Predicting Bitcoin Market Trends Using Sentimental Analysis of Tweets

Hsi-Min Chou
Dept. of Computer Science
hchou026@ucr.edu

Henry Wu
Dept. of Computer Science
hwu093@ucr.edu

Lin Cong
Dept. of Statistics
lcong004@ucr.edu

Neil Ian Franklyn Castellino
Dept. of Computer Engineering
ncast085@ucr.edu

Varun Chawla
Dept. of Computer Science
vchaw001@ucr.edu

ABSTRACT

Twitter has evolved as a popular micro-blogging platform where people from all over the world interact with each other and exchange their views. Owing to the virality in reach of cryptocurrency with advancements in transaction processing and the enormous growth of Bitcoin since its inception, global conversations surround BTC and its appeal. From hedge fund companies to global business tycoons to space entrepreneurs, Bitcoin has been backed by people, who other people look up to. Similarly, Twitter has advanced to become a true reflection of how users in the real world react towards things of importance. As observed in real life, the overall sentiment of people influences market trends, it is a very good idea to analyze the correlation between sentiments and Bitcoin market trends over time. This analysis can not only greatly aid in showing historical correlation but also help in making better predictions on Bitcoin market trends in the future.

1 Introduction

With the growth of the modern internet, the world has witnessed a massive user presence over social media across the globe. Among several companies, Twitter has been one of the top applications used by people daily to check what is trending, how retweets of news can affect their political perspectives, and how stocks that they invested in can change. It has become a great source to note sentiments of people towards different entities which can affect its worth and one particular entity out of these is #Bitcoin which is a topic of interest for many cryptocurrencies enthusiasts. The goal of our project is to study the impact of tweet text of Bitcoin over its price. Due to the restriction of collecting tweets through Twitter API, we combine those tweets with Kaggle data to robust our prediction. In our first step, we use a streaming pipeline to fetch tweets and save them in parquet format. Parallelly, after receiving the tweets, we pre-process them to avoid stop words and remove unnecessary information. For sentiment analysis, we peruse Textblob and Vader. After collating all the required features for prediction, we append all of them in a single CSV file that includes the tweet text, the tweet sentiment score, the Bitcoin price features and is then fed to ML models either by combining the Bitcoin price features with the sentiment score or by combining the price features with the word embeddings of the tweet contexts. We predict the Bitcoin price with both Machine Learning algorithms, such as Gradient Boosting, and deep learning models, such as the Long short-term memory (LSTM) model. To evaluate the prediction, we use model evaluation criteria like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R square to check the model performances. Section 2 below introduces the literature survey that is related to our project. Section 3 introduces how we deal with our data, including data preprocessing and pipelining. Section 4 shows how we visualize the result through a webpage. Section 5 indicates the methods that we analyze the sentiment of tweets and predict the Bitcoin market and the evaluation of Machine Learning Models, Section 6 briefs our concluding views and inside Section 7 we enlist technical challenges we faced during implementation of this project and how we tried to overcome them.

2 Related Work

The data is often disorganized and unsuitable to analyze directly. To make it feasible to work with, it needs to be modified and cleaned up to enable detailed analysis. As noted by Kharde & Sonawane, (2016) [5], tweets have many elements that must be removed or modified, such as URLs, hashtags, stop words. Specifically, URLs, hashtags, and stop words must be removed and emoticons substituted with the sentiments they represent. Through the paper Medhat Hassan & Korashy H (2014) [6], sentiment analysis is defined by the intensity of each word in the sentence, which means it requires a pre-defined dictionary classifying negative and positive words. Moreover, the paper stated that there are three main approaches in order to compile the opinion word list, such as dictionary-based, corpus-based and lexicon-based, and natural language processing techniques. We can see that the dictionary-based approach is the first method to collect a set of opinion words. However,

it has the disadvantage of the inability to find opinion words with domain and context-specific orientations. Later, the corpus-based approach helps to solve the problem. In our project, we decide to compare two different tools of sentiment analysis, which are TextBlob and Vader, both tools are all sentiment analyzers with lexical-based approaches.

The stock price has been famous for its hardness to predict, the well-known Efficient Market Hypothesis (EMH) states that stock market prices are largely driven by new information and follow a random walk pattern, which makes it impossible to predict the market with 100% accuracy. So, for many years, researchers have been predicting the stock price based on the historical stock prices using the statistical regression models, such as ARIMA (Autoregressive Integrated Moving Average) models [1] and the machine learning algorithms, such as SVM (Support Vector Machine) [8]. However, recent research has revealed that the stock price can be forecasted at least better than the random walk by using sentiment information in the market, such as opinions of investors on social media like Facebook and Twitter. This type of analysis for stock price came out as a result of the evolution of sentiment analysis, such as sentiment classification of textual information.

A major type of recently proposed stock price prediction model with sentiment information is to directly incorporate the sentiment analysis into the commonly used stock price prediction models as the first stage. For example, Wu et al. (2014) [9] proposed combining sentiment analysis and SVM to explore the relationship between stock price volatility and stock forum sentiment; Mittal and Goel (2012) [7] utilized the Twitter data to predict public moods and the degree of membership into 4 mood classes - Calm, Happy, Alert and Kind, then combined the degrees with the previous days' DJIA (Dow Jones Industrial Average) values to predict the stock market movements.

