TWITTER SENTIMENT ANALYSIS FOR CRYPTO CURRENCIES

ARSHAD MANAMAKKAVIL

219033472

September 2022

Dissertation for MSc Data Analysis for Business Intelligence

University of Leicester

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES:

.

DECLARATION

All of the sentences or passages in this project that were taken from other people's work were cross-referenced to the author, work, and page. I understand that if I do not do this, it will be seen as plagiarism and will be a reason for me to fail this module and the degree exam.

Name: Arshad Manamakkavil

Signed: Arshad Manamakkavil

Date: 26/08/2022

ACKNOWLEDGEMENT

Completing this dissertation would not be fulfilled without the assistance of the following persons. First and foremost, I'd like to express my gratitude to my DABI program director Dr Andrey Morozov and my academic supervisor Dr Jason Semeraro for all the helpful input and direction they provided during this project.

Second, I want to thank my industrial supervisor Seckin Tataroglu (CEO of Lambda BI), for his aid with each week's review. As a result, I developed new ideas for methods and achieved the required results in each phase. Also i would like to thank my team members Subash Kasyap Yeleti, Poojitha Paleti, and Rajeswari Kamani, for their assistance with this project.

ABSTRACT:

The internet is becoming a hub of idea sharing, expressing opinions, broadcasting, and global messaging. Social network services like Twitter, Google+, Instagram, and Facebook play a significant part in this. The development and growth of online technologies create a significant amount of data and are made available to internet users. For example, Twitter is a website where users can share their thoughts through short posts. Many cryptocurrency traders rely on Twitter tweets to help them make daily trading decisions. Several sentiment analysis applications can be developed based on the information provided by these tweets, such as reviews, elections, and marketing. Sentiment analysis is a method of taking information from many data and categorizing it into several groups, known as Sentiments. These sentiments are either positive, negative, or neutral.

The dissertation topic, “Sentiment analysis on cryptocurrency”, primarily focuses on sentiment analysis using Natural Language Processing for a selected cryptocurrency (Ethereum). The Twitter API (application program interfaces) is a platform where users can access the information found in tweets, like hashtags, and we Can use the Twitter API to obtain all currently available tweets. The sentiment analysis takes the Ethereum influencers based on their tweets. First, influential Twitter users are chosen and gathered their tweets based on their hashtags using Twitter API. The tweets were then filtered to exclude any mentions of anything other than Ethereum. Later these datasets are examined by considering how these tweets affect the value of the daily Ethereum market price.

Python is used to perform data cleaning and analysis. In addition, the model was built using two methods, such as LSTM (Long Short-Term Memory) and random forest regression. Python's NLTK (Natural Language Toolkit) package is a foundation for creating applications and categorizing data. In addition to supplying example data to train and test different classifiers, supplies graphical demonstrations for different results or patterns. We trained and tested an LSTM model classifier on the labelled data, and the test set accuracy was 82%.

For optimal results and to compare many models, the bidirectional LSTM model was also constructed, which delivers 84% accuracy, and the Random Forest regressor model, which offers around 86% accuracy.

Keywords: Natural Language Processing, Sentiment analysis, LSTM, Bidirectional LSTM, NLTK.

1 INTRODUCTION

A cryptocurrency is a mechanism for generating virtual "coins" and securing their ownership and transactions through a cryptographic problem. Lambda BI (Business Intelligence) Ltd is a Business Intelligence company that offers big data management, data analysis, AI (Artificial Intelligence) modeling, and prototype production by the technology areas that are expected to grow in demand by businesses. The new monetary and financial systems have a ground-breaking strategy like Blockchain. Undoubtedly, Bitcoin was the most significant step toward achieving this goal. By 2017, most of the community was trading Bitcoin and other alternative currencies due to rising Bitcoin demand. (Franco Valencia, Alfonso Gómez-Espinosa, & Benjamín Valdés-Aguirre, 2019)

The Sutardja Center for Entrepreneurship and Technology Technical Report In 2008, a person or group using the alias Satoshi Nakamoto released a paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System." This study presented a peer-to-peer form of electronic currency allowing internet payments to be transmitted directly from one party to another, bypassing banking institutions. Bitcoin initially implemented this notion. Currently, cryptocurrency refers to any networks and mediums of exchange that use encryption to safeguard transactions, as opposed to systems where transactions are routed via a centralized trusted institution. (Bahrawi, 2019)

A blockchain is a distributed database or public ledger of all completed transactions or digital events that is shared by all parties involved (“A review on blockchain applications in the agri-food sector”). The consensus of most system users confirms each transaction on the public ledger. Additionally, once data has been input, it cannot be erased. Every transaction that has ever occurred is recorded in a unique, verifiable block on the blockchain. The most well-known use of blockchain technology is Bitcoin, which is a decentralised, peer-to-peer form of digital money. (“Shubham Ghodke - Graduate Engineer Trainee - Compucom | LinkedIn”) Bitcoin is a very controversial digital currency, but the underlying blockchain technology has performed flawlessly and offers various financial and non-financial applications.

The entire operation of a blockchain are determined as in the figure below.

