Tweetbit: Bitcoin Price Prediction Using Twitter

Teo Yong Quan, 1002828
Sara Leong, 1003116
Koe Jia Yee, 1003017

Abstract

We present a method for predicting daily and hourly Bitcoin prices direction change utilizing Twitter data and Google Trends data. Bitcoin is the largest cryptocurrency in terms of market capitalization. However, Bitcoin has experienced significant price swings on both daily and long term valuations. Twitter is a popular social media platform that has been used as a news source influencing cryptocurrency purchase decisions. Being able to quickly understand the impact of tweets and the public opinion on price direction can provide useful purchasing and selling insights to cryptocurrency users. Through our analysis, tweet sentiment and Google trends individually are insufficient as a predictor of price. Through a collection of data and Bitcoin price data, we used various binary classification models, and evaluated the quality of each model with a confusion matrix. We found that using Bi-Directional LSTM model produced the best results for Bitcoin hourly price direction change. The link to our codes can be found here: https://github.com/teoYQ/Bitcoin-Twitter-sentiment-analysis

1 Introduction

As of November 2019, many people consider Bitcoin as "digital gold". With the market capitalisation of cryptocurrency worth $135.90B USD [1] it is no wonder people consider it as 'gold'. Hence, many people are interested in dealing with Bitcoin. However, due to the volatility of Bitcoin's value,which can range from 0.81% to 16.11% [2], it is difficult to make an accurate prediction if it is the right time to purchase or to sell Bitcoin.

Out of the top five factors [3] that affect Bitcoin, we realise that Media Influence has a huge impact on the Bitcoin. With the use of social media, any news may go viral and Bitcoin's value will be impacted. Furthermore, many existing Bitcoin price predictors do not take into account 'real time' public opinion to predict prices. Hence, we want to predict if the value of bitcoin will increase or decrease in the subsequent hour and day based on 'real time' public opinion.

We approach this task by collecting Bitcoin Related Tweets with feature engineering, followed by conducting sentiment analysis on tweets. We determined the relationship between Tweet sentiment and price direction before using the average sentiment analysis scores to predict if the Bitcoin price direction. We have also included the use of Google Trends Data to see if we could gain any insight on Bitcoin searches during price direction changes. We applied various binary classification models, Logistic Regression, Random Forest, KNN and LSTM with various inputs and evaluated their outputs. Based on the outputs, we are able to conclude which model works best for price prediction using Tweet sentiment analysis.

2 Dataset

2.1 Data Collection & Selection

2.1.1 Tweets

Twitter is a minefield of information due to the volume and variety of users. This has resulted in Crypto Twitter being significantly influential on cryptocurrency trading decisions. Thus, we collected tweets to perform sentiment analysis on.

We used GetOldTweets3to query tweets day by day as Twitter's Official API has limited access to older tweets for free. GetOldTweets3 supports execution in the Command Line (CL) and we wrote a script to write these CL instructions for every day between 1st January 2019 to 30th October 2019. With the ampersand command, we were able to run over 300 queries at the same time in the background which greatly sped up the time spent collecting the tweets.

```
GetOldTweets3 --querysearch 'bitcoin' --since 2019-9-2 --until 2019-9-3
--maxtweets -1 --lang en --output 'clsearch92.csv' &
```

The query above used the 'bitcoin' as the keyword filter to select Bitcoin related tweets, the since and until option to provide the date range, en to specify only tweets in English and maxtweets -1 to collect every single tweet in the day. This gave us a total of **6,265,266 tweets** (See Fig 1) to work with.

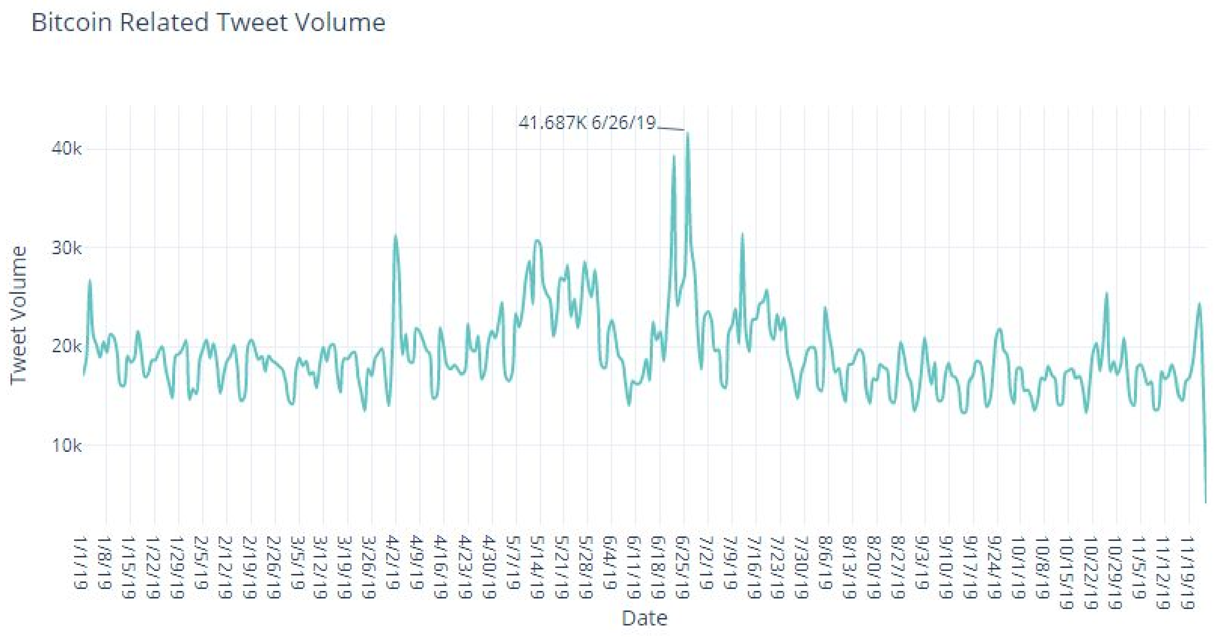


Fig 1: Tweet Volume of Bitcoin related Tweets from 1/1/19 to 19/11/19. Values retrieved from BitinfoCharts [4] The largest Volume occurs on 26/6/19 with 41.6897K. On this day, Bitocoin was reported to have hit a 17-Month high, rising above $12900 for the first time since 21/1/2018 [5] Possibly explaining this spike in Bitcoin related tweets.

Fig 2: wordcloud created from our tweets

2.1.2 Bitcoin Prices

We obtained the data on past prices from Coinbase as it is one of the top BTC exchanges, especially in the USA where the majority of BTC operational nodes and Twitter users are located. Data from : http://www.cryptodatadownload.com/data/northamerican/.

Bitcoin prices are known to be volatile and the market for cryptocurrency is still evolving. The main difference of BTC prices and usual stock prices is that it changes on a much greater scale than local currencies. Thus, we collected the **daily and hourly closing prices** to experiment the time range in which our model's features will affect a price change.

2.1.3 Google Trends

With Google being the ubiquitous search engine, up to 74.52% of all internet searches is done with Google. Search data from Google could provide insights in the interest of majority of the internet population. Google Trends is a tool and data store of popular searches and when they occur, found at https://trends.google.com/trends/explore?q=btc%20usd&geo=US.

Using Google Trends, we collected trend data in relation to interest in Bitcoin customising the region, time frame and keywords. We used global Bitcoin search trends from 1st Jan 2019 to 31st October 2019. Our selection of keywords boiled down to 'bitcoin' and 'BTC USD'.