With the rapid development of deep learning, more and more deep learning techniques have also been implemented in the stock price prediction area. Ko and Chang (2021) [10] applied the popular Bidirectional Encoder Representations from Transformers (BERT) model to create the sentiment scores and sent to the Long Short Term Memory neural network (LSTM) model together with the stock prices information for future stock price forecast, because of the superior performance of LSTM model on time series prediction problem. Despite using sentiment analysis to derive intermediate results, such as sentiment score, polarity score, etc to combine with historical stock prices on predicting future stock prices, deep learning models are introduced in this task by directly using the text information without creating the score metrics mentioned above. For example, Mohan et al. (2019) [11] posed a method of using a convolutional network as a feature extraction tool to extract a 10-dimension vector from the word2vec representation of words in the documents and fed to the recurrent neural network (RNN) along with the normalized historic price to predict the stock price.

3 Pipelining and Preprocessing

In this section, we are going to brief about the pipeline, dataset, and data preprocessing to be used in this project. There will be two data sources for this project. We are going to fetch the data through the Twitter API via Tweepy Streaming Pipeline and pass it directly to for Sentiment score calculation. For further vectorization of the tweet and to make predictions over bitcoin price using Spark ML and Keras, we are going to use the Kaggle dataset having Bitcoin tweet information of 16 million Tweets and the Kaggle dataset having Bitcoin historic price information.

We design our Streaming Pipeline by creating a Tweet Listener Class which creates an instance of StreamListener and connects to Twitter API and retrieves one tweet at a time. We also create another Python file that will fetch the responses from the TweetListener Class routed over the bound host and connect those real time with Spark SQL. This serves as Tweepy-Spark streaming pipeline to trigger the listener class, retrieve tweets, perform preprocessing and generate polarity scores (explained in-depth in the following paragraphs). Inside the StreamListener instance, we define certain parameters like host number, port number, Twitter API Keys, type of authentication, and exception handling mechanism. The result in the form of refined text and polarity scores are saved in Parquet files which are then combined in a single CSV file. This generated file serves no help in modeling as some of the tweets fetched are irrelevant in nature and owing to stream processing problems detailed in section 7, we decided to use the Kaggle dataset of BTC tweets and combine it with the Kaggle dataset of Bitcoin prices. We then move forward with preprocessing, sentimental analysis, and modelling over the resultant dataset. We import packages PySpark SQL, Tweepy, socket, and JSON for doing these operations.

The tweet data contains many elements that are not helpful for performing sentiment analysis, so they need to be removed. The text is then tokenized to remove most punctuation and examine words that need to be removed. Non-English tweets are ignored and hyperlinks, special characters, stop words, hashtags, and usernames are removed. Each word is then lemmatized to standardize the word forms and converted into lowercase to avoid false unique words. Word embedding is used to represent words as numeric vectors, making it easier to perform sentiment analysis. This set of preprocessing techniques is similar to the one used by Angiani et al. (2016) [12], where the paper describes the preprocessing techniques it used, which include removing hashtags and filtering out emoticons that are not positive or negative.

4 Visualization

4.1 Back-End

A Flask server is used for the backend. The data is received in the form of CSV files after performing data preprocessing, sentiment analysis, and from the bitcoin price prediction model. This data is then passed on to the front-end to help the user visualize it.

4.2 Front-End

The UI was built using HTML, CSS and JavaScript. The User Interface is minimalistic. It is designed with an intention to make it easy and intuitive for the user.