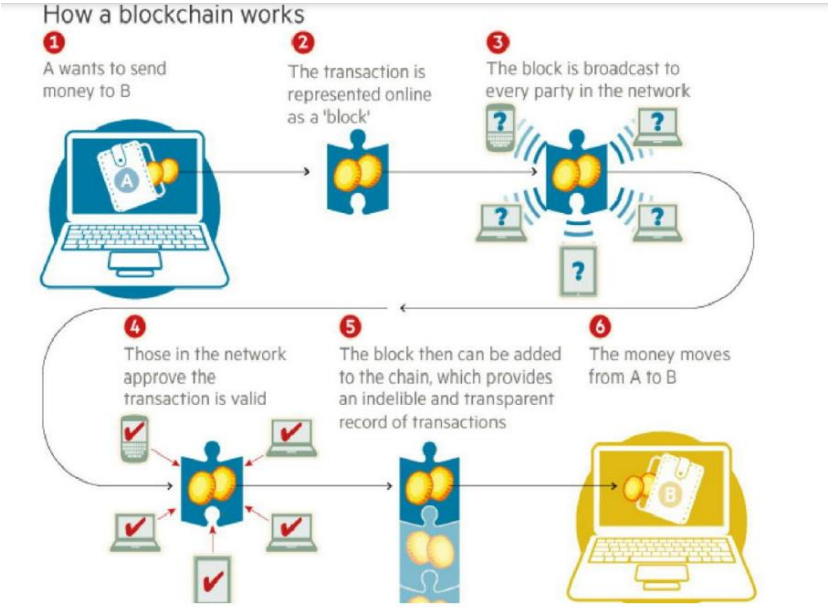


Figure 1 Transaction using block chain

When trying to understand a foreign language, we always begin by finding the meaning of words. The more our knowledge, the better we will understand a new text. When we discover unfamiliar terms in a new text, we disregard them and concentrate on the ones we already know (Michael Crosby, October 16, 2015). Then, we must decide its essence; this is where grammar rules and context come into play. *Sentiment analysis* is often seen by those who engage in data science. Additionally, sentiment analysis is derived by examining diverse data from views or opinions to draw conclusions based on various viewpoints. A proportion of positive, negative, or neutral sentiment may be the outcome of sentiment analysis (Wilson, J. Wiebe, , & P. Hoffmann, Oct 2005). Due to the various aspects and uncertainties that interplay in the markets, including economic and political situations and human behavior, sentiment analysis may be tricky. Sentiment analysis is one of the most critical jobs in Natural Language Processing (NLP), intending to figure out a given text's positive, negative, or neutral sentiment (Batra & D. Rao).

Social media like Twitter are extensively utilized by society nowadays. Twitter's premise is easy and quick since the messages are brief. Therefore, traders frequently use Twitter as a social media in natural language processing (NLP). Moreover, Twitter offers an API that makes it simple for researchers to get the data. Compared to other markets, the financial viability of the cryptocurrency market has been demonstrated. In addition, the algorithms in other domains are confirmed behind the operation of cryptocurrencies. As a result, the Ethereum market seems to run independently of other financial markets. Part of the attractiveness of this market stems from the fact that the technology used to mine cryptocurrencies offers a practical alternative to more conventional commodities such as gold. As a result,

predicting market prices often is challenging but possible. According to several studies, market price fluctuations are not random but show highly nonlinear and dynamic behavior. Earlier research has also shown that it is not needed to estimate the future price to receive help from financial forecasting accurately. In truth, forecasting the market's direction compared to its value may lead to more significant gains. (Deebadi, 2020)

2 BACKGROUND OF THE PROJECT

Machine learning has been used to predict the stock market during the last few decades. The most popular approaches are neural networks (NNs), support vector machines (SVMs), and random forests (RFs). For example, deep learning algorithms derived from NNs have been used to estimate the price of Bitcoin, and Ethereum, and recurrent neural networks have been used to forecast the market direction of change for the NASDAQ composite index as per the studies. (Bekiros, Stelios, & Georgoutsos, Dimitris.;, 2008).

The sentiment in social networks, specifically Twitter, may be used to forecast the movement of stock indexes. While there is little evidence that forecasts based on sentiment generate large profits in stock trading, research developed a trading strategy for the Bitcoin cryptocurrency based on tweets. In addition, another study enhanced the quantity of research on other cryptocurrencies and offered a method employing dynamic analysis to expect changes in the values of Bitcoin, Ethereum, and Ripple. We will assess the evaluation outcomes of the algorithm we employ in this study, which sentiment analyses Twitter data using the Random Forest algorithm and LSTM model based on earlier studies.

Twitter was found in March 2006 and debuted it in July of the same year. One hundred million users sent an average of 340 million tweets per day, with an average of 1.6 billion daily search queries as of 2012.

The Twitter Sentiment Analysis is a classification problem that seeks to identify tweets as positive or negative based on the opinions they convey (Crestani, Jun. 2016.). Pre-processing is the first stage in exploring sentiment analysis. This may be achieved by data cleansing, data evaluation, and data reduction mainly to convert the raw text data to a machine understand format (J. Akilandeswari & G. Jothi, 2018, pp. 426–433).

Sentiment aspects are associated with the positive and negative words and feelings that may be retrieved using various features (M. Bouazizi & T. Ohtsuki, , 2016.). For example, syntax-based characteristics indicate a relationship between a question mark, an exclamation mark, parentheses, and quote marks and their count in phrases. We can also have the semantic aspects, which focus on the reasoning behind

the phrases, such as passive and active forms. The top words feature also extracts words with many occurrences in a text to extract emotion patterns.

Random Forest is a collection of decision tree algorithms, mainly for classification and regression. More trees, in general, equate to improved performance and efficiency when applying this method. Here, several approaches can be used to extract a sample set of data points from a specified training set. First, create a decision tree based on the results of the previous phase. Then, we will receive the number of trees if every tree built casts a vote for a data point. Finally, determine the decision tree classifiers' majorities (Breiman L, 2001).

Derbentsev simulated the short-term dynamics of the three cryptocurrencies with the highest market capitalization, i.e., Bitcoin, Ethereum, and Ripple, using several advanced prediction models. Specifically, they assessed the predictive accuracy of an ANN (artificial neural network), an RF (random forest), and a BART (binary autoregressive tree) model (Derbentsev, Matviychuk, & Soloviev, 2020). The data yielded 1583 daily Bitcoin values from August 1, 2015, to December 31, 2019. The average accuracy of the ANN and BART was much greater than the "naive" model, according to their experimental findings.

(Pintelas, Livieris, Stavroyiannis, Kotsilieris, & Pintelas, 2020) did intriguing research assessing advanced deep learning models for forecasting the values and movements of cryptocurrencies. Their study demonstrated the substantial limits of deep learning models when generating reliable predictions. Based on their experimental study, the authors emphasized the need to adopt more sophisticated algorithmic techniques to create efficient and accurate cryptocurrency models. Additionally, (Livieris, Pintelas, Stavroyiannis, & Pintelas, 2020) examined enhancing the forecasting performance and dependability of deep learning models using three frequently used ensemble techniques, averaging, bagging, and stacking. The authors used hourly Bitcoin, Ethereum, and Ripple values from January 1, 2018, to August 31, 2019. In addition, they thoroughly studied the performance of diverse ensemble models using multiple Convolution-based and LSTM-based learners as base models. Their investigation revealed that deep learning and ensemble learning could be easily applied to create robust and trustworthy prediction models for cryptocurrencies, although at a high computational cost. Finally, the study concluded that the hybrid method for cryptocurrency prediction that focuses on Litecoin and Monero suggested that the model is based on an LSTM and GRU-layered recurrent neural network architecture (Patel, Tanwar, Gupta, & Kumar, 2020).

3 REQUIREMENTS OF THE PROJECT

Several firms supply Bitcoin trading advisory services. However, most of them add their opinions to the analysis, making it subjective. The project's stakeholders Lambda BI Ltd.'s objective is to develop a deep learning model capable of collecting new tweets, analysing existing sentiment, finding current trends, and producing signals to forecast impending market movements.

