.com* and corresponding historical tweets were extracted from the *tweepy* API by searching with the keyword *bitcoin*. These data spanned a two month period from 13th March 2018 to 15th May 2018.

C. Making Predictions

The live streaming feature of the *tweepy* API was utilised to provide up-to-date tweets, again filtered by the keyword *bitcoin*, from which my trained model could make predictions.

III. METHODOLOGY

The investigation comprised of two key phases. Phase one constituted the use of supervised machine learning techniques to build a sentiment analysis tool for classifying tweets. Phase two involved the application of machine learning to predicting the *bitcoin* price using twitter sentiment as the principal feature. All learning algorithms were

*This work was supported by Trinity College, Oxford.

[1]J. Peters-Gill is a member of the University of Oxford Mathematical Institute, Andrey Wiles Building, Radcliffe Observatory Quarter, Woodstock Road, Oxford james.peters-gill@trinity.ox.ac.uk
Supervised by Andrey Kormilitzin, University of Oxford.

written in *Python 3.6* using the *JupyterLab*[4] computational environment. The structure of my project can be summarised in the following flow chart:

A. Phase One

The aim of this phase was to build a function for which you may input a string of text, namely a tweet, and its sentiment is interpreted and output in the form of a numerical value where 0 is the most *negative* and 1 is the most *positive*.

$$f(text) \in [0, 1]$$

To achieve this, it was necessary to firstly implement some pre-processing techniques. All text was converted into lowercase and tokenised into separate words using the *NLTK*[5] module's built-in tokeniser. The text was then filtered by removing words which were members of the *stop-words* corpus - these are words considered to have no effect on the sentiment of the text. Lemmatisation was then applied, reducing each word to its stem, and any remaining words which were not members of *NLTK*'s *words* corpus were subsequently removed. Finally, the occurrence frequency of each word was counted and the text was reduced to a feature vector where each entry depicted the respective occurence frequency of each word.

Consider the following piece of text.

$$'\#bitcoin\ is\ looking\ GREAT\ right\ now -$$
$$Buy!\ Buy!\ Buy!'$$

The pre-processing steps described above reduce this tweet to the following feature vector:

$$\mathbf{x} = (x_1, ..., x_n) = (0, ..., 1, ..., 1, ..., 1, ..., 1, ..., 3, ..., 0)$$

where the non-zero entries represent the words *look*, *great*, *right*, *now* and *buy* respectively.

- **Naive Bayes**[6] classification is a simple probabilistic classification algorithm. It is a direct application of Bayes' theorem, assuming full independence of the features. We begin with each tweet represented as a feature vector in the form described above. We then consider the sample space $\Omega = \{C_1, C_2\}$ where the C_i are the events that the tweet has a positive or negative sentiment respectively. We may apply Bayes' theorem to the general outcome, C_i:

$$P(C_i \mid \mathbf{x}) = \frac{P(C_i)P(\mathbf{x} \mid C_i)}{P(\mathbf{x})} \propto P(C_i)P(\mathbf{x} \mid C_i)$$
$$= P(C_i) \prod_{j=1}^{n} P(x_j \mid C_i)$$

  The final line uses our *naive* assumption that the features are independent. The model is trained using the classified tweet data to calculate each of the $P(x_j \mid C_i)$. We then make class predictions by calculating $P(C_1 \mid \mathbf{x})$, the probability that the tweet has a positive sentiment given the entries in its feature vector and our naive assumptions. The tweet is then deemed positive if this probability is greater than $1/2$ and negative if not.

- **Logistic Regression**[7] is one of the most commonly used algorithms for binary classification. It exploits a linear equation with weight parameters θ_i to generate a value using the feature vector, $\mathbf{x}$:

$$z(\mathbf{x}) = z((x_1, ..., x_n)) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$$

  The resulting value is then inserted into the *sigmoid* function below so as the output variable lies in the interval $(0, 1)$:

$$h_\theta(\mathbf{x}) = \frac{1}{1 + e^{-z(\mathbf{x})}}$$

  The algorithm attempts to find maximum likelihood estimators for the parameters θ_i by maximising the following likelihood function in terms of the feature vectors, $\mathbf{x}^{(j)}$, and their corresponding binary outcomes in the training set, $y^{(j)}$.

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

  Maximising this likelihood function is equivalent to minimising the following logarithmic *cost* function

$$l(\theta) = \sum_{j=1}^{m} [y^{(j)} log(h_\theta(\mathbf{x}^{(j)})) + (1 - y^{(j)}) log(1 - h_\theta(\mathbf{x}^{(j)}))]$$

  The *cost* can be minimised in the standard fashion via taking partial derivatives with respect to each parameter θ_i and employing the stochastic gradient descent method. Once the parameters have been tuned, a tweet is then classified by applying the function to its feature vector $h_\theta(\mathbf{x})$ and deeming it positive if $h_\theta(\mathbf{x}) > 1/2$ and negative if $h_\theta(\mathbf{x}) < 1/2$.

n.b. this section began by stating the aim of producing sentiment scores in the range [0,1] and yet the above algorithms classify text into binary classes. To ensure the sentiment scores provided a valuable feature in phase two of the investigation, the probability of positive classification was used as the output of my text analysis tool rather than the binary sentiment class.

