Predicting Ethereum Price Change Using News Articles

Ashutosh Bansal
19323385

Ming Jun Lim
21337483

Markus Scully
16321114

I. Introduction

THE cryptocurrency market have witnessed an explosion in the recent years with Bitcoin[1] hitting lots of new time high and many investors moving to this trading space [5]. With the internet constantly evolving, social networking sites such as Facebook, Twitter, Instagram and etc have become a means of communication between people. There have been researches in text sentiment analysis to predict price changes for Bitcoin [5][6] as Bitcoin is considered a dominant cryptocurrency. However many other cryptocurrency have been developed and have been in demand such as Ethereum [2]. The project will be tackling Ethereum market price by estimating if the future price increases or decreases based on text processing sourced from the web such as social networking sites and forums. The problem is converted into a classfication problem by comparing the future market price with current market price instead of a regression problem of estimating the future market price. Can news or posts of Ethereum articles really affect the market price?

The input to our algorithm is text sourced web scrapers on social networking sites. We then compare several machine learning methods such as k-Nearest Neighbours (kNN), logistic regression and Support Vector Machine (SVM) to output a predicted label, +1 if price increased or -1 if price decreased.

II. Dataset and Features

The dataset consists of Ethereum related articles and class labels gathered from Twitter, Reddit and Binance[3] over the last four years. Data from Twitter and Reddit are merged with a similar column names: *date content* and *popularity*.

A. Data Source

1) Twitter

Twitter provides an official API for accessing tweets but access to this requires an approval process [4]. Instead, the open-source Python library snscrape [3] was used to collect 240 tweets for each day in the range January 2017 to October 2021, filtered by search for the word ”Ethereum”. The popularity content for each tweet was calculated by summing together the number of retweets, likes and replies it received. This popularity value was then normalised by dividing it by the highest popularity value of all collected tweets.

[1] Bitcoin is the first well known cryptocurrency started in the late 2000s

[2] Ethereum is a programmable blockchain and cryptocurrency described as Ether

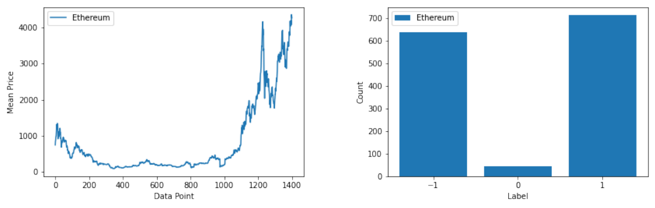
[3] Binance is one of the largest cryptocurrency exchange

2) Reddit

Reddit is divided into smaller sub-communities relating to specific themes called ”subreddits”. The subreddits ”EtherMining”, ”Cryptocurrency”, ”Cryptocurrencies”, ”CryptoMarkets”, ”EthTrader”, ”Ethereum”, and ”Bitcoin” were selected as source of data because of their relevancy. Comments posted on submission on each of these subreddits were scraped using the library psaw which itself pulls data from service Pushshift. The popularity value of each comment was determined by its ”karma” value, a measure of how much positive feedback it has received from other users of the site. Up to 240 comments were scraped per subreddit per day, for a maximum of 1680 per day if that many comments had actually been posted.

3) Binance

Binance provides a set of API [2], WebSocket endpoints available to collect live data and old historical data [1]. Monthly trade data of ETHUSDT[4] symbol is collected from August, 2017 to October, 2021 with a total of 657,446,190 data points. These data points are processed by calculating the average price by days and discretizing the continuous values into +1 and -1 class labels by comparing it with the previous day resulting into 1,537 data points.



(a) Mean price against Data points (b) Count against labels

Fig. 1: Overview of Binance dataset collected

There is another class label ”0” in figure 1.b. This is due to the first day of every month unable to calculate different with the day before. It was set to ”-1” to have binary labels.

B. Feature Extraction

Several pre-processing steps are applied. The first step is tokenization, splitting input character sequences into tokens. The endings of the tokens are chopped of by stemming. The tokens are pruned using a set of stop words as they contribute little to no information such as words ”is”, ”the”, ”a”, etc. Tokens appearing frequently and rarely were also pruned through the `min_df` and

[4] ETHUSDT is a symbol for Ethereum and USDT pair, 1 USDT is 1 USD.

max_df parameters in tokenization, this step was very important to reduce the feature size. The algorithm uses Natural Language Toolkit nltk library providing stemming functions such as PorterStemmer and a list of stopwords nltk.corpus.stopwords.words("english").
Some general pre-processing on text data from [6] were applied onto our dataset consisting of:

- Converting all words to lower case.
- Removing words containing numbers.
- Removing symbols from words (”#”,”@”).
- Removing words containing nonenglish characters.