Reviewing the Blockchain architecture and Bitcoin infrastructure is one of the Customer Requirements for the project activities. First, decide the most prominent Twitter accounts for Bitcoin news during currency volatility. Next, review sentiment analysis deep learning models such as RFR, LSTM, and BERT after collecting the tweets from the accounts. Later, apply the suitable deep learning model and evaluate the sentiment output.

Here, we gather tweets using Twitter's API to do this. First, we must determine how tweets are gathered based on influencers, here each tweet must be verified for appropriate cleaning processes. Next, models are constructed to establish a precise framework for analysing data distribution throughout the system. Next, the model's creation must provide conclusions about the optimal model. With these results, we must develop a future project scope and determine how to make the project more engaging. We must consider the initiative contributing to this project to the present market. Moreover, we must determine where this initiative stands from a commercial viewpoint. Here, it is necessary to consider the future perspective of the models and how to implement these models in a future project. Alternatively, this study is applied to real-world projects where a general audience can view the outcomes.

4 AIMS

The dissertation's primary aim is to set up a correlation between influencer tweets and the impact of each tweet on the Ethereum market price. The impact of each tweet on the market price is analyzed, and logical conclusions are drawn from these findings.

We chose the influencers depending on their popularity. Various sources of data preprocessing have been performed, and we have done some feature engineering. Also, the project should provide a defined conclusion. Our goal is to build a project that will provide effective results. A dependable source of information must be provided for the project, and numerous feature extraction approaches must be used. Model development must occur within the existing space of outcomes. To construct a very dependable system without affecting the existing model, we must provide specific outcomes that can be updated in the future.

5 LITERATURE REVIEW

The literature review of the project supplies a comprehensive overview of the relevant details supporting the topic, Sentiment analysis and a Brief explanation of how sentiment research is implemented in the cryptocurrency market. Essential components of various methodologies for data visualisation and exploratory analysis are described, followed by a discussion of the data model and visualisation. In addition, several cleaning operations and their execution are described. Finally, the implementation of several approaches, such as machine learning and the deep learning algorithm, is described.

5.1 NATURAL LANGUAGE PROCESSING (NLP) FOR TWITTER

Sentiment analysis of a particular topic is a rapidly growing field of Natural Language Processing, with research spanning from document-level categorisation to learning the polarity of words and expressions. Given Twitter's character constraint, finding the emotion of tweets most closely resembles sentiment analysis at the sentence level. However, the informal and specialised language used in tweets and the microblogging domain make twitter sentiment analysis an entirely different undertaking. Collection of all words in NLP is known as corpus.

The computer is unable to grasp every language the way humans do. Therefore, there must be a complete conversion into a system-understanding format. Natural language processing (NLP) can convert words to vectors, or "word vectorization." Language models are used to map words into vector space. For example, a vector of real numbers represents each word in a vector space. It also permits equivalent representations for words with similar meanings. Several feature extraction techniques must be used to convert a complete sentence to vectors. All text must be transformed into d-dimension vectors to be understood by the system.

Why do convert text to vectors.

In order to execute machine learning on text, we must convert our texts into vector representations so that numeric machine learning may be applied. This procedure is known as feature extraction or, more simply, vectorization, and it is a crucial first step.

5.2 SENTIMENT CLASSIFICATION

Vectorisation can be done by taking some food review as example here the d – dimension representation of reviews can draw a hyper plain and point all the positive and negative reviews placed in both side of the plane. Vectorization can be performed by taking some reviews as an example. Here the d-dimensional

representation of reviews can be used to create a hyperplane and place all good and bad reviews on opposite sides of the plane.

5.2.1 Text pre-processing and Feature Extraction:

We can take W as the plane taken from the corpus of review. It is a plane portion were all the reviews are placed as corpus and W is the portion of plane parallel to the reviews. Taken W as the normal plain, one side of the plane is all positive and other side is entirely negative. Let assume some point in positive and negative can be termed as x_i. Here W is the normal to the plane. If the possible W can be found on the plane the equation can be written as:

$W^t x_i > 0$ when the review is positive,

$$W \text{ is normal to plain and } x_i \text{ is the point taken on the d\_dimension}$$

$W^t x_i < 0$ then the review is negative. These are the simple linear algebra example for how to convert the text into vectors.

Studied the vectorisation by taking some reviews as example. Suppose we have 3 review represented as r_1, r_2 and r_3. The vectors corresponding to them are (v_1, v_2, v_3).

Consider the review r_1, r_2 if these reviews are similar in meaning in English and we can represent the similarity of each review as $similarity(r_1, r_2)$. But the r_1 and r_3 are not similar compared to r_1 and r_2.

The similarity of the reviews can be on any space of d-dimension vectors. This similarity can be termed as $Similarity\ (r_1, r_2) > similarity\ (r_1, r_3)$ then the length of the vectors d-dimesion always will be $Len\ (v_1 - v_2) < len\ (v_1 - v_3)$.

The whole process this equation is that if two reviews are similar in terms of their meaning they always will be closer in the vector dimension. If the similarity of words is not similar then it will farer in space. This length can be considered for the similarity of the tweets. We can conclude that the similar text must be closer geometrically as per the equation. This primary method for an NLP but after that the method is developed and used in different techniques likes BOW (Bag of Words), TF-IDF, word2vec, averageword2vec, TF-IDFword2vec.

Basic test pre-processing can be done by

1. Removing stopwords: The stop words are the predefined English words which are not taken into account while doing the vectorisation purpose. Stopwords can be imported by using NLTK library using nltk.corpus import stopwords.
2. For the next is to make every wording to lower case. Many pythons' library can be used for this. For this we can use ".islower()" from the python library.

3. Stemming: Stemming basically means taking the words and converting them into the root form. We can take an example like both tasty and tasteful are come from the base word called taste. Instead of separately vectorise them we can only the base word for conversion. Such an operation is called stemming. Stem is taken, porter stemmer, snowball stemmer libraries can be used for this purpose designed by NLP engineers.
4. Lemmatisation: Lemmatisation is done by breaking up the whole sentence into words. This is very language dependent so very prior consideration of exceptional words are need to be taken consideration before lemmatisation.