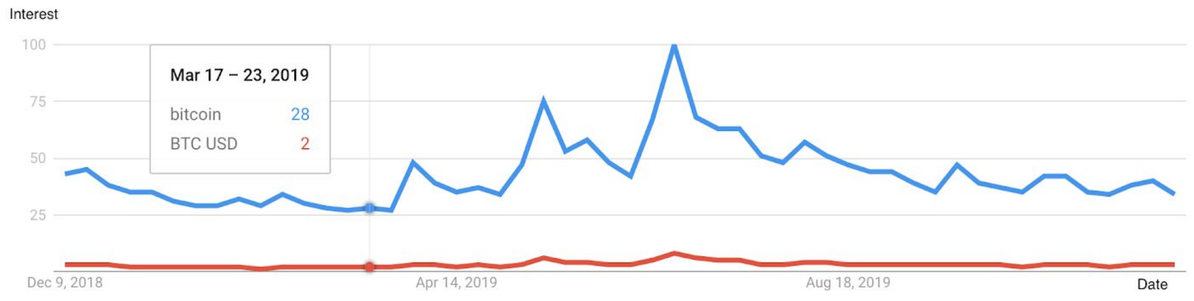


Fig 3: Google Keyword Search Interest against Time in 2019

Based on the graph (Fig 3), we realised that the term 'bitcoin' may be much noisier as it is a buzzword used by all people alike, experienced or inexperienced in the market. An active Bitcoin user is more likely to check the price periodically with the **keyword 'BTC USD'**. Thus, it is more reliable to use it as a proxy to the growth and engagement of active Bitcoin users over time[6].

2.2 Data Pre-Processing

2.2.1 Tweets

2.2.1.1 Cleaning tweets

After using our tweet scraper, we had to drop some columns that we deemed irrelevant for our analysis, this includes the tweet locations, tweet id, the list of mentions, hashtags and permalinks. We kept the following columns which we deemed useful, the date, username, number of replies, retweets, favorites and the text of tweet.

Before we are able to start doing any form of sentiment analysis, the tweets collected have to be cleaned.

Tweet Samples :

> "Who is the most underappreciated industry mover and shaker of bitcoin? pic.twitter.com/q3UdXfHoe2"
> "Bitcoin price index https://www.worldcoinindex.com/coin/bitcoin #USD #EUR #CNY #GBP #RUBpic.twitter.com/UDhg2Sbp8H"
> "Bitcoin Climbs Above 11,316.7 Level, Up 6% https://yhoo.it/2YV6wKZ #bitcoin #crypto #blockchain #btc #news #cryptocurrency pic.twitter.com/mPv3rd4kLn #eth #ltc #xrp"

These tweets contain large amount of noise, such as hashtags, URLs, pictures and some may even contain non English characters which our Sentiment Analyzer cannot process. Using regex expressions, all these noises were removed. Preprocessing is a very important aspect of sentiment analysis, if we were to pass these raw tweets to our analyzer, chances are it will perform very poorly and take up much more time.

After Cleaning:

> "Who is the most underappreciated industry mover and shaker of bitcoin "
> "Bitcoin price index"
> "Bitcoin Climbs Above Level Up"

After processing the tweets, we managed to cut down our csv file size by 33.3% which allows us to spend more time tweaking our model, while preserving the important parts of the tweets. This was enough for VADER (Valence Aware Dictionary for sEntiment Reasoning) as it was tuned for social media content. However, as we were also using Textblob, which unlike VADER is not trained to deal with social media content, more tweet processing were carried out.

Next, we set all the characters to lower cases and also removed stopwords in our tweets. Stop words are commonly used words such as "a","the","as" which provide no valuable information in Sentiment analysis. As mentioned earlier, we do not want to waste time processing invaluable words, so we made use of Natural Language Toolkit(NLTK) to get rid of them. After which, we performed stemming, again with the help of NLTK and used PorterStemmer, a process for removing the commoner morphological and inflexional endings from words in English. For example, the tweet "Who is the most underappreciated industry mover and shaker of bitcoin will look something like "who underappreci industri mover shaker bitcoin". Using pandas groupby functionality, we managed to obtain the Daily and hourly tweet counts as well.

2.2.1.2 Topic Modelling and LDA for Advertisement and Spam handling

In our first attempt on sentiment analysis (see Preliminary Model), our sentiment analysis did not yield good results as it gave advertisement tweets a high positive sentiment score eg, "Free bitcoin faucet with bonuses Get free bitcoins now" : +0.8807 (Using Vader)

This is an issue as our prediction model was unable to differentiate between a useful tweet and ads. Hence,we identified and tagged these ads tweet by using Latent Dirichlet Allocation (LDA) to cluster and identify possible tweet topics. This is useful as it allows us to gain insight on the type of Bitcoin topics people discuss on Twitter and within these topics, to identify an ad topic cluster. The reason for choosing LDA it is most effective in generalizing to new documents easily. Furthermore, it allows us to use pyLDAvis [7], to create interactive visualisations. (Interactive visualisations can be found on our Github).

To train our LDA model, the following parameters were varied:

1. Corpus: Bag of Words generated from already pre-processed tweets from April 2019
2. id2word: Dictionary mapping of word ID to words in tweets
3. Number of topics: The number of topic clusters to be generated.

The model parameters were varied by using different number of tweets from April 2019 (1000, 10000, 100000, All = 594018) and number of topic clusters. Each tweet is represented as a distribution over topics and each topic is represented as a distribution over words. To evaluate our model parameters we used the following:

1. Visual analysis of clusters with pyLDAvis
Through the visual analysis of 10, 6 ,3 ,20 topic clusters, we found that 20 topic clusters gave the best result. In the case of 3 topic clusters where none of the clusters overlap in the Intertopic Distance Map in Fig 4 ,indicating that there are 3 clear distinct topics, the topics were too broad and vague to use them to identify an ad cluster. The histogram on the right represents the most probable topic word for that cluster of tweets. Hence, we increased the cluster size until we found that our clusters gave us unique most probable topic clusters.

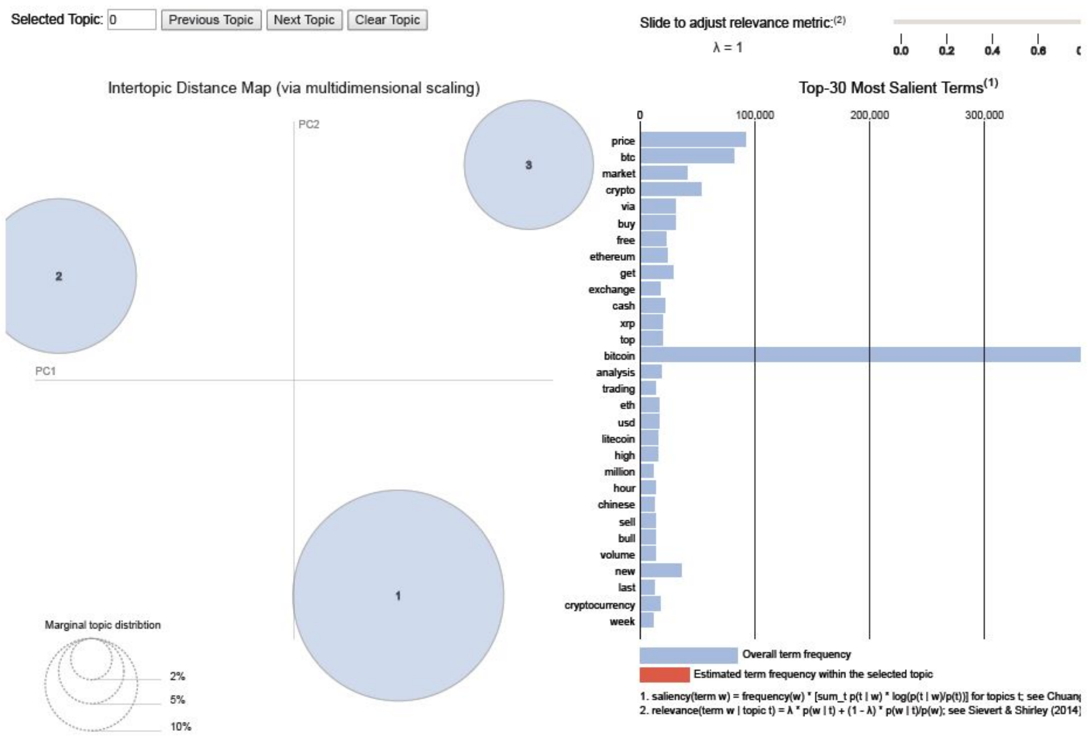
As seen here in Fig 5, we can see that the various possible Bitcoin topics are: Bitcoin, market, price, btc, mining, analyst, new, day, transaction, news, time, analysis, blockchain, dont, eth, payment, exchange, million, social media, resistance. Based on the cluster topics, we suspected that the unusual 'don't' cluster(Fig 6) (cluster 14 on pyLDAvis, topic 13) is likely to be the ad cluster.