4.3 Visualization

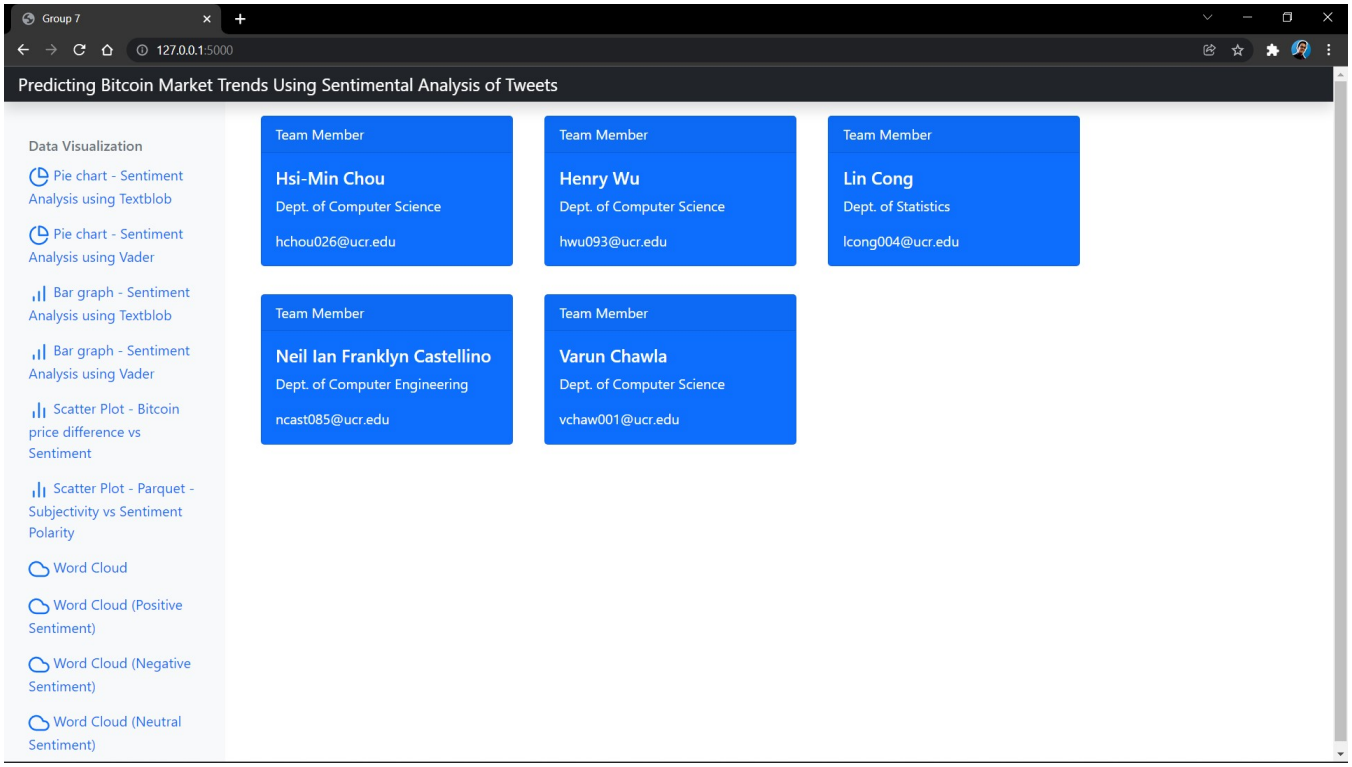

Figure 1. Website homepage

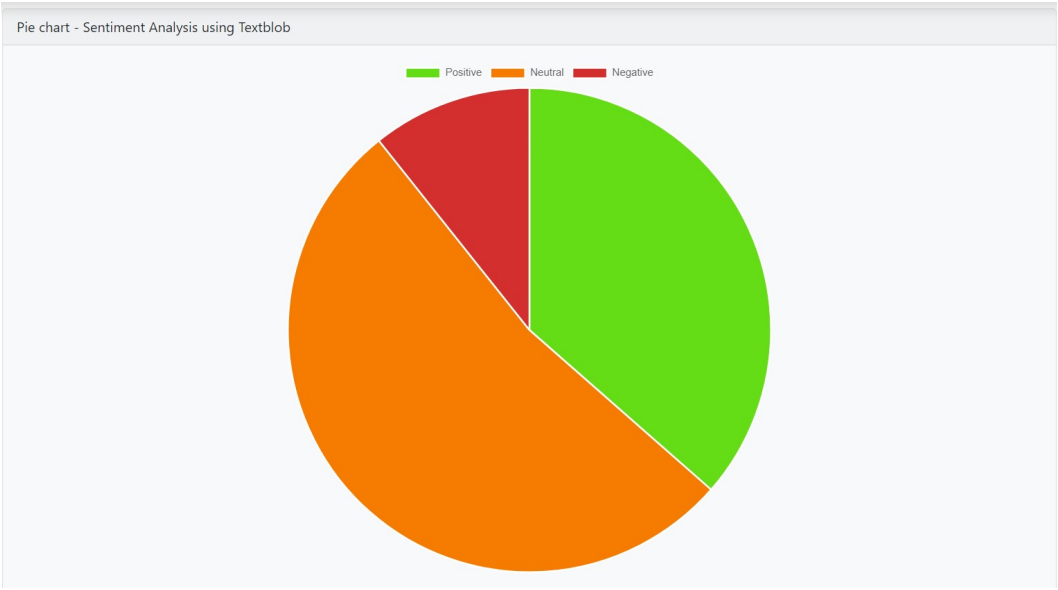

Figure 2. Pie chart generated using data from sentiment analysis of Bitcoin tweets with Textblob