5.2.1.1 BOW Bag of words model:

A.A.A. Karim and R.A. Sameer (2018) mentions that Natural Language Processing (NLP) and Information Retrieval (IR) use the BOW model, a simplified representation. This model shows texts like sentences or documents as a bag of words. It only looks for duplicate words and does not care about grammar or the order of the words. The BOW model is primarily used in document classification methods, where the number of times or occurrences of each word is used to train a classifier [H.K. Kim and S. Cho (2017)]. The bag of words model is straightforward to comprehend and use. It is a technique for extracting text characteristics for use in machine learning algorithms. Problems such as NLP, information retrieval from documents, and document classifications have been solved with considerable success. The flow chart are given below

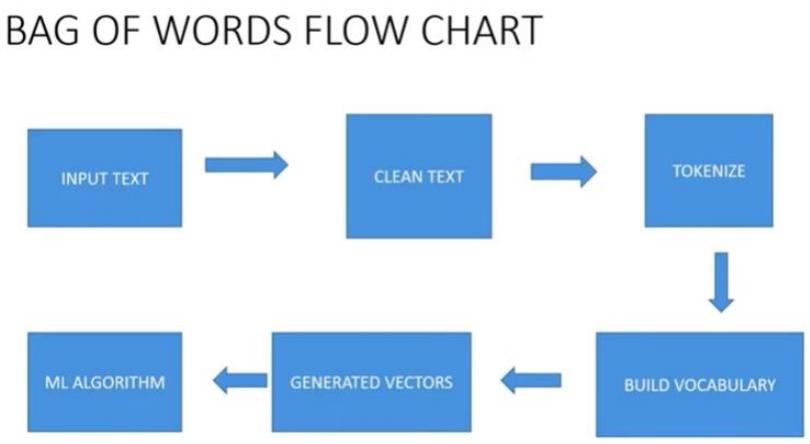


Figure 2 BOW flow chart

The bag-of-words (BoW) model is used to assess documents and text based on the number of word occurrences. The model does not take word order within a text into consideration. Semantic meaning of words is not considered in BOW. We can prevent this problem by using word2vec method. For Bag of Words there should be a separate value for each words but there is a chance for considering the negative and positive text as same on different sentences.

Consider an example of reviews below.

$r_1 \rightarrow$ "this burger is very tasty and affordable"

$r_2 \rightarrow$ "this burger is not tasty and is affordable"

$r_3 \rightarrow$ "this burger is delicious and cheap"

$r_4 \rightarrow$ "burger is good, and burger tastes good"

First, we transform each word into a dictionary format. Then, we eliminated the stopwords and the common phrases used in the English dictionary and considered general terms. We eliminate stopwords by utilising the Python NLTK package. These reviews include the stop words 'this,' 'is,' 'and,' 'not,' and 'are. After eliminating the stopwords from r_1 and r_2, the remaining words are

$r_1 \rightarrow$ {burger tasty affordable}

$r_2 \rightarrow$ {burger tasty affordable}

After deleting the stopwords, both reviews are considered similar. This issue occurs when the word "not" is used as a stopword in English. However, according to the semantic interpretation of the two reviews, these reviews have the opposite meaning. One is a positive review, and the other is a negative one, but the result is unsatisfactory because we only analyse one word at a time. Therefore, other strategies, which will be explained in greater detail in the future, are used to overcome these drawbacks.

5.2.1.2 Bi-grams:

Consider the words tasty and not tasty, for BOW after removing the 'not' which a stopwords both the text maybe misinterpreted as tasty but both the words are exactly the opposite to each other. This problem is resolved by considering the Pairs of consecutive words as vector. So here the same wording 'very tasty' and 'not tasty' are considered as pairs of words as different vectors. If we take three words into consideration instead of two words it will be called as Tri-grams.

5.2.1.3 N – grams:

N-grams are the method of taking the combination of words in a text. Here the n represents the no of words to taken. The disadvantages of BOW can be eliminated but the drawbacks of this will that if we take

the bi-grams then it creates more space than the unigrams we can say that bigram > uni-grams, trigram > bigram, so the for n-grams the dimensionality increases drastically.

5.2.1.4 TF: TERM FREQUENCY

Term frequency quantifies the frequency with which a term appears in a text. When we are vectorizing documents, we count the number of words in each line. In the worst-case scenario, if the term does not exist in the text, the TF value will be 0, and if all the terms in the document are identical, it will be 1. The final normalized TF value will fall within the interval 0 to 1. We can say that the Term frequency (TF) is how often a word appears in a document. The equation can be written as

$$TF(t) = \frac{Number\ of\ times\ term\ 't'\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

Similar to Term Frequency, Document Frequency assesses the relevance of a document within the corpus as a whole. TF is the frequency counter for a word t in document d, while DF is how often terms t occur in document N. *Term frequency* is the frequency with which w_i appears in r_j. The frequency of a phrase increases as its occurrences rise.

5.2.1.5 IDF: Inverse Document Frequency:

Inverse document frequency (IDF) measures the uniqueness or rarity of a document's occurrences of a specific term. As the name says, TF-IDF is the product of Term Frequency and the inverse frequency of documents.

$$IDF(t) = log(\frac{Total\ number\ of\ documents}{Numberof\ documents\ with\ term\ t\ in\ it})$$

Can also written as

$$TF\ (w_i, r_j) = \frac{no\ of\ times\ w_i\ occurs\ in\ r_j}{total\ no\ of\ words\ in\ r_j}$$

r_i Represents reviews, whereas *w* represents words. The value of Term Frequency is always between 0 and 1. It is done to retrieve information and also to locate instances of the query terms. The IDF equation can also be written as

$$IDF(w_i, DF) = \log\left(\frac{N}{ni}\right)$$

$$n_i - no\ of\ document\ which\ contain\ w_i,$$

$$no\ of\ documents.ni \leq N.$$

$$then\ IDF\ is\ always > 0.$$

$If\ n_i\ increases, then\ (\frac{N}{n_i})\ decreases$, we can also say that if word is more frequent then IDF is lower. IDF increases when n_i increases, it is clear the IDF is more means the word is rare.

5.2.1.6 TF-IDF ALGORITHM OR WORD FREQUENCY ALGORITHM

TF-IDF (Term Frequency Inverse Document Frequency) is a method for quantifying a word in texts; typically, we assign a weight to each word that reflects its significance in the document and corpus. This methodology is often used in information retrieval and text mining. Because we grasp the semantics of the words and the phrase, it is simple for us to comprehend the sentence. However, the computer can only comprehend information in numerical form. Therefore, we vectorise all languages so that the computer can better comprehend them.

$$TF(w_j, r_i) * IDF(w_j, df)$$

For the TF-ID, we assign more weight to uncommon terms in the corpus. Greater significance is given if a term is more frequent. Unfortunately, these approaches also disregard semantic terms, one of the model's flaws. Using the Word2Vec approach, anyone can overcome this constraint.