2. Running new unseen documents through our model
Advertisement Tweets Examples

1. 'is airdropping den to participants sign up here'
2. 'claim daily bitcoins at for free'
3. 'bitcoin black v bitcoin get free coins value '
4. 'get free bitcoin'
5. 'claim free bitcoin now link moon bitcoin is a bitcoin faucet with a difference you decide how often to claim april at pm'

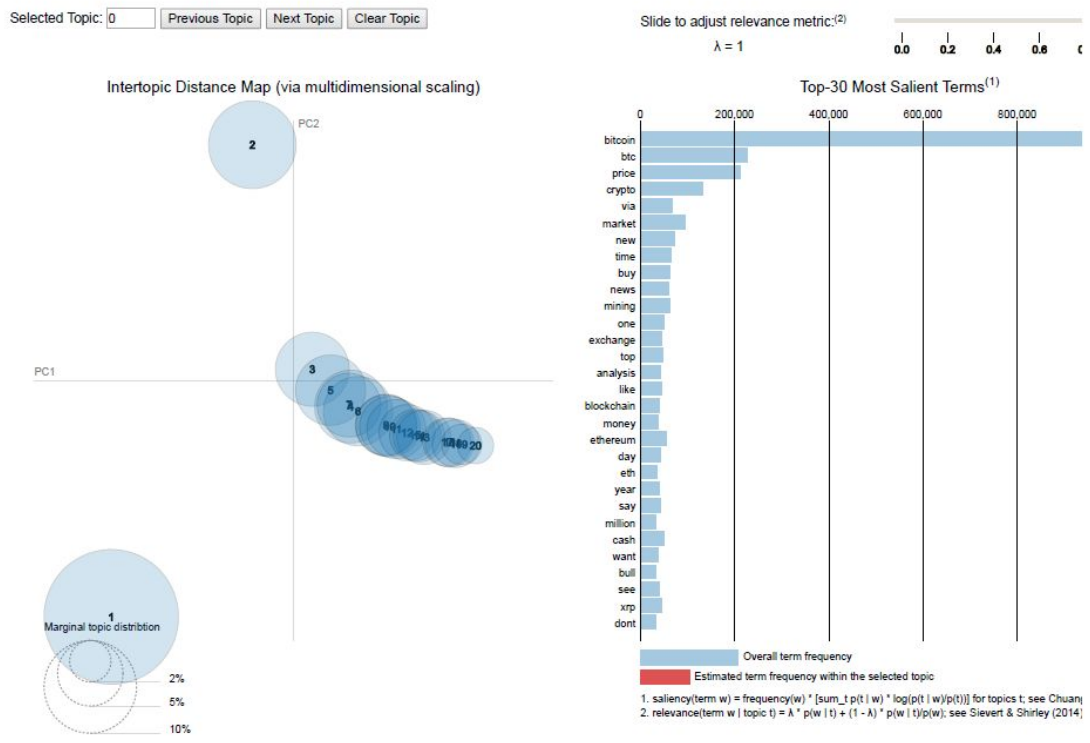
To confirm our hypothesis, we used tweets that we know are ads (above) and ran it through the LDA model. We wanted to find the model parameters that will result in all 5 tweets' LDA results returning topic 13 is the most probable topic . Hence, we are able to establish that topic 13 is our ad cluster and our ideal model parameters are to use all the tweets with 20 topic clusters.

To tag the ad tweets, we used all tweets from 2019 and ran it through our LDA model, clean4model_full_20 ,and filtered out all Topic 13 tokenized tweets and extracted the words to form an "ad word list" which is used in the preprocessing stage to tag the ad tweets.



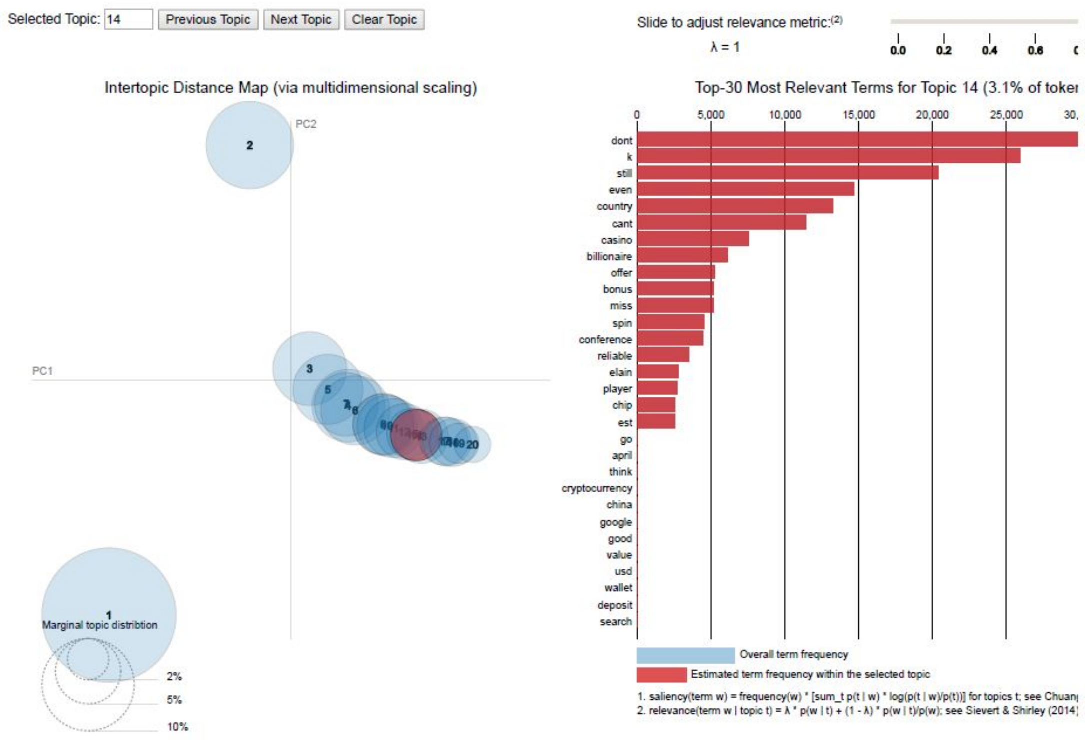
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig 4: pyLDAvis interactive visualisation of Topics generated from LDA. Model parameters: # of tweets used =All, # of clusters = 3. On the left, is a PCA Plot, where the size of area of the circle refers to the importance of each topic over the entire corpus, distance refers to the similarity. On the right, the histogram shows the top 30 most probable topic term. To view this interactive graph, click and download LDA_full_3.html.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuan
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig 5: pyLDAvis interactive visualisation of Topics generated from LDA. Model parameters: # of tweets used =All, # of clusters = 20. To view this interactive graph, click and download LDA_full_20.html.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuan
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig 6: pyLDAvis for LDA_full_20.html for suspected ad topic cluster

2.2.2 Bitcoin Prices

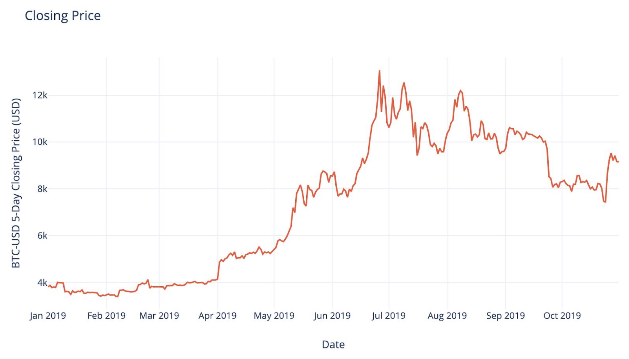
The price data also required some form of preprocessing.