These sequence of tokens are mapped to a feature vector using the bag of words model, the model consists of hyper-parameter n-grams and cross validation for 1-gram single token and 2-gram pairs of token are evaluated in section ???. The n-gram hyper-parameter enables model to preserve some word ordering but feature vector size can grow quickly.

III. Methods

A. k-Nearest Neighbours

K-Nearest Neighbours (kNN) is an instance-based model that uses training data to make predictions. Given a feature vector x to make prediction, the distance of x is calculated with all training data, selects k closest training points and assigns majority label of the neighbours for classification problem. kNN requires a distance function, hyper-parameter *k* of neighbours to be considered, weighting of neighbour points and method for aggregating *k* neighbour points N_k.

$$d(x^{(i)}, x) = \sqrt[2]{\sum_{j=1}^{n}(x_j^{(i)} - x_j)^2} \tag{1}$$

$$w^{(i)} = 1 \tag{2}$$

$$w^{(i)} = e^{-\gamma d(x^{(i)},x)^2} \tag{3}$$

$$sign(\frac{\sum_{i \in N_k} w^{(i)} y^{(i)}}{\sum_{i \in N_k} w^{(i)}}) \tag{4}$$

Equation (1) is Euclidean distance commonly used for kNN. Equation (2) uniform and equation (3) Gaussian are methods weighting of neighbour points. For the Gaussian weighting, training points further away from the target variable *i* are attached with lesser weight compared to nearer ones. Equation (4) is the method for aggregating *k* neighbour points N_k, it is a majority vote when uniform weighting neighbour points is used.

B. Logistic Regression

Logistic regression is a linear model that uses $sign(\theta^T x)$ to quantize continuous values to output labels $+1$ when $\theta^T x > 0$ and -1 when $\theta^T x < 0$. Decision boundary is $\theta^T x = 0$ and dataset is considered to be linearly separable when classes can be separated by decision boundary.

$$\theta^T x = \theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n \tag{5}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} log(1 + e^{-y^{(i)}\theta^T x^{(i)}}) \tag{6}$$

Equation (5) is the weighted combination of the input features and weight. Equation (6) is the cost function for logistic regression. The learning algorithm uses gradient descent to minimises cost function $J(\theta)$ by pushing weights θ towards direction away from decision boundary to minimise loss. The cost function $J(\theta)$ is convex and gradient descent iterates by moving downhill reaching the minimum point.

$$R(\theta) = \sum_{j=1}^{n} |\theta_j| \tag{7}$$

$$R(\theta) = \theta^T\theta = \sum_{j=1}^{n} \theta_j^2 \tag{8}$$

Regularisation can be added to logistic regression by using penalty term in the cost function such as L1 penalty equation (7) and L2 penalty equation (8). These penalty terms uses hyper-parameter by affecting minimisation of first term or penalty term in learning. L1 penalty encourages sparsity of solution, causing weights to be 0. L2 penalty encourages weights θ to have smaller values.

C. Support Vector Machine

Support Vector Machine (SVM) is a classifier similar to logistic regression, the prediction uses $sign(\theta^T x)$ and the output is mapped to $+1$ and -1.

$$max(0, 1 - y\theta^T x)) \tag{9}$$

The main difference lies in the cost function as shown in equation (9), SVM uses a ”hinge” loss function. The cost function of SVM is non-smooth. There is no penalty to all values of θ when the prediction is correct. Penalty for regularisation is mandatory for SVM to get sensible behaviour due to $\theta^T x$, scaling the term up will output loss of 0. Example prediction is $+1$ and label is $+1$, Equation 9 will be $max(0, 1 - (+1)(+45)) = 0$

$$\theta^T\theta = \sum_{j=1}^{n} \theta_j^2 \tag{10}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \frac{\theta^T\theta}{C} \tag{11}$$

Equation 10 is the sum of squares elements in parameter vector that penalises large values of θ. Equation 11 is the final SVM cost function. The first term depends on how much the model predicts training data and C in the penalty term is a hyper-parameter affecting training in minimising which terms in the equation. The learning selects θ that minimises cost function $J(\theta)$.

IV. Experiments

5-fold cross validation were used to select hyper-parameters for models mentioned in section III. The dataset metioned in section II-A were grouped by dates resulting with a total of 1400 data points from *2018-01-01* to *2021-10-31*. The dataset was split into training and test set with a ratio of $80:20$. The baseline models used to evaluate against these models were a model always predicting the most common class and a model randomly predicting class.
The choice of metric used for evaluation F_1 score, recall, precision and accuracy. The ROC curve and AUC were also plotted and compared with other classifiers and baseline.

$$F_1\text{score} = \frac{2TP}{2TP + FN + FP} \tag{12}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{13}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{14}$$

Equation (12) is the F_1 score metric measuring the harmonic mean of precision and recall. Equation (13) is the recall metric. Equation (14) is the precision metric. *TP* represent true positive, *FP* represents false positive, *TN* represents true negative and *FN* represents false negative.