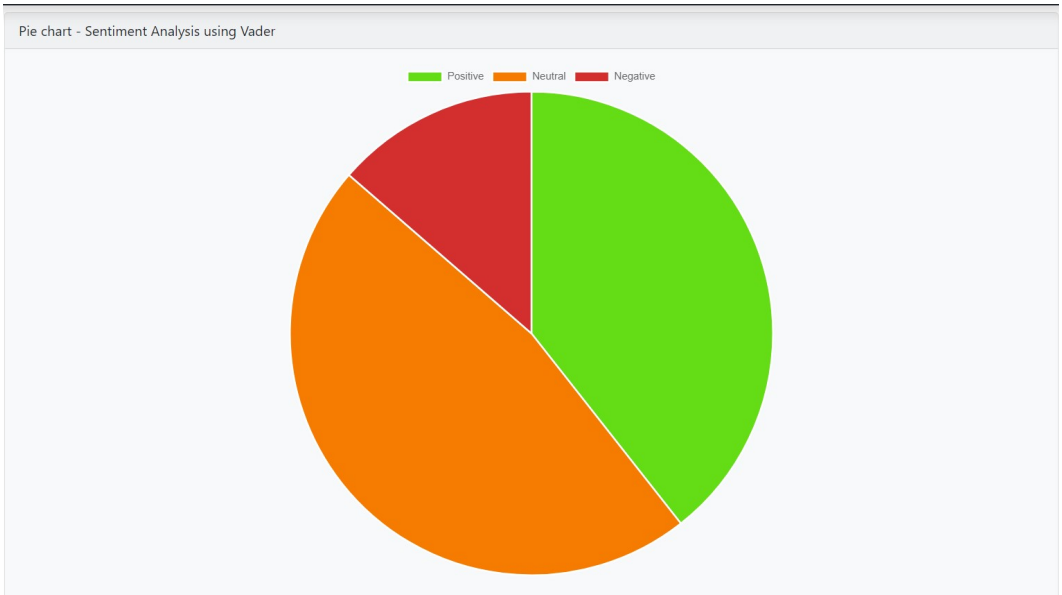

Figure 3. Pie chart generated using data from sentiment analysis of Bitcoin tweets with Vader

Figure 4. Bar graph generated using data from sentiment analysis of Bitcoin tweets with Textblob

Figure 5. Bar graph generated using data from sentiment analysis of Bitcoin tweets with Vader