1. Calculated the **short-term moving average** of the price to attempt to encode more information at each calculation:

$$\textit{Simple Moving Average} = (nA_1 + A_2 + \ldots + A_n) / n$$

where A=average in period n, n=number of time periods

This cuts down the noise and smoothens the price chart giving better insight to the basic price trend [8]. We chose the timeframe of 5 days and 5 hours for daily and hourly price data respectively due to the volatility of BTC prices and general short-term effect of social media influence.



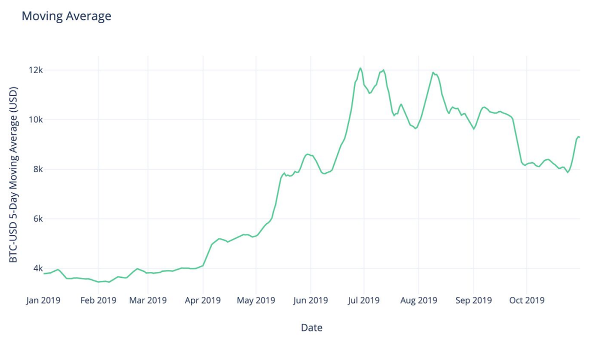


Fig 7 & 8: Closing Price vs Moving Average Price Chart

2. Calculated the **price direction** between each timestep (daily and hourly) and created a column that gives the binary classification based on the price difference.
3. As the API for collecting bitcoin prices were different from Twitter's data, we had to convert align the dates and time accordingly to the date and hour of the tweets.

2.2.3 Sentiment Analysis

Our goal is to apply sentiment analysis to our collected tweets to measure the subjective emotion or opinion of these tweets regarding Bitcoin.

Due to our Tweet data set being unlabelled, we attempted various ways of labelling the data such that we could use document vectors or TF-IDF models, which requires labelled data) for sentiment analysis. As our dataset required for training the sentiment analysis model is very large, it is beyond our project time and manpower to manually label the sentiment of each tweet. We explored using the Hashtags (see Fig 9) and Topic Modelling (see section above) to help automate data labelling. However, we realise that the majority of Hashtags used do not give any useful information about the sentiment of tweets. As for using Topic Modelling, the topics generated do not always correlate with the sentiment of the tweets within the cluster. Hence, we are unable to assign sentiment to tweets based on the topic cluster they fall under.

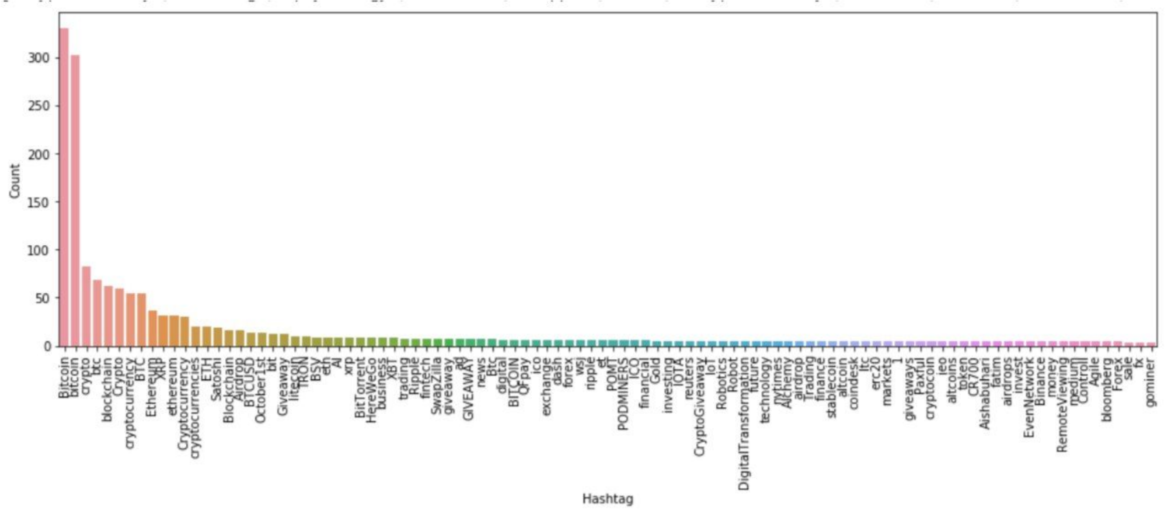


Fig 9: Count of top 100 Hashtags used in Bitcoin related tweets for 2017. Hashtag terms alone are neutral in nature.

As we were unable to get accurately labelled data for our tweets, we chose to use Lexicon-based approaches. In particular, we used VADER (Valence Aware Dictionary and Sentiment Reasoner) [9] and TextBlob for analysis [10]. It is worth noting that VADER's sentiment analysis tool is specifically attuned to social media sentiments, which may produce more promising results as our data is scraped from a social media platform.

TexBlob's sentiment function returns a polarity score which lies in the range of [-1, 1], where 1 refers to positive sentiment.

VADER's sentiment function returns a compound polarity score. The score that lie in the range of [0, 1] which we have scaled to [-1 , 1] to match TextBlob's range.

Based on observation, we selected a few tweets to compare if VADER or TextBlob performs better. We noticed that the sentiment score for advertisements and spam tweets were mostly quite positive. Hence, we should filter out such tweets which has been described in detail in the Topic Handling section.

The performance of VADER and TextBlob sentiment analysis will be discussed in further detail in the Models section. After the sentiment of each Tweet is computed, we take the average sentiment for daily and hourly. However, simply taking the average sentiment results in the loss of information. For example, a prominent user's positive tweet (eg, score of +1.0) may have greater influence over public opinion as compared a common users' negative tweet (-1.0). In this case, the simple average sentiment is neutral (0.0), yet the effect on public opinion should be positive. To address this issue, we decided to use weighted Sentiment analysis scores.

2.2.3 Feature Engineering and Selection

Our collection of features consists of Tweets text, Time of Tweet, Number of Retweets, Number of Replies, Number of Favourites, Advertisement score, Google Trends ('BTC USD'), Tweet Volume, Sentiment Analysis Score (VADER), Sentiment Analysis Score (TextBlob) and Tweet direction. To determine which features we want to use for our models, we did the following:

1. Feature Engineering to created Weighted Sentiment Analysis score
2. Analyse the relationship between remaining Features and price/price direction.

As we have used the Tweets to generate the sentiment analysis score, we can exclude this feature.

2.2.3.1 Weighted Sentiment Score

To prevent the loss of information by taking the average sentiment, we have generated weights to create weighted sentiment scores. Each tweet will have a weight score.

These are the Weight Rules we have established:

	Rule	**Weight Score**
1.	Prominent User	If the Tweet is made by user on the list, +1 to the weight score
	We generated a list of prominent Cryptocurrency Twitter users. Just to name a few accounts, @iamjosephyoung, @BitcoinMagazine, @BTCNewsletter, @coindesk ,@MoonOverlord This list of Twitter users were found through recommendations online and friends who actively follow Bitcoin news. [11]	
2.	Bitcoin Keyword List	Add weight +1 if at least one or more keywords are used.
	We generated a list of keywords that are used in Bitcoin. If any of the words are used, it is likely that the tweet contains useful information.	
3.	Number of Retweets, Replies and Favourites	Added the round(log(no of retweets + replies + favourites) to the weight score. As some tweets may have thousands of retweets

	The number of retweets, replies and favourites are indicators of impact and reach. The larger the number, the greater the weight.	while some have none, we took log to normalise the data.
4.	Time As our price data is based on the closing price, the tweets made nearer to closing price are more reflective of the current public opinion during the closing price. Hence, we used a naïve method of splitting the time period into half. Tweets made in the later half carry more weight.	For daily, if the tweet is made from 13:00 to 23:59 and for hourly, if the tweet is made in the last 30 mins, +1 is added to the weights.
5.	Advertisement Score As mentioned previously, we needed a way to filter out the ad tweets. In the data pre-processing stage, based on the LDA ad cluster's results, we are able to tag each tweet with an ad_score.	The Advertisement Score weight is the ad_score obtained during data pre-processing.