5.2.1.7 Word2vec:

For Word2vec the semantic meaning of words takes into consideration. It chooses a word and converts it to vector format. The vectors v_1 and v_2 will be closer if w_1 and w_2 are similar. We examine the association between the terms with comparable meanings. Male and female, nation and capital are some examples. W2v learns all the relationship from raw text. The main feature here is that the neighbourhood of words is compared. If word neighbourhoods are similar, then the vectors are also similar.

5.3 MODEL BUILDING:

5.3.1 Parameters for finding the summary table:

Different parameters are required to analyse the data for model building. These parameters are,

5.3.1.1 Accuracy:

This is the proportion of true positives and true negatives to true positives, true negatives, false positives, and false negatives.

$$Accuracy = \frac{True\ positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

5.3.1.2 Precision:

Precision refers to the ratio of projected positive observations to the total number of positive observations.

$$Precision = \frac{True\ Positive}{True\ positive + False\ Positive}$$

5.3.1.3 Recall:

The recall is the ratio of accurately anticipated positive observations to all observations in real class yes.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

5.3.1.4 F-score:

The weighted average of recall and accuracy is known as the f-score. When data have an unbalanced class distribution, precision is a less critical feature.

$$F_{score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

5.3.2 LSTM Model:

Long Short-Term Memory networks, often referred to as "LSTMs," are a specific kind of RNN that can learn long-term dependencies. They were presented by Hochreiter & Schmidhuber (1997) and improved and popularized by many authors in subsequent years. 1 They are currently frequently utilised and perform very well on various difficulties. General diagram of LSTM model is given below.

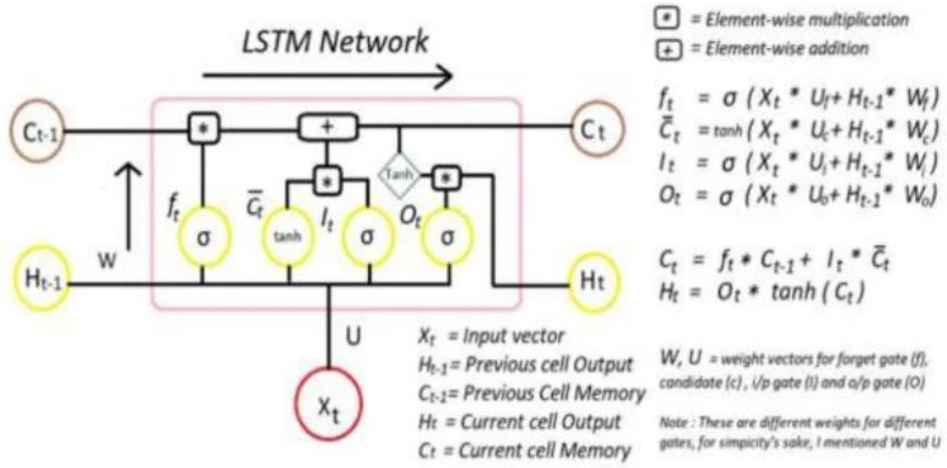


Figure 3 : LSTM model

LSTMs are purposefully intended to overcome the issue of long-term reliance. Long-term knowledge retention is their default habit, not something they must work hard to acquire. All recurrent neural networks consist of a series of neural network modules that repeat. This repeating module in ordinary RNNs will have a relatively basic structure.

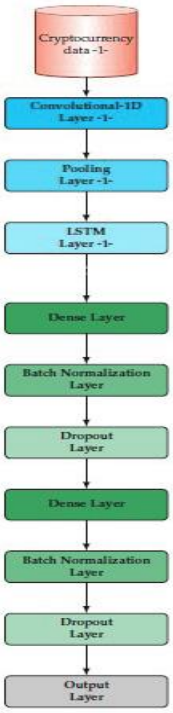


Figure 4 Architecture of LSTM model

5.3.2.1 Different layers of LSTM model

1. Convolutional layer: These are a distinct type of neural network layers distinguished by their capacity to learn the internal representation of their inputs. This is accomplished by performing convolutional operations to the input data and producing new feature values using convolution kernels, often known as "filters". (Goodfellow, Bengio, Courville, & Bengio, 2016)

2. Pooling layers minimise spatial dimensions and reduce the number of operations necessary for all subsequent layers. Less spatial information corresponds to fewer weights, which reduces the likelihood of overfitting the training data and reduces the computational cost. Probably the most popular options are max pooling and average pooling layers.

3. LSTM layers are recurrent neural network layers equipped with a distinct memory cell and adaptive gate units (input, forget, and output) for managing the information flow. Utilizing gates in each memory cell enables data to be filtered, deleted, or added, prolonging necessary information retention. (Demuth, Beale, De Jess, & Hagan, 2014.)

4. Dense layers are the most common option for constructing the hidden layer of a deep neural network. Batch normalisation is an elegant training technique for deep neural networks that focuses on stabilising the learning process by standardising the inputs of the next layer for each mini-batch.

5. Dropout layer: *Dropout* is a non-learnable layer added to a neural network model between existing layers. It temporarily sets a random set of outputs to zero with a predefined probability p, referred to as the dropout rate, which is then supplied to the subsequent layers. (Livieris, Stavroyiannis, Pintelas, Kotsilieris, & Pintelas, 2020)

5.3.3 Bidirectional LSTM:

Bidirectional LSTM is a recurrent neural network primarily used for natural language processing. Unlike regular LSTM, input travels both ways, and information from both sides may be used. Additionally, it is a potent instrument for modelling the sequential dependencies between words and sentences in both directions.

In short, BiLSTM adds an LSTM layer that reverses the flow of information. In short, it indicates that the input sequence flows in reverse in the extra LSTM layer. The outputs from both LSTM layers are then combined in various methods, such as averaging, summing, multiplying, or concatenating. An example is given below. (Zvornicanin, 2022)

5.3.4 Random Forest Classifier Model:

The Random Forest classifier is based on trees. Multiple classification trees are used to estimate the class label. Each tree will vote for a specific category label for a given data set, and the category label with the most votes will be assigned to each data point. The correlation between any two trees determines the error rate of this classifier. To reduce the error rate, the trees must be robust, and the level of associativity must be as low as possible. The internal nodes of the classifier tree represent the alternatives, the sides of a node represent tests on the weight of the feature, and the leaves represent category classes. It begins classification at the root node and progresses gradually downstream until a leaf node is identified. The document is then assigned the class that identifies the leaf node. Sentiment analysis using the Python tweepy package Using the Tweepy library, we collected daily tweets using selected hashtags.

6 METHODOLOGY AND RESULT

6.1 DATA COLLECTION:

We analyzed two distinct data sources in this project:

1) We obtained Twitter tweets for sentiment analysis using the Python tweepy package. Using the tweepy library, we collected daily tweets using selected hashtags.