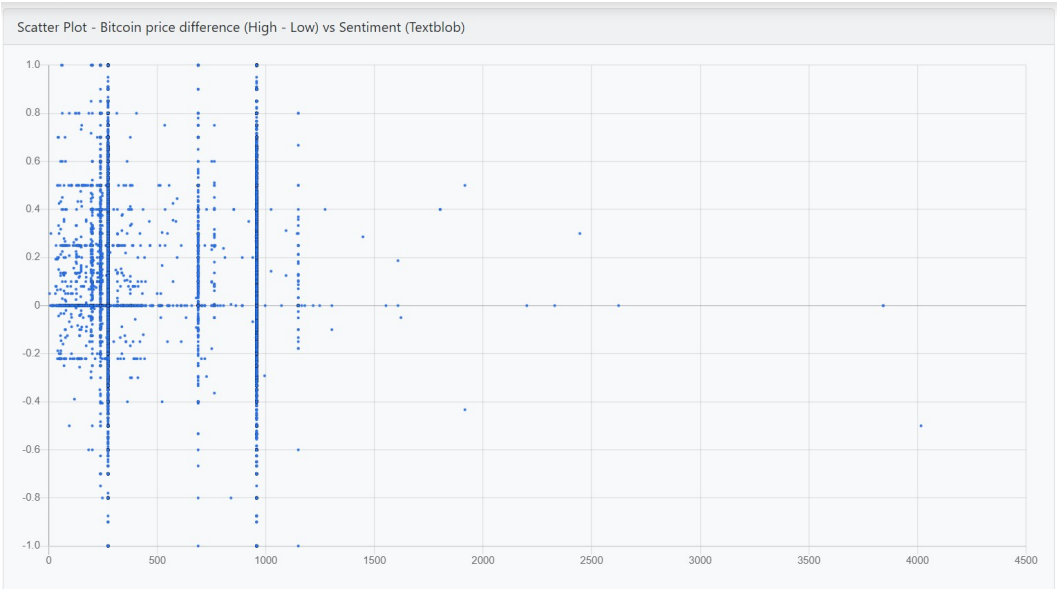

Figure 6. Scatter plot of Bitcoin price difference and Sentiment score

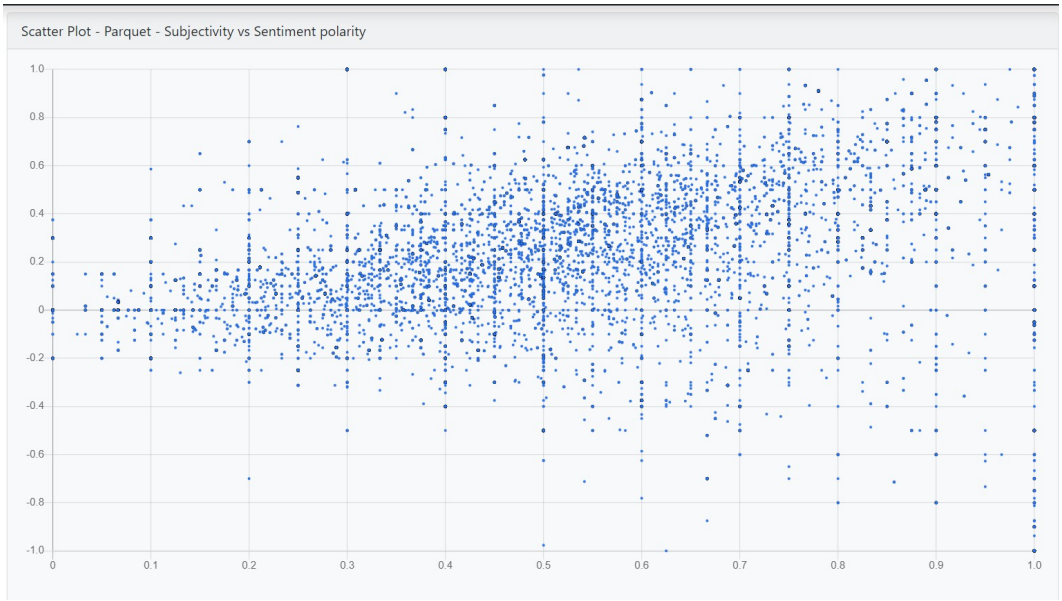

Figure 7. Scatter plot of Parquet Subjectivity and Sentiment polarity

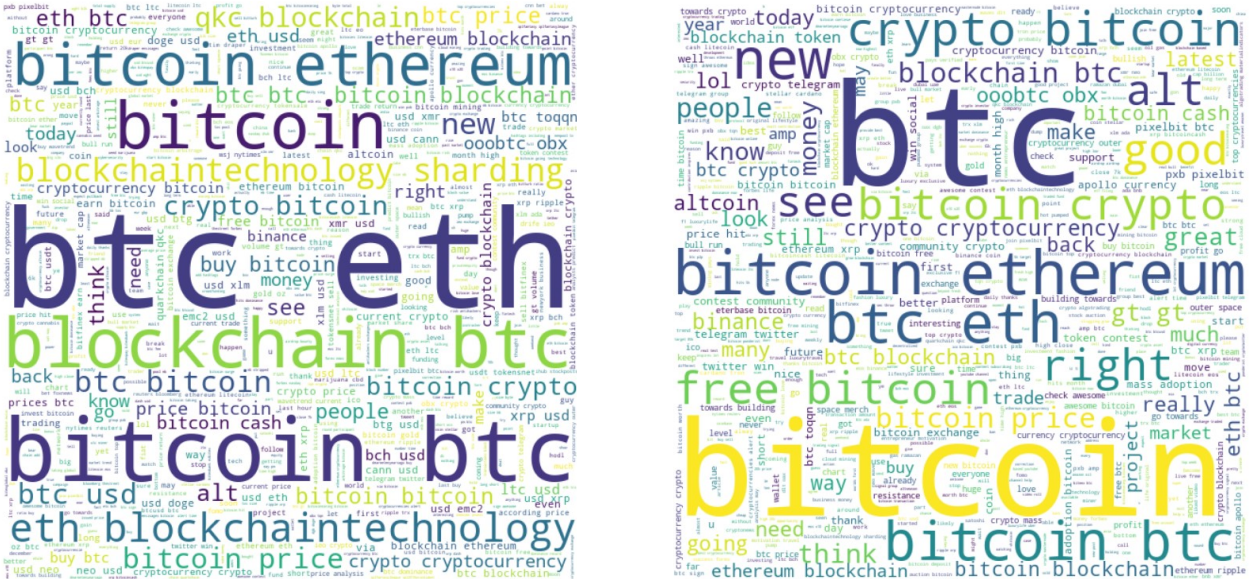

Figure 8. Word cloud generated using all the tweets and only the positive tweets collected