The final weights equation is as follows:

Weight *= [weight(prominent user) + weight(keyword) + weight(# of retweets) + weight(# of replies) + weight(# of favourites) + weight(time)] * ad_score*

Weighted Sentiment per tweet *= sentiment score of tweet * Weight*

Weighted Sentiment per day/hour *= sum(Weighted Sentiment per tweet)/ # of tweets in time period*

To address the issue with VADER and TextBlob assigning ads high polarity scores, we multiply the ad_score to the other weights. As ad_score for ads are assigned as 0, resultant weight is 0, which in turn, Weighted Sentiment will be 0. As the weighted sentiment of ads are now neutral, we have effectively filtered out the ads as it will not affect our model prediction.

We have generated 2 new features, weighted sentiment score (VADER) and weighted sentiment score (TextBlob). Hence, we can exclude the features we used to generate these new features. Based on Fig 10, we can intuitively guess how the two weighted sentiment analysis will perform. They follow the same general trend, however, VADER appears to be more sensitive as seen from the period of 5/8/19 to 5/26/19, where the change in sentiment (from very negative to very positive) is distinctively amplified. Hence,the results from VADER might perform better than TextBlob.

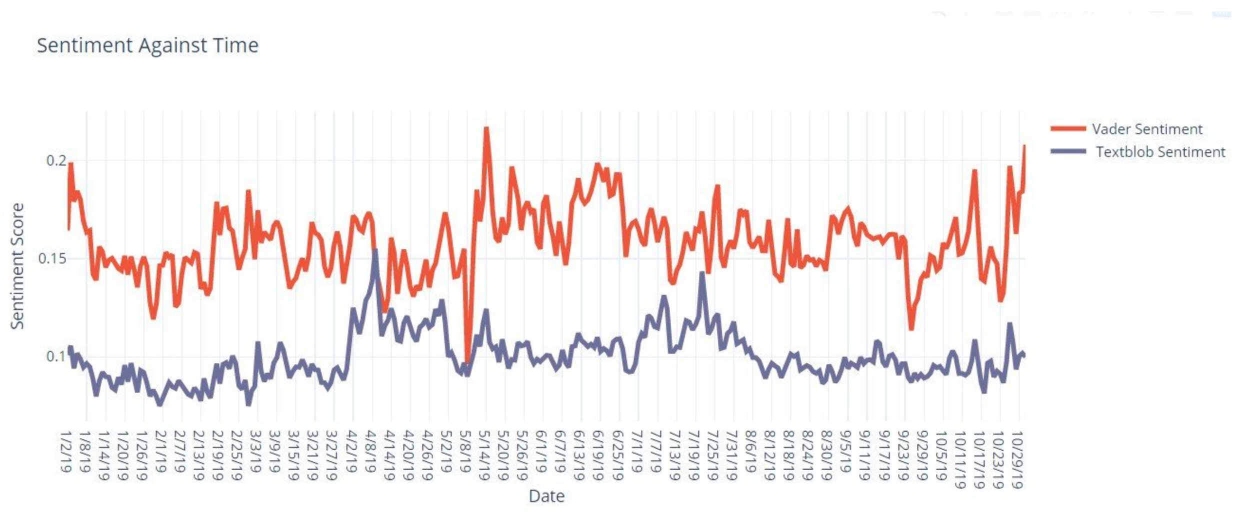


Fig 10: Daily sentiment Score after scaling of data (VADER and TextBlob) from 1/1/2019 to 30/10/29 .

2.2.3.2 Tweet Volume and Movement Direction

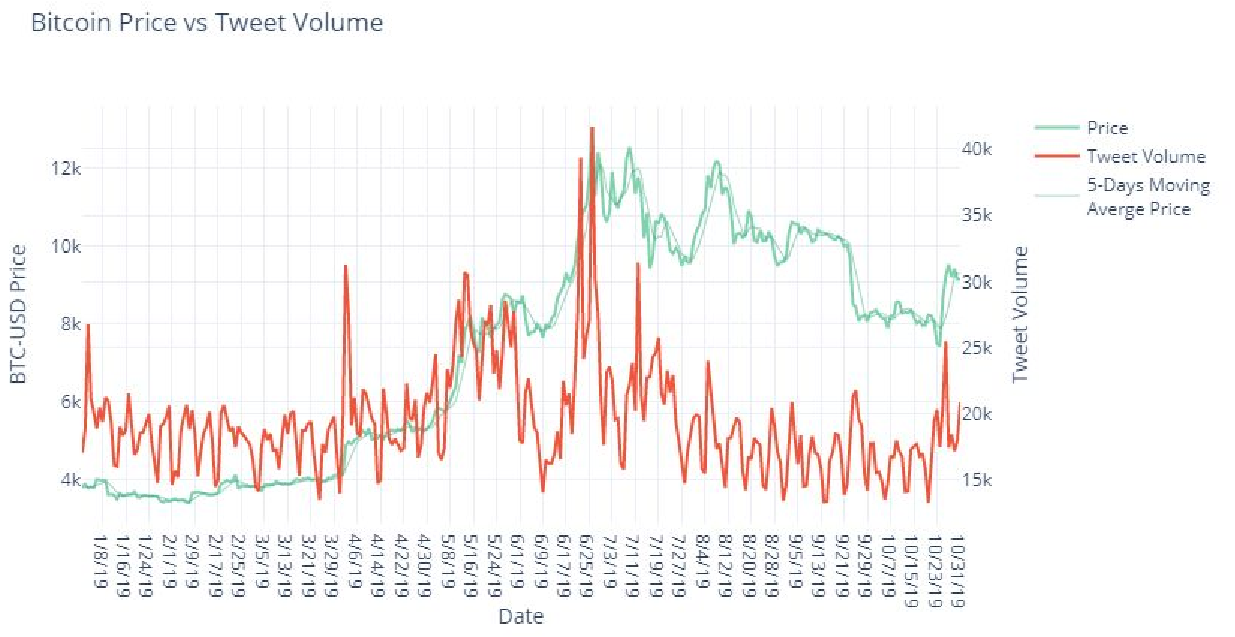


Fig 11: Bitcoin Closing Price (Normal and 5-Days Moving Average) and Tweet Volume

As seen in Fig 11, we can observe a relationship between Tweet Volume and Price. The tweet volume generally increases and decreases with the price change. Hence, it is possible to use Tweet volume as a feature. We also chose to use the Tweet Volume movement direction. As seen in Fig 12, the shape of the two plots are very similar, hence indicating that there may be a relationship between Bitcoin Price movement and Tweet Volume Movement.