2) We collected daily Ethereum prices and volumes from the Yahoo Finance website (Aug 07, 2015 - Sep 19, 2022).

6.1.1 Data from Twitter

The Twitter API as a platform where users can access the information found in tweets. For analysis reasons, these data can be used more ethically and securely. Access to the data should be restricted; however, these restrictions can be overcome by requesting elevated or academic access. By doing this, we may use the Twitter API to obtain all tweets that are currently available.

Tweepy is an open-source Python program that helps easy access to the Twitter API using Python. Tweepy has a collection of classes and methods that reflect Twitter's models and API endpoints, and it handles many implementation details transparently, including data encoding and decoding.

We collected the data using the Twitter API. Created the Twitter API account, which only gets access to limited tweets that will not be enough for this dissertation; these problems were resolved using elevated or academic access. Applied the Twitter API elevated access for retrieving the data from the Twitter API. We can access up to 2 million Tweets per month using the elevated access. This API helps to retrieve the data based on the tweets.

Twitter includes a developer area where a developer account can be requested. The dashboard for Twitter's API is currently using Twitter API v2. This developer account may be set up using our standard Twitter account, which is often used to tweet. A developer account is a unique environment designed specifically for collecting data from Twitter. Twitter API v2 is the most recent version and consists of three access windows. These areas are categorized mainly by the total number of tweets whose data we can scrape from Twitter. According to the tweets' access, each access window is categorized as follows:

a) Essential access:

This is free and instantaneous access after creating a Twitter developer account. There is no application needed to access, but there is only one permissible environment per project. Here, depending on our access, we can retrieve about 500,000 Tweets every month. These 500K tweets are accessible for free, but only for one project. This restricted access to tweets is insufficient for this project because we need to pull several tweets, verify their accuracy, and retrieve more data, even if the project requires us to do it later. So as to counter this restriction in retrieving tweets from Twitter, we may utilize an alternative access technique to retrieve additional data. There is elevated access and academic research access, but neither of them is predefined access that cannot be used immediately, like essential access.

b) Elevated access:

Elevated access cannot be accessed straightaway once a developer account has been created. This is not a free-to-use product like Essential Access, but we may request Elevated Access and receive an email confirming its availability. Here, the restriction of 500K tweets is increased to 2,000,000 tweets. Here, up to two million Twitter tweets are accessible. The Twitter developer account dashboard is seen below. As seen in the figure below, up to three project environments can access two million tweets every month. Considering the data requirements, Twitter's elevated access is more than sufficient for this project.

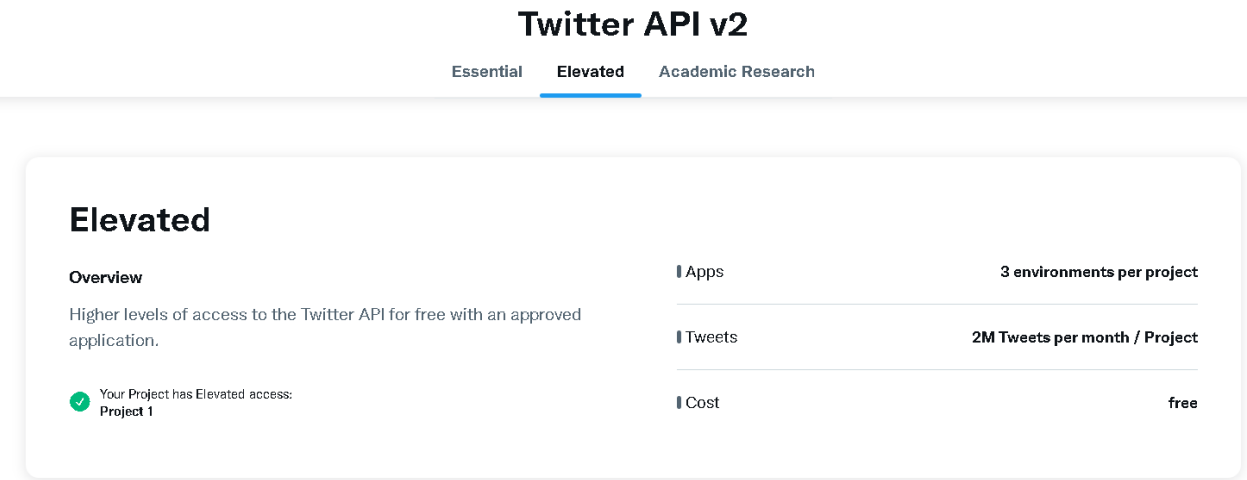


Figure 5 Twitter Elevated Access dashboard

Additionally, there is another access, which is

c) Academic Research:

Only academics with a research project requiring convolutional data can use this access window. Access is necessary, although examining Twitter's conversational data is essential or would be helpful. Access is at no cost. There is a requirement for an application, and we can have 10 million tweets monthly. This is solely for non-commercial use and requires a license to activate. For this project, though, elevated access is sufficient. In order to access all tweets from Twitter for a future study, we require research access.

6.1.2 Data from Yahoo finance:

To determine the influence of the Ethereum market, we must collect appropriate daily Ethereum pricing data. Yahoo Finance is the most acceptable source to download the daily opening and closing prices for Ethereum. In addition, several finance-related data sources are accessible for free download at yahoofinance.com. We retrieve Ethereum pricing information by specifying the period for which we require the data. Below is the raw data taken from Yahoo Finance between August 7, 2015, and September 19, 2022. Here the raw data from the yahoo finance is given below.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
1	2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2	2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
3	2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
4	2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984
...	...	...	...	...	...	...	...
1754	2022-08-29	1430.439453	1556.309570	1427.728394	1553.037354	1553.037354	17965837488
1755	2022-08-30	1553.188965	1600.461182	1480.831787	1523.838867	1523.838867	21835784470
1756	2022-08-31	1524.286499	1612.358887	1524.286499	1553.684937	1553.684937	20591680941
1757	2022-09-01	1553.756348	1593.082764	1520.188354	1586.176758	1586.176758	16434276817
1758	2022-09-02	1583.881592	1602.590088	1571.791260	1596.431030	1596.431030	16078590976

1759 rows × 7 columns

Figure 6: Ethereum price from yahoofinance.com

There are 1759 rows and seven columns in the raw data; the row corresponds to the number of days that we already selected earlier. The seven features are date, Open, High, Low, Close, Adj Close, and volume. All of the first data features extracted are unnecessary for the project. However, during the steps of data pre-processing, it must be cleaned.