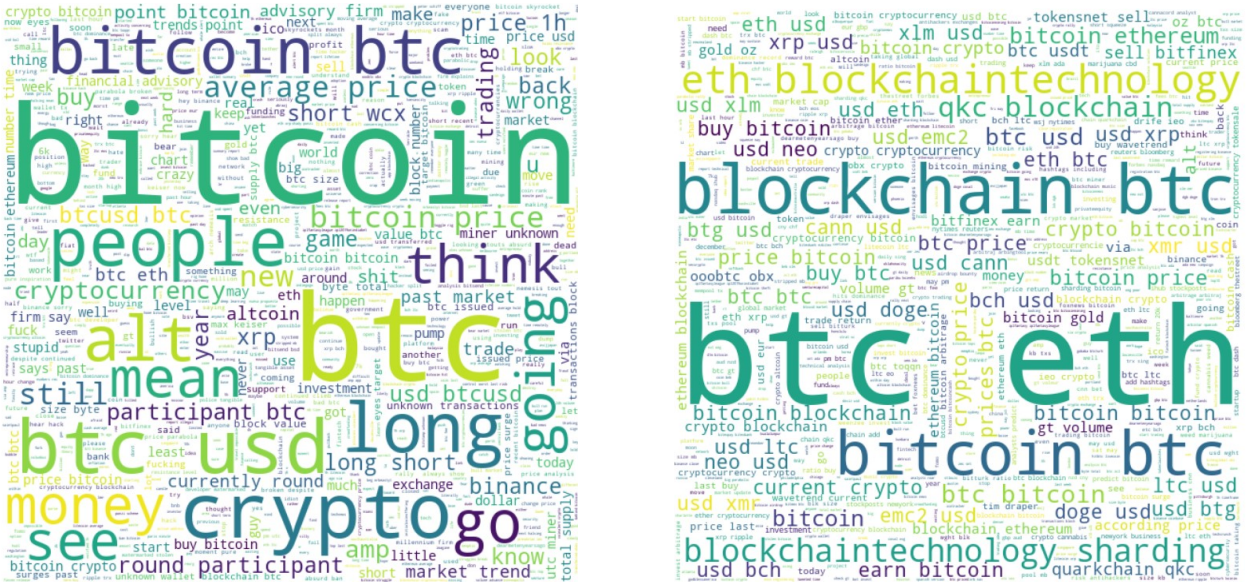

Figure 9. Word cloud generated using only the negative and neutral tweets collected

The data received by performing sentiment analysis on tweets from Twitter is represented in the form of bar graphs, pie charts, scatter plots, and word clouds. This is done to help the user visualize the data and draw conclusions from it.

5 Methodology

5.1 Sentiment Analysis

Sentiment analysis is also called opinion mining or emotion AI. In our project, we decide to compare two different sentiment analyzers, which are TextBlob and Vader. For both tools, we determine the results with positive, negative, and neutral. Moreover, we deal with the language English only at this moment.

5.1.1 Sentiment Analyzer - Textblob

In order to apply Textblob [3], we need to install the library first. We then use tweets as our input data to analyze the sentiment, which the data is already preprocessed. There are two sentiment scores that we receive, one is the real-time data that was fetched in recent days, and the other is from the Kaggle website which is from 2019. To run with the dataset, we use built-in functions "TextBlob.sentiment.polarity" to get our polarity score. In Textblob, it uses a complicated technique known as "averaging" in mathematics to compute the scores. If the score is smaller than 0, it represents negative; if the score is bigger than 0, it represents positive; else it represents neutral. After receiving the result, we can see the polarity and the sentiment of each tweet showing in the plot.

5.1.2 Sentiment Analyzer - Vader

For Vader [4], we do the same thing as Textblob, installing the Vader library and calling the object "SentimentIntensityAnalyser". We can see that in Vader, it returns a form like a dictionary which indicates the compound score and the distribution of 'neg', 'neu', and 'pos'. In this approach, the dictionary of 'neg', 'neu' and 'pos' means the percentage of combining every word's rating in the sentence that is rated by Vader. Also, the compound score is the same as the polarity score in Textblob. The computation of this score is based on summing the valence scores of each word in the lexicon. To be more precise, Vader uses Amazon's Mechanical Turk to get most of their ratings, which is a very quick and cheap way to get their scores. The threshold of the score at first depends on -0.05 and 0.05, however, it is normalized to be shown between -1 and 1 for easier representation.

5.1.3 Comparison Between Tools

In the resulting Figure 12, we can see that most of the sentiment categories are the same. However, there's a little difference between the positive and negative. Through the observation, we think it's because Vader has a better analysis with social media since it can deal with emoticons. Another discovery between both analyzers is that we found that the run time of Textblob is faster than Vader. We think that the reason for the different runtime might depend on the principle of their sentimental score computation. In conclusion, we think that Vader might be a better tool for our project.

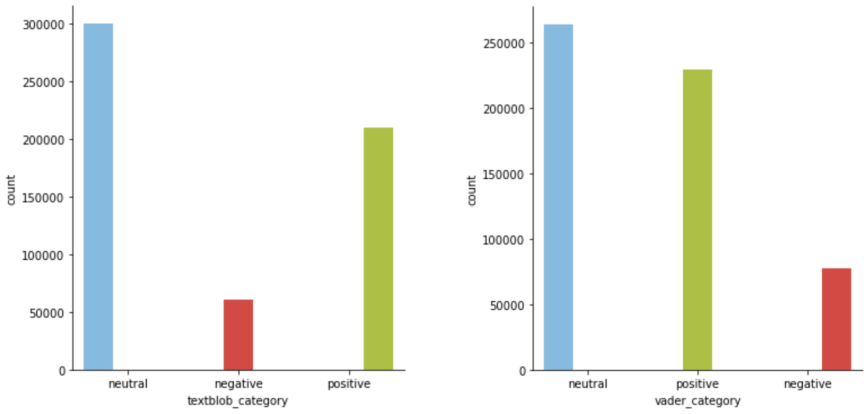

Figure 12. Comparison of sentiment analysis distribution between TextBlob and Vader. Bar graphs were generated using our own code.

5.2 Bitcoin Price Prediction

In this section, we use two types of models to incorporate the information from sentiment analysis to predict the Bitcoin price, combining with the historic stock data. The difference between the models is that the first type of model is to use the result derived from the previous sentiment analysis, which is the sentiment score for each tweet, to predict the Bitcoin price; while the second type of model is directly using the embedded tweet information to build deep learning models for the prediction task.

To compare between the two types of models, and consider the regression problem of predicting Bitcoin price, the commonly used model evaluation criterions such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R square are implemented to evaluate the model performances.