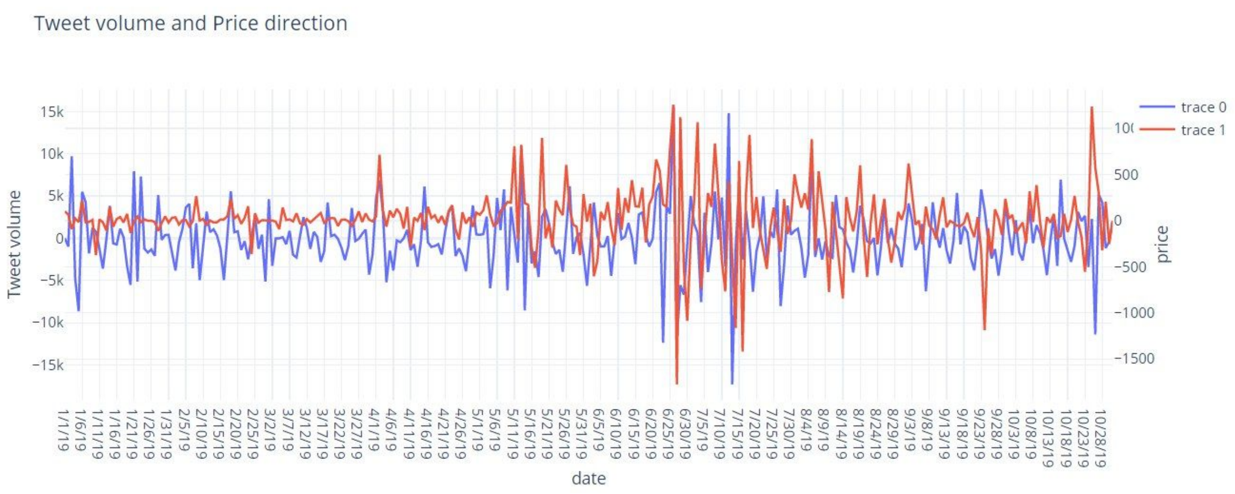


Fig 12: Tweet Volume Movement (Blue) and Bitcoin Price Movement (Red) from 1/1/19 yo 10/28/19.

2.2.3.3 Google Trends

We decided not to use Google Trends data with keyword 'BTC USD' as a feature for shorter term price prediction. There is a weak correlation when we compared the graph of daily Google search with the BTC moving average. There was a roughly similar trend from April 2019 till mid October 2019. However, the trend was not reflected from a January to April 2019.

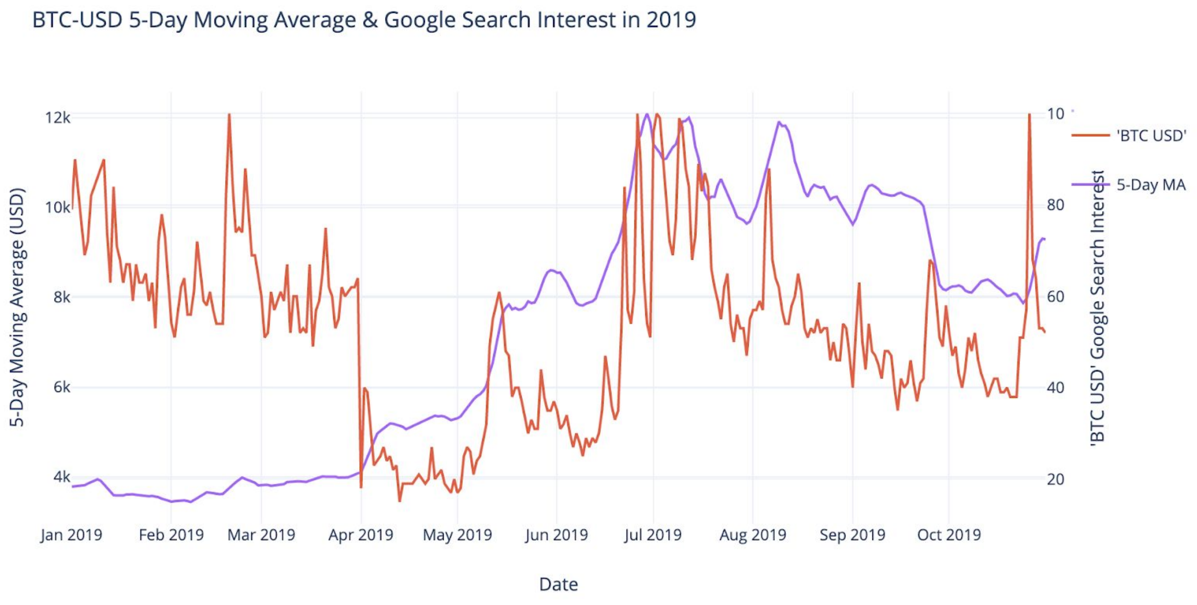


Fig 13: Correlation between daily Google Trend and BTC 5-Day Moving Average

Hence, based on our analysis, we have chosen 4 features to use as input to our classification model. They are Weighted sentiment score (VADER), Weighted sentiment score (TextBlob), Tweet Volume and Price Direction.

3 Model Variants

The set of inputs, chosen used for our models are:

1. Weighted Vader Sentiment
2. Weighted Textblob Sentiment
3. Tweet Volume
4. Tweet Volume Movement Direction
5. Multi Feature (input all features)

3.1 Preliminary Naive Model

The first model that we attempted was to use logistic regression to classify our tweets. Logistic regression is a linear classifier that outputs a probability of occurrence. Logistic regression uses the sigmoid function (exp(s)/1+exp(s)), graphically resulting in an S shape that lies between 0 and 1.[12] This makes it very suitable for binary classification as the output of logistic regression which can predict how probable that bitcoin prices will increase.

Using the daily average tweet sentiment as our feature and daily price movement as our labels, we attempted to predict the price movement. The following is the confusion matrix for this initial model:

	Predicted No	Predicted yes
Actual No	0	22
Actual Yes	0	31
Score	0.585	

The confusion matrix showed that our model classified and predicted all points as positive. Hence, we decided to plot the graph of Sentiment against direction of price change. (Fig 14)

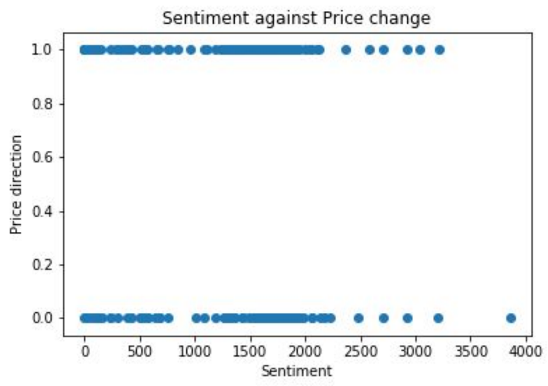


Fig 14: Sentiment score (VADER) against Price Direction (1 for positive and 0 for negative)

From this, we realised two issues. Firstly, we lost too much data by aggregating the price and sentiment over a day. Secondly, there might not be any correlation between sentiment and price movement. Hence to address these 2 issues, we added more features and reduced the time step from daily to hourly to predict our results.

3.2 Evaluation procedures

Train/test split

- Split the dataset into training and test set
- Better estimate of out-of-sample performance, but still a "high variance" estimate
- Useful due to its speed, simplicity, and flexibility

K-fold cross-validation

- Creating K train test splits and average the result together
- Even better estimate of out-of-sample performance
- Runs "K" times slower than train/test split

Scoring

- For all models, we used a **confusion matrix** and **accuracy** to measure how well it performs. Accuracy is a reliable indicator as our dataset contains an almost equal number for both classes, unlike the case of a spam classifier where predicting all emails as not spam will still result in a high accuracy.

Note that in our confusion matrices, predicted no implies that the price movement for that hour will be negative, while predicted yes implies that price movement is positive.

3.1 Logistic Regression

We initially used a keras' train_test_split with test size of 0.2 for our data to split into train and test. And the feature that performed the best was with our weighted VADER sentiment.

	Vader		**Textblob**		**Tweet Direction**		**Tweet Volume**		**Multi-Feature**	
	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes
Actual No	122	571	15	678	3	690	39	654	189	504

Actual Yes	104	634	16	722	3	735	38	700	168	570
Score	0.528		0.515		0.515		0.516		0.530	

As we can see, even though our accuracy increased merely by 0.015 over the initial, this was an improvement there are at least some that predicted "No". Using all the features we have allowed for the best result. However, 0.53 is still too low of a score to be satisfied and we attempted using k-fold cross validation instead.

K-fold Cross Validation

This resulted in a slightly better score, however this was mainly due to the fact that there was a higher ratio of "yes" in the overall dataset as compared to the test set earlier and that our model prefered to predict "yes" most of the time.