6.2 SELECTING THE INFLUENCERS:

We can use many sources from the internet in order to get the best influencers out there which drives the ethereum market. This influencers either be a person or an organization. I have selected both into consideration whoever tweets are based on ethereum search.

The first step is to take data according to the influencers selected. Some of the best influencers who are popular on the internet and impact more on the Ethereum market based on the findings are given below.

1. **Vitalik Buterin** @VitalikButerin

Ethereum famous founder is the undisputed leader of Twitter's crypto influencer base. Vitalik posts often on the platform, to the enrichment of his over 2.1 million Twitter followers.

2. **Roger Ver** - @rogerkver

Known as "Bitcoin Jesus," He spent a significant amount of money in firms involved in the cryptocurrency industry, including Bitcoin, Ripple, Kraken, and Purse.io, and was one of the early users of blockchain technology.

3. **Joseph Schweitzer** @JBSchweitzer: Blockchain & Politics: Governance debates

4. **Ethereum** @ethereum : Official account of the Ethereum Foundation

5. **Anthony Di Iorio** @diiorioanthony : Anthony Di Iorio is a Canadian businessman best known for co-founding Ethereum and investing in Bitcoin at an early stage. As the Co-Founder of Ethereum, Anthony holds a lot of power within the Ethereum community.

6. **Chris Burniske** @cburniske : The co-author of the book Cryptoassets, Chris Burniske, is also a co-founder of Placeholder, a New York venture capital company that focuses on cryptoassets.

7. **Camila Russo** @CamiRusso : The creator and CEO of The Defiant, a decentralised finance-focused knowledge portal. The Infinite Machine, the first book about the history of Ethereum, was released in July 2020 by HarperCollins. 8. Stephan Taul @stephantual

These are the influencers selected initially, but after analysing the data, we need to remove specific influencers due to their infrequent tweets on Ethereum. So we added some new influencers. Below is a list of the other influencers.

Robert Ekanem '@Ekpenyongodusu',: Entrepreneur, Professional in the Networking Industry, Cryptocurrency Investor, and Online Researcher.

'@InsiderEthereum', : Tweets about the latest Ethereum and Crypto currency news

'@CoinFees' , : Cryptocurrency fee estimates for Bitcoin, Etherum and more.

'@Cyber_FM' : Free radio about crypto currency.

6.3 COLLECTING THE DATA USING TWEEPY:

After the prior selection of influencers, we must collect the necessary data. Here, a Python notebook like Jupyter Notebook or Google CoLab is used. Both of them are compatible with the project's requirements. If we take data from Twitter every time, the data we utilize for this project may become outdated. Additionally, accessing the keys and tokens in each step takes longer. We can divide the project's programming across many Ipython notebooks to increase the system's reliability.

```
#Import the libraries
import tweepy
from textblob import TextBlob
import pandas as pd
import numpy as np
import re
```

Data is imported using the tweepy package. The Twitter API key is used to get tweets from Twitter using the Twitter developer account's access token. Each month, we get access to two million tweets. Access token and secret token numbers are both secret codes that may be changed after each usage. The API was created using tweepy.API.

```
[ ] # Twitter API data Keys
    consumerKey = "[redacted]"
    consumerSecret = "[redacted]"
    accessToken = "[redacted]"
    accessTokenSecret = "[redacted]"

[ ] #Created authentication object
    authenticate = tweepy.OAuthHandler(consumerKey, consumerSecret)
    #Set access token and access token secret
    authenticate.set_access_token(accessToken, accessTokenSecret)
    #Create the API
    api = tweepy.API(authenticate, wait_on_rate_limit=True)
```

For the sake of efficiency, we generated a different ipython notebook for each distinct dataset. One ipython notebook is used for data collecting, while the other is for data analysis. Doing so can prevent repeatedly collecting the same data from Twitter, making the project easy and effective.

```
number_of_tweets = 20000
users = []
tweets = []
likes = []
time = []
retweets = []
names = []
user_id = '@VitalikButerin', '@rogerkver', '@ethereum', '@JBSchweitzer', '@diiorioanthony', '@cburniske', '@CamiRusso', '@Ekpenyongodusu', '@InsiderEthereum', '@CoinFees'
for j in range(11):
  for i in tweepy.Cursor(api.user_timeline, q = "#ethereum -filer:retweets" ,id=user_id[j], tweet_mode='extended').items(number_of_tweets):
    users.append(i.id)
    names.append(user_id[j])
    tweets.append(i.full_text)
    likes.append(i.favorite_count)
    time.append(i.created_at)
    retweets.append(i.retweet_count)
```

```
[ ] eth_df = pd.DataFrame({'User':names, 'tweets':tweets,'likes':likes,'time':time, 'retweets':retweets })
    eth_df
```

Importing Tweets involves setting a restriction on the number of tweets per user. Here, the number of tweets variable is specified to a maximum of 20,000 tweets per user. Then, we generate an empty list of users, tweets, likes, time, and retweets. This is the essential data needed for the analysis. Another important feature is to extract tweets depending on the users selected. Create a variable named user id to do so. Here, the user id is distributed to the previously chosen influencer. Then, each influencer is picked based on the name of their Twitter account, as seen below.

```
user_id = '@VitalikButerin', '@rogerkver', '@ethereum', '@JBSchweitzer', '@diiorioanthony', '@cburniske', '@CamiRusso', '@Ekpenyongodusu', '@InsiderEthereum', '@CoinFees' , '@Cyber_FM'
```

Here all the list of data we scaped from the internet is converted into dataframe by using panda's library pd.DataFrame. This creates a dataframe which is defined as below.

```
'User':names, 'tweets':tweets,'likes':likes,'time':time, 'retweets':retweets
```

Here all the names are printed as column name.