B. Phase Two

This phase involved employing Random Forests and Gradient Boosting algorithms to develop a model for forecasting the daily change in the *bitcoin* price. The historical tweet data was loaded into a *DataFrame* and formatted chronologically. The sentiment polarity scores for each tweet were added in a

new column as well as subjectivity scores obtained using the *TextBlob*[8] module. The *DataFrame* was then compressed into rows indexed by day, taking the mean average of the sentiment polarity and subjectivity scores. The *bitcoin* price data was loaded and inserted abreast. Finally, aligning the data such that each row contained the percentage change in *bitcoin* price for the following 24 hours enabled the model to be trained so as to build an forecast for tomorrow's valuation based on the live *Twitter* feed.

- **Random Forests**[9] are a method for regression which operate by constructing multiple decision trees and outputting the mean prediction from the individual trees. The foremost benefit of using a random forest over a simple decision tree regressor is that it can correct for overfitting.
- **Gradient Boosting**[10] is a more sophisticated prediction model comprising an ensemble of weaker prediction models, namely decision trees. These models are implemented in a stage-wise fashion so as to iteratively reduce the errors in the predictions. The following flow chart indicates how this model works:

C. Model Validation

In both phases of learning, cross-validation was used to test the accuracy of the models on unseen data. The data were partitioned into a 70:30 split for training and testing respectively. *Mean Abolute Error* scores were calculated as well as percentage success rate scores for binary classifiers. The procedure was then repeated with multiple random partitions and averages were calculated.

D. Making Predictions

Once a successful application for a *Twitter* developer account had been made, I could access the *tweepy* API. To make predictions, tweets were streamed by searching with the keyword *bitcoin* and collated into a *DataFrame*. The sentiment analysis tools were then utilised to process the tweets and the resulting features were input into the *bitcoin* model(s). The resulting output was a forecasted percentage change in the market *BTC* valuation for the following 24 hour period.

IV. RESULTS

When cross-validating my sentiment analysis tool using unseen data, I achieved an overall success rate of 74.75% using the Naive Bayes model. The logistic regression model however, was less successful. The results can be interpreted best in the following table.

TABLE I
ERROR ANALYSIS FOR LOGISTIC REGRESSION MODEL

	Precision	Recall	F1-Score
Negative	0.49	0.31	0.38
Positive	0.60	0.76	0.67
Avg / Total	0.55	0.57	0.54

It is clear from the table that both the *precision* and *recall* were significantly poorer for the texts with negative sentiment. This resulted in a poor *F1-score* of 0.54, which reflects only slightly better performance than what you might expect from an untrained uniform binary classifier. A likely explanation for this is that the model has suffered overfitting. It appears to have a greater inclination to classify texts as positive, due to their making up a majority of the training data.

As a result, I opted to use the Naive Bayes model as the primary feature for making my *bitcoin* predictions. When testing the model on the example tweet above, *bitcoin is looking GREAT right now - Buy! Buy! Buy!*, the function outputs a probability of 0.872 that it lies in the *positive* class.

Once the text analysis tool was applied to historical tweet data, I was able to compare the average *Twitter* sentiment polarity with the *BTC* price graphically and obtained the following:

Upon cross-validating the trained *bitcoin* model against unseen testing data, I obtained a *Mean Absolute Error* of 2.28% when using the optimal *maximum leaf nodes* parameter of 5. To place this in context, the *Mean Absolute Error* I obtained when implementing the most basic of models which assumes that tomorrow's valuation is simply the same as todays, was 3.22%.

V. CONCLUSIONS

This *bitcoin* price model yielded significant errors in its forecasts. It did however prove successful in predicting the valuations with smaller errors than the baseline model which assumed no change. For this reason, I believe the model still provides a valuable insight into the crypto coin markets, though its crucial source of error is that it fails to consider the plethora of other variables at play. A natural extension to this work would be to combine my findings with another price forecasting tool in an attempt to improve the accuracy of the predictions.

ACKNOWLEDGEMENTS

This research was supervised by Dr. Andrey Kormilitzin, Affiliate Researcher at the Mathematical Institute, University of Oxford.

Thanks to Trinity College, University of Oxford, for supporting this research financially with their *Mitchell Grant*.

REFERENCES

[1] https://www.bitdegree.org/tutorials/what-is-cryptocurrency/ (Cryptocurrency Mechanism), Sections 5,6,7&8, September 17th 2018, Ray King.
[2] https://www.cryptocompare.com/coins/guides/how-does-a-hashing-algorithm-work/ (SHA 256 Cryptographic Hash Function), September 10th 2018, Antonio Madeira.
[3] https://www.omnicoreagency.com/twitter-statistics/ (Twitter by the numbers), September 18th 2018, Salman Aslam.
[4] JupyterLab (Next-generation web-based user interface for Project Jupyter. The Jupyter Trademark is registered with the U.S. Patent & Trademark Office. Revision 7ebeb1ff.
[5] Natural Language Toolkit (NLTK) (leading platform for building Python programs to work with human language data), Update for Python 3, NLTK Project.

[6] https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ (Naive Bayes Classification Algorithm), September 11th 2017, Sunil Ray.
[7] https://towardsdatascience.com/the-logistic-regression-algorithm-75fe48e21cfa (Logistic Regression Classification Algorithm), May 5th 2018, Niklas Donges.
[8] TextBlob Module (library for processing textual data), consistent API for part-of-speech tagging, noun phrase extraction, sentiment analysis.
[9] https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd (The Random Forest Algorithm), February 22nd 2018, Niklas Donges.
[10] https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d (Gradient Boosting), December 8th 2017, Prince Grover.