	Vader		**Textblob**		**Tweet Vol. Direction**		**Tweet Volume**		**Multi-Feature**	
	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes
Actual No	623	2764	120	3267	12	3375	361	3026	0	3387
Actual Yes	576	3188	119	3645	16	3748	470	3294	0	3764
Score	0.533		0.527		0.525		0.511		0.527	

Since there was no significant advantage of using K-fold cross validation, we stuck with **train_test_split** for the rest of the models. However, the alarming number of false positives were too high, and thus we continued with the other models mentioned below.

3.2 Random Forest Algorithm

The Random Forest algorithm [13] works as a large collection of decorrelated decision trees. It utilises the bagging technique to average noisy and unbiased models to create a model with low variance. The two key concepts of this classifier are:

1. Random sampling of training data points to build trees
2. Random subsets of features considered in splitting nodes

As we have several features with each correlating differently to the BTC price movement, random forest will minimise the effect of features with individually weak prediction power.

We used train_test_split to obtain the train and test set. Upon instantiating the random forest model, we tuned [14] random_state to 0 and limited max_depth of each tree to 2 to prevent overfitting. We then looped through different forest sizes (n_estimators) from 100 to 500 in order to select the biggest tree size for stable and stronger predictions. 400 trees gave the best predictions albeit the overall accuracy being quite poor.

	Vader		**Textblob**		**Tweet Vol. Direction**		**Tweet Volume**		**Multi-Feature**	
	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes	Predicted No	Predicted yes
Actual No	319	374	330	363	293	400	305	388	312	381
Actual Yes	345	393	352	386	322	416	336	402	304	434
Score	0.4976		0.5003		0.4955		0.4941		0.5213	

Correlated features are overall reduced in importance compared to one same decision tree built without correlated counterparts. Overall, the Random Forest did not perform well with a marginally better performance in predicting True Negatives.

3.3 K-Nearest Neighbors (KNN)

The K-Nearest Neighbor (KNN) algorithm is a non-parametric method that can be used for classification [15]. The value of K will determine the number of neighbours used to assign the most common class. Eg, if K = 3, the predicted price direction change would be positive if majority out of 3 nearest neighbours are labelled to be positive. There is no weighting scheme used for our KNN model and we set the model parameters of K to be 3,4,5.

The following is the confusion matrix result on the various inputs:

	Vader					
N. neighbours	3		4		5	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	311	382	315	378	313	380

Actual Yes	346	392	347	392	344	394
Score	0.491		0.493		0.494	

	Textblob					
N. neighbours	3		4		5	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	332	361	329	364	335	358
Actual Yes	346	392	342	396	344	394
Score	0.506		0.507		0.509	

	Tweet Dir					
N. neighbours	3		4		5	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	307	386	423	270	303	390
Actual Yes	353	385	487	251	369	369
Score	0.484		0.471		0.470	

	Tweet Volume					
N. neighbours	3		4		5	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	298	395	421	272	311	382
Actual Yes	356	382	474	264	364	374

Score	0.484	0.471	0.470

	Multi-Feature					
N. neighbours	3		4		5	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	328	365	331	362	325	368
Actual Yes	330	408	324	414	321	417
Score	0.514		0.521		0.519	

While KNN was a lot better at predicted true negatives, however, false negative rates were much higher and as a whole performed far worse than our other classifiers based on accuracy.

Without the use of a neural network, it seemed like twitter sentiment could not be used as a feature in predicting prices.

3.4 LSTM-RNN

Recurrent neural network (RNN) [16] is used for time-series prediction. They are networks with loops, allowing previous information to persist in solving present task. Long Short Term Memory networks (LSTM) [17] are a type of RNN, which solves the problem in long-term dependencies. Each cell decides what to store, allowing reads, writes and erasures, via gates implemented using sigmoid function.

Our unidirectional implementation LSTM only preserves information of the past features which are used as inputs. As we are inputting four different features to predict the price direction, our model can be visualised as seen below:

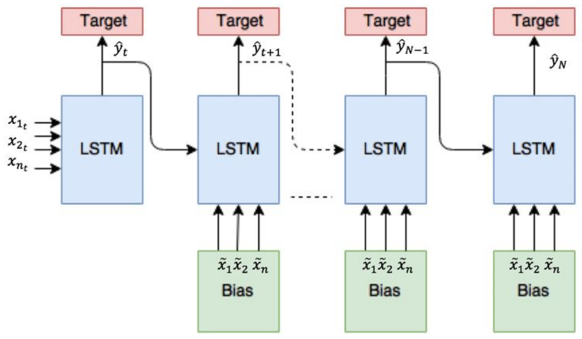


Fig 15: Multivariate LSTM with 4 inputs and single output [18]

	LSTM	
	Predicted No	Predicted Yes
Actual No	477	178
Actual Yes	177	598

Our first attempt at our LSTM model was to use the sentiment score and price at the previous hour to predict if the price will be higher in the next hour. Based on all our earlier classifiers, we found that VADER performed the best out of all the features and thus we decided to use its sentiment score instead of Textblob. Our initial single LSTM layer gave us a much better score of 73% and we spent the remaining of our time optimizing the neural net.

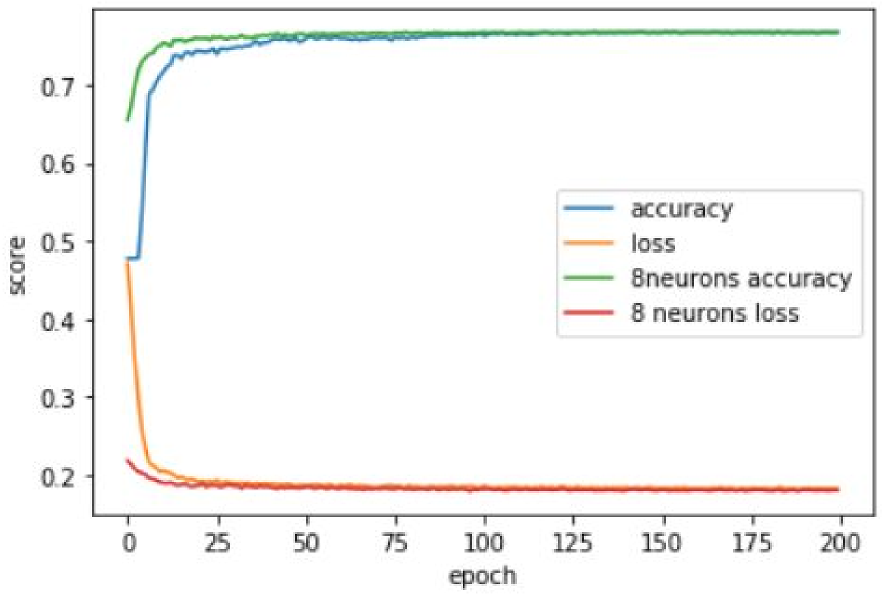


Fig 16: Accuracy and Loss for 4 and 8 Neurons in LSTM

We see that with a regardless of LSTM(4) or LSTM(8), the loss and accuracy both converged around 50epoch, and since the number of neurons were not important, we decided to stick with use 4.

We also attempted to find out if using VADER alone in the LSTM model would give us better results, and the results shown below proved otherwise.

	LSTM (VADER ONLY)

	Predicted No	Predicted Yes
Actual No	172	488
Actual Yes	179	591
Score	0.533	

Similarly, when we repeated it with the other features, we obtained the same results. Next, we attempted to improve the model by feeding the LSTM model one additional feature, tweet volume at time t-1.

	LSTM (with initial model + tweet volume)	
	Predicted No	Predicted Yes
Actual No	507	189
Actual Yes	161	573
Score	0.755	

This gave us a 2% score improvement over our initial model. After which we tuned the layers in an attempt to improve our results. One approach was to stack multiple LSTM layers, this causes the middle LSTM layer to output at every time step rather than just one output for all the timesteps. This allows our hidden layer to operate at a different timescale and hopefully improve our result. However, this did not give us a better result than the previous and we attempted to build a bidirectional LSTM model.