A. k-Nearest Neighbours

B. Logistic Regression

C. Support Vector Machine

The range of C values used are $[1, 50, 100, 300, 600, 800]$ to select hyper-parameter for the penalty term in equation (11) through cross validation.

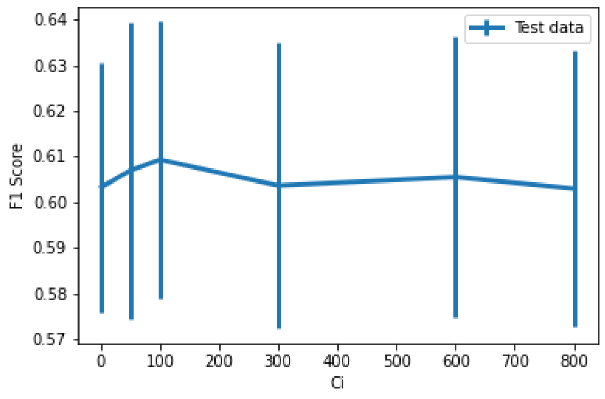


Fig. 2: Range of C values

Figure (2) is an error plot of cross validation in the range of C. The selected C value hyper-parameter is 100 as shown in the figure. The weight of the penalty is controlled by hyper-parameter C, higher value results in lower weight likewise for lower value as described by equation (11).

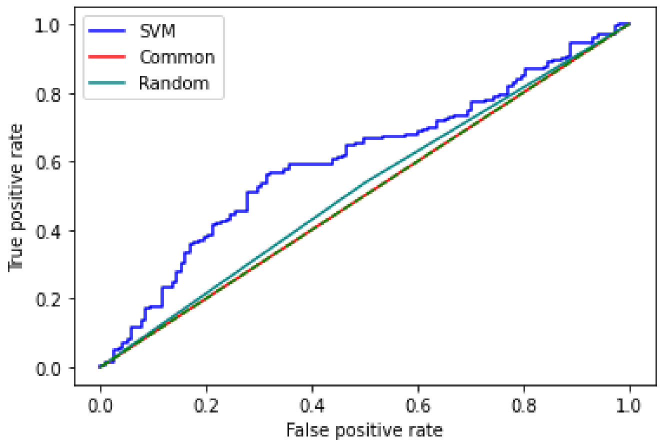


Fig. 3: ROC curve plot of all algorithms

V. Results

VI. Discussion

VII. Summary

References

[1] Binance public data. https://github.com/binance/binance-public-data. Accessed: 2021-12-02.
[2] Binance websocket docs. https://github.com/binance/binance-spot-api-docs/blob/master/web-socket-streams.md. Accessed: 2021-12-02.
[3] Snscrape github repository. https://github.com/JustAnotherArchivist/snscrape. Accessed: 2021-12-02.
[4] Twitter api. https://developer.twitter.com/en/docs/twitter-api. Accessed: 2021-12-02.
[5] Nguyen Huynh Huy, Bo Dao, Thanh-Tan Mai, Khuong Nguyen-An, et al. Predicting cryptocurrency price movements based on social media. In *2019 International Conference on Advanced Computing and Applications (ACOMP)*, pages 57–64. IEEE, 2019.
[6] Otabek Sattarov, Heung Seok Jeon, Ryumduck Oh, and Jun Dong Lee. Forecasting bitcoin price fluctuation by twitter sentiment analysis. In *2020 International Conference on Information Science and Communications Technologies (ICISCT)*, pages 1–4. IEEE, 2020.

Algorithm	**F1 Score**		**Recall**		**Precision**		**Accuracy**	**AUC**
	+1	-1	+1	-1	+1	-1		
Most Common (baseline)	0.73	0.00	1.00	0.00	0.58	0.00	0.58	0.500
Random (baseline)	0.57	0.53	0.51	0.61	0.64	0.47	0.55	0.519
k-Nearest Neightbours (kNN)	0.72	0.06	0.96	0.03	0.58	0.36	0.57	0.495
Logistic Regression	-	-	-	-	-	-	-	-
Support Vector Machine (SVM)	0.63	0.53	0.60	0.56	0.65	0.51	0.59	0.604

TABLE I: Results of algorithms with evaluation metrics