In addition, at the end of this section, another popular deep learning model - Long Short-Term Memory (LSTM) is implemented as an extension of the two types of the models mentioned above, because of its larger size of the model and advantage on modeling time series data.

5.2.1 Predictive Model with Sentiment Score

Based on the sentiment score from the sentiment analysis and the historic Bitcoin price data, traditional machine learning models are implemented. Two models are picked: the linear regression model as a baseline model, and the Gradient Boosting model. Similar works have been done, for example, Gupta and Chen (2020)[2] used latent Semantic Analysis (LSA) to approximate the human labels of the StockTwits website tweets and implemented Naive Bayes, SVM, and logistic regression to classify sentiments of the tweets into positive and negative sentiments, those labels are further used to create features such as daily positive tweets percentage which are combined with historical Bitcoin price data to classify the directions of movements between consecutive daily Bitcoin prices.

In our implementation, a window of 1 day is picked to include the related Bitcoin price features, which means that the historic Bitcoin price features include the six Bitcoin price-related features in the previous day: the opening price, highest price, lowest price, closing price, the volume of transactions, and market capitalization (USD). In this section, the six features from D-1 Day (D Day is the current day) are combined with the sentiment score to predict the closing price on D Day. A total of 52,652 tweets are split into 80% training and 20% testing, and the training set is utilized to build the linear model and the Gradient Boosting model using Spark ML through PySpark, while the testing set is utilized to assess the model performance.

5.2.2 Predictive Model with Embedded Tweets

Instead of using an intermediate result from the sentiment analysis, the original tweets information can be directly incorporated into the model through word embeddings, such as Word2vec, Global Vectors for Word Representation (GloVe), etc. The advantage of word embedding is that it can transform each tokenized word to a numerical vector based on similarity among words in a large number of the corpus, then the embedded numeric vectors can be further incorporated into different types of statistical models and machine learning models. Considering the limitation on the number of words that can be included in the tweet, the cleaned tweets are mostly short in our example, so the Word2vec algorithm, which is a neural network, in essence, is used to create the word embedding of 100 dimensions for each word. Then the embedded vectors are aggregated on the tweet level by averaging over all embedded vectors of the words in every single tweet.

After the word embedding, the aggregated 100-dimensional embedded vectors for each tweet are combined with the six D-1 Day Bitcoin price features mentioned in section 5.2.1 to feed into the linear model and the Gradient Boosting model. The same training and testing split as that in section 5.2.1 is used to train and evaluate the model performances.

5.2.3 Model Evaluation

As stated at the beginning of section 5.2, four commonly used model evaluation criteria are implemented to compare the model performances based on the testing set: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R square, results are shown in Table 1.

	The model with Sentiment Score		The model with Embedded Tweets	
	Linear Model	Gradient Boosting Model	Linear Model	Gradient Boosting Model
R square	0.80	0.93	0.79	0.93
Mean Absolute Error	227.39	14.64	223.77	14.53
Root Mean Square Error	257.68	148.24	262.83	155.41

Table 1. The evaluation results with different models.

It is revealed that compared with the baseline linear model, the Gradient Boosting model improved the model performance with respect to all three criteria, and across two types of models. Meanwhile, it can also be found that the Word2vec word embedding did not improve the model performances as expected, however, slightly decreased the criteria, which is potentially due to the fact that the Word2vec creates the embedding only based on our own text which is mainly corpus with small size.

5.2.4 *Deep Learning Model with Embedded Tweets*

Instead of using the commonly used models to model the Bitcoin prices which ignore the sequencing of the time series data. The Bitcoin price time series data can be directly incorporated into the model through the deep learning models, such as Recurrent Neural Networks (RNN), Long short-term memory (LSTM), (Gated Recurrent Units) GRU, Bidirectional Encoder Representations from Transformers (BERT), etc. However, what differs the deep learning models from the models mentioned above is that the deep learning models inherently deal with sequence data, which requires the data to be aggregated to the daily level. So, the tweet sentiment scores used in section 5.2.1 are further aggregated into the daily levels. In our example, a total of 312 consecutive days' data are collected from 4/6/2018 to 5/27/2019, where the first 252 days are used as the training set and the last 60 days are used as the test set. Then the Long short-term memory (LSTM) model is implemented for the training set and predicts the "future" price for the last 63 days. The model is trained with Tensorflow's Keras through Python due to the small sample size.

Figure 13 presents the comparison between the predicted and the true time series of the last 60 days' Bitcoin prices. The model picks a window of 3, which means the previous three days' data are used to predict the current day's closing price. And only after 100 epochs, the LSTM can significantly capture the trend of the changes in the Bitcoin price with 2,601 parameters.