This involves duplicating the first recurrent layer in the network, and provides the first layer the input as usual and the second layer takes in the reverse of the first. Performing the bidirectional layer gives us a very minimal improvement, but it was a step in the right direction. To further improve our model, we fed data up to 4 hours before the current price to see if it would prove our model.

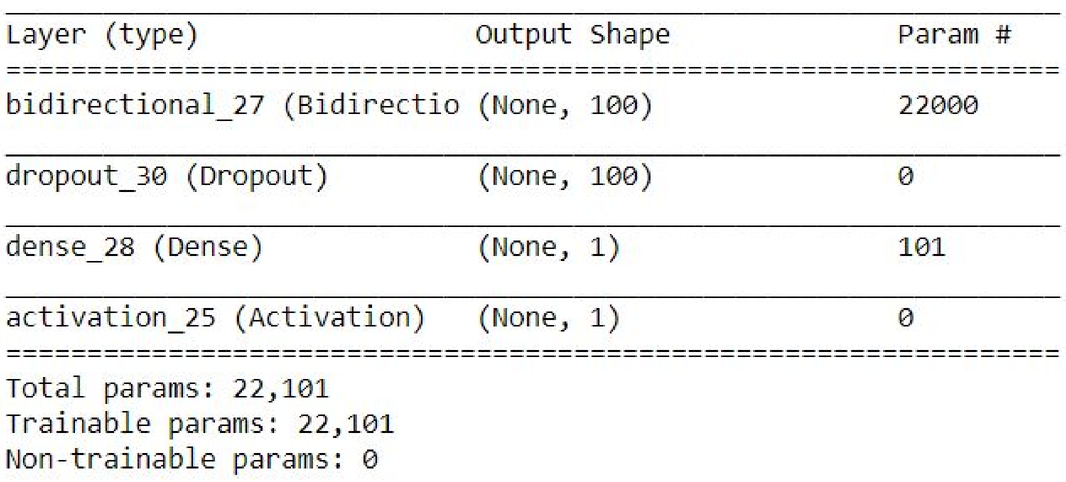
	LSTM - Bidirectional							
Timeseries	1		2		3		4	
	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes	Predicted No	Predicted Yes
Actual No	499	184	526	168	481	186	502	179
Actual Yes	160	587	149	587	176	587	164	585

Score	0.759	0.778	0.7468	0.760

As seen in the table above, we obtained the best result when we used 2 hours worth of features. Using more than 2 hours worth of features seemed to produce more noise and decreased the accuracy of our model.

Bidirectional LSTM model runs our inputs in two ways, one from past to future and one from future to past. This provides a better understanding of the context which contributed to the better performance.

Our final LSTM model uses a bidirectional LSTM with 4 units, 50 epochs, 100 batch size.
We also used a dropout layer of 0.2 to prevent overfitting.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_27 (Bidirectio (None, 100)               22000
_________________________________________________________________
dropout_30 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_28 (Dense)             (None, 1)                 101
_________________________________________________________________
activation_25 (Activation)   (None, 1)                 0
=================================================================
Total params: 22,101
Trainable params: 22,101
Non-trainable params: 0
```

Fig 17. Summary of Bidirectional LSTM model

5 Conclusion

Challenges

One of the main challenges was to find a good dataset, which we tried and in the end resorted to collecting data on our own. A labelled dataset would have allowed us to build our own corpus built specifically for analyzing crypto currency or explore other forms of sentiment analysis. This may result in more accurate sentiment scores to predict the price movement.

Collecting the dataset was a challenge too, as the GetOldTweets3 API actually crawls the Twitter webpage, and on certain requests, we faced some HTTP errors and were missing a few hours of data on certain days.
The collection and preprocessing of data took a long time to run, even after optimizing our code by using dataframe.apply rather than looping through the dataframe. Running sentiment analysis on 6million tweets using textblob took 8-10 hours with 8 threads. At that point in time, we were unfamiliar with Apache Spark and Hadoop and perhaps the use of a distributed system would have sped things up.
Hence we limited ourselves to just tweets in 2019, as we wanted to capture a year with both bull and bear trends. If time and computational power was not an issue, we would have used data dating back to 2017, which may improve our model.

Unlike other problems where the choice of model was obvious, we spent a large amount of time experimenting with all the options available, and did not have much time to optimize our final model. However, we were also satisfied with an accuracy of 77.8% which was far better than what we expected.

Summary

In conclusion, we found that using Twitter sentiment alone was insufficient at predicting Bitcoin prices. However, a combination of sentiment and price was more than enough to improve the accuracy of a LSTM model. The figure below summarizes our entire process with this project.

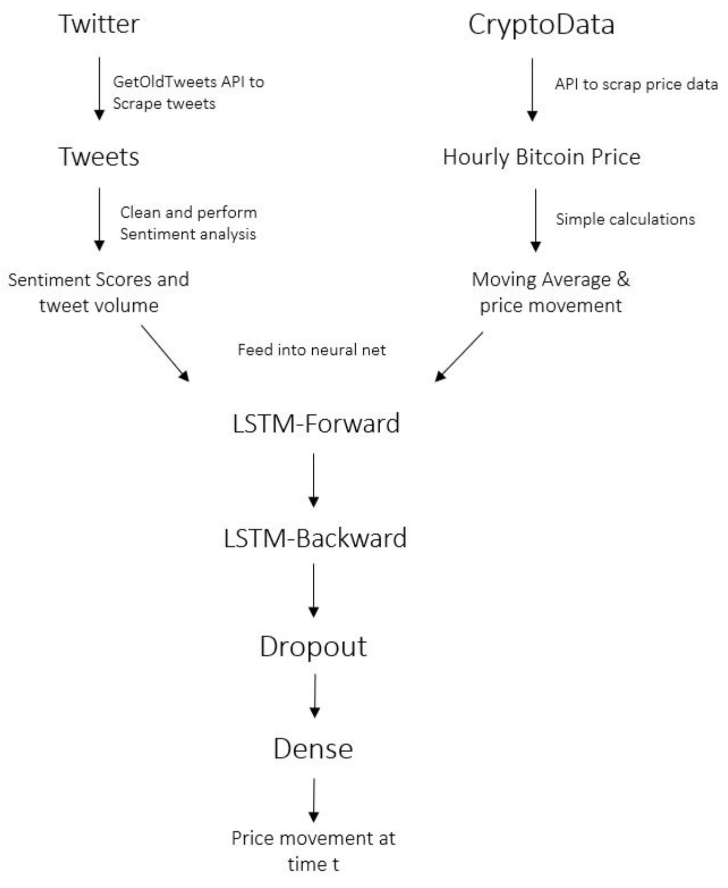


Fig 18 : Summary of Project

In addition, Bitcoin price prediction a task too complex for classical machine learning algorithms and required the use of neural networks. Given more time, we would be able to tackle all our challenges mentioned and further improve our model. Future works include predicting not just the movement of the next hour, but a few hours later or even the actual price of Bitcoin.

References

[1] https://www.coindesk.com/price/bitcoin
[2]https://www.buybitcoinworldwide.com/volatility-index/
[3] https://medium.com/swlh/top-5-factors-influencing-bitcoin-price-441cd9479bba
[4] https://bitinfocharts.com/comparison/bitcoin-tweets.html
[5] https://www.coindesk.com/bitcoin-hits-17-month-high-above-12-9k
[6] https://woobull.com/woos-law-of-bitcoin-user-growth-bitcoins-adoption-curve/
[7] https://pyldavis.readthedocs.io/en/latest/
[8]
https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/
[9] https://github.com/cjhutto/vaderSentiment
[10] https://textblob.readthedocs.io/en/dev/
[11] https://github.com/Mottl/influencers/blob/master/cryptocurrencies_news.txt
[12] https://medium.com/greyatom/logistic-regression-89e496433063
[13]https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652
[14] https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/
[15] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[16]
https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912
[17] https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[18]
https://www.researchgate.net/figure/Multivariate-LSTM-with-4-features-and-a-single-output-The-output-of-LSTM-at-time-t-is_fig5_324600237