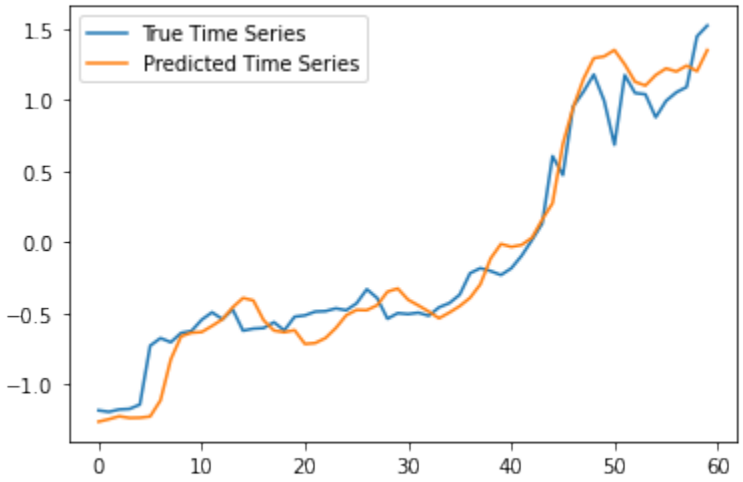

Figure 13. Predicted times series and the real time series of the "future" Bitcoin prices.

6 Conclusion

In this project, we used widely acknowledged tools and techniques used in Big Data Processing like Spark ML for modeling, Spark SQL for preprocessing, and Tweepy for Streaming. Sentimental Analysis is not a very difficult task if we use Libraries and take care of filtering relevant things, but when we couple this with price prediction it becomes a challenge. Based on the comparison between the models using the sentiment score and the models using the word embedding, it can be found that the Word2vec did not provide more information compared with the sentiment score on prediction. This is potentially due to the size limit of each tweet and the preprocessing. An alternative to this could be the GloVe word representation which is based on a large number of the corpus collected from tweets. In addition, the LSTM model presented good generalization performance on predicting "future" Bitcoin prices even with the previous three days' data and a relatively small number of epochs of training. For applications like these, the choice of sentimental analyzers, the quality of the dataset, the quality of preprocessing, and the choice of models define its overall quality and modelling efficiency.

7 Challenges

In this section, we will talk about the challenges we encountered while implementing this project

1. Tweets fetched via Streaming are not reliable and can not be directly fed to the analyzer. They need to be saved in HDFS/ parquet/ CSV or other formats for moving on with the next steps like preprocessing and Sentimental Analysis. The streaming must always be consistent and sometimes the streamed tweets are found to be irrelevant in nature (spams, advertisements, etc). To make sure this does not hamper our Machine Learning Models' efficiency, we have additionally used the Kaggle Dataset of Bitcoin Tweets(A total of about 16M tweets) which are then passed through our spark SQL based preprocessing functions and sentimental analysis libraries. This ensures our overall architecture is robust and we always have a Plan B.
2. Translation of tweets from the Kaggle Dataset of Bitcoin Tweets is not feasible due to connectivity issues when using translation libraries on large amounts of data. Processing more than 100000 tweets without a dedicated Spark cluster is impractical because it takes way too long to preprocess all of them (1 million tweets takes about an entire day to preprocess).
3. One of the challenges in sentiment analysis is that we need to first encode the CSV file with "utf-8". Moreover, when setting the input data to Textblob and Vader, the data type cannot be series. Similarly, the datatypes of outputs with both tools are different, so we need to be aware of this part to prevent the errors from happening. To be more specific, we encountered a TypeError that the 'text' argument passed to ' '__init__(text)" must be a string. However, when we check the datatype of 'text', it shows that it is a string. Eventually, we found that it might be because of the wrong way of adding values to the list. Although we are stuck in this part for a while, we finally solve the problem with "append()" in order to prevent changing the datatype of the string and produce the result and the plots.

REFERENCES

[1] A. A. Ariyo, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 106-112.

[2] Gupta, R., & Chen, M. (2020, August). Sentiment Analysis for Stock Price Prediction. In 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp. 213-218). IEEE.

[3] Hazarika, D., Konwar, G., Deb, S., & Bora, D. J. (2020). Sentiment Analysis on Twitter by Using TextBlob for Natural Language Processing. In *ICRMAT* (pp. 63-67).

[4] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[5] Kharde, V., & Sonawane, P. (2016). Sentiment analysis of Twitter data: a survey of techniques. arXiv preprint arXiv:1601.06971.

[6] Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal, 5(4), 1093-1113.

[7] Mittal, A., & Goel, A. (2012). Stock prediction using Twitter sentiment analysis. Standford University, 15.

[8] Z. Liu, Z. Dang and J. Yu, "Stock Price Prediction Model Based on RBF-SVM Algorithm," 2020 International Conference on Computer Engineering and Intelligent Control (ICCEIC), 2020, pp. 124-127

[9] Wu, D. D., Zheng, L., & Olson, D. L. (2014). A decision support approach for online stock forum sentiment analysis. IEEE transactions on systems, man, and cybernetics: systems, 44(8), 1077-1087.

[10] Ko, C. R., & Chang, H. T. (2021). LSTM-based sentiment analysis for stock price forecast. PeerJ Computer Science, 7, 408.

[11] Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., & Anastasiu, D. C. (2019, April). Stock price prediction using news sentiment analysis. In 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataSer- vice) (pp. 205-208). IEEE.

[12] Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., Iotti, E., Magliani, F., & Mani- cardi, S. (2016, September). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter. In KDWeb.