The figure below shows the data frame. The name of the data frame which I use throughout the project will be eth_df.

```
eth_df = pd.DataFrame({'User':names, 'tweets':tweets,'likes':likes,'time':time, 'retweets':retweets })
eth_df
```

	User	tweets	likes	time	retweets
0	@VitalikButerin	@lexfridman 🤩 \n\nhttps://t.co/JP1Ucky8h6	442	2022-08-23 20:58:21	21
1	@VitalikButerin	If I had to make a bear case for Anglo civiliz...	2378	2022-08-23 15:23:31	250
2	@VitalikButerin	RT @VitalikButerin: @thefaketomato @palladiumm...	0	2022-08-23 15:18:40	50
3	@VitalikButerin	@thefaketomato @palladiummag "X sounds great, ...	367	2022-08-23 15:16:45	50
4	@VitalikButerin	@Max_Stirn @palladiummag Many kinds of technol...	15	2022-08-23 13:56:50	3
...	...	...	...	...	...
29329	@Cyber_FM	Welcome 2021! https://t.co/jS8vCVRX5S GET PAID...	0	2021-02-22 14:30:35	0
29330	@Cyber_FM	Positivity for 2021 from https://t.co/jS8vCVRX...	1	2021-02-22 14:30:34	0
29331	@Cyber_FM	Ahora sonando en CyberFM Latino: Manantial de ...	1	2021-02-22 14:26:24	0
29332	@Cyber_FM	An amazing 2021 w/ https://t.co/jS8vCVRX5S GET...	1	2021-02-22 14:20:27	0
29333	@Cyber_FM	Kick off 2021 w/ https://t.co/jS8vCVRX5S GET P...	0	2021-02-22 14:05:17	0

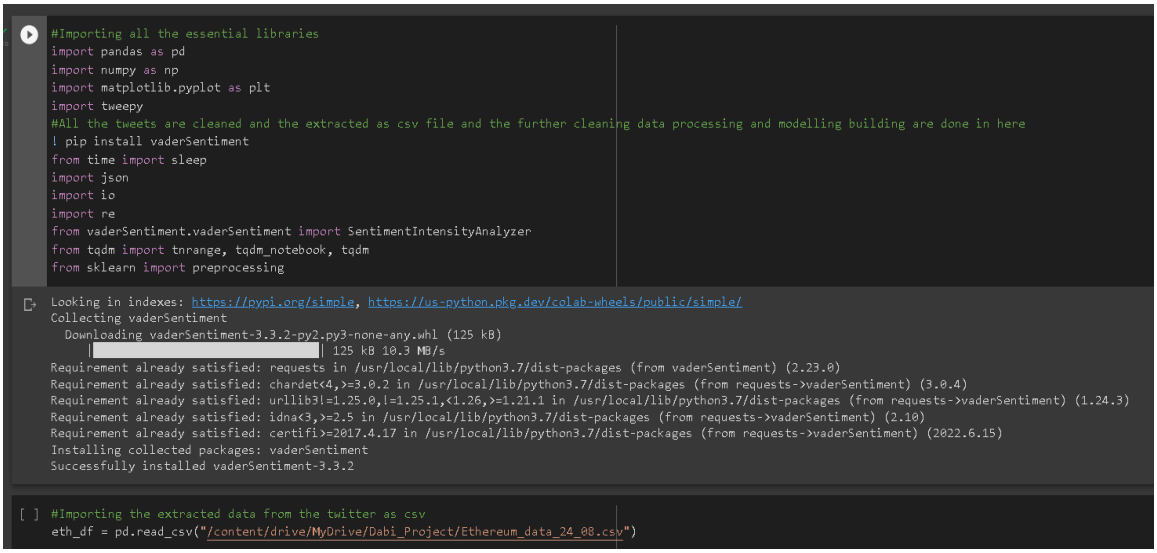
29334 rows × 5 columns

Table 1 Twitter raw data

```
eth_df.to_csv("Ethereum_data_24_08.csv")
```

The created data frame has 29334 rows and five attributes. Tweets, likes, time, and retweets align with the user. The extracted data are transformed into a CSV file that will be analysed in a separate ipython notebook.

The exported csv file having all the information from the tweets is cleaned and further exploratory data analysis is made. Created a new google colab notebook for the further analysis.



```
#Importing all the essential libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tweepy
#All the tweets are cleaned and the extracted as csv file and the further cleaning data processing and modelling building are done in here
! pip install vaderSentiment
from time import sleep
import json
import io
import re
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tqdm import tnrange, tqdm_notebook, tqdm
from sklearn import preprocessing
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     |████████████████████████████████| 125 kB 10.3 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from vaderSentiment) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->vaderSentiment) (2022.6.15)
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

```
#Importing the extracted data from the twitter as csv
eth_df = pd.read_csv("/content/drive/MyDrive/Dabi_Project/Ethereum_data_24_08.csv")
```

All libraries for data cleaning, preprocessing, and model construction are imported here. All the essential libraries are imported along with the csv file as the dataframe eth_df which is imported using the panda's library.

All of the raw data are taken from Twitter, as described previously. These raw data must be cleansed before they can be used for analysis. After cleaning, the primary purpose of this project is to do Exploratory Data Analysis (EDA). As per the preceding steps, all Twitter API data extraction is complete, and the raw data are converted to a CSV file. This CSV file contains all the data necessary for analysing the entire project. The primary advantage of doing things individually is avoiding several Twitter data pulls. If we retrieve the data each time we work on the project, the project will require additional time.

6.4 DATA PRE-PROCESSING:

The data extracted from tweets using tweepy contains time information that cannot be used in subsequent data extraction and preparation phases. Therefore, the first step is to extract the date from the time column. We may obtain the date using the Panda's library function, pd.todatetime. Below, we can see the date after its format has been altered.

	User	tweets	likes	time	retweets	Date
0	@VitalikButerin	@lexfridman 😳\n\nhttps://t.co/JP1Ucky8h6	442	2022-08-23 20:58:21	21	2022-08-23
1	@VitalikButerin	If I had to make a bear case for Anglo civiliz...	2378	2022-08-23 15:23:31	250	2022-08-23
2	@VitalikButerin	RT @VitalikButerin: @thefaketomato @palladiumm...	0	2022-08-23 15:18:40	50	2022-08-23
3	@VitalikButerin	@thefaketomato @palladiummag "X sounds great, ...	367	2022-08-23 15:16:45	50	2022-08-23
4	@VitalikButerin	@Max_Stirn @palladiummag Many kinds of technol...	15	2022-08-23 13:56:50	3	2022-08-23
...	...	...	...	...	...	...
29329	@Cyber_FM	Welcome 2021! https://t.co/jS8vCVRX5S GET PAID...	0	2021-02-22 14:30:35	0	2021-02-22
29330	@Cyber_FM	Positivity for 2021 from https://t.co/jS8vCVRX...	1	2021-02-22 14:30:34	0	2021-02-22
29331	@Cyber_FM	Ahora sonando en CyberFM Latino: Manantial de ...	1	2021-02-22 14:26:24	0	2021-02-22
29332	@Cyber_FM	An amazing 2021 w/ https://t.co/jS8vCVRX5S GET...	1	2021-02-22 14:20:27	0	2021-02-22
29333	@Cyber_FM	Kick off 2021 w/ https://t.co/jS8vCVRX5S GET P...	0	2021-02-22 14:05:17	0	2021-02-22

29334 rows × 6 columns

Here we can see that the time is changed into the date format.

Next, we use the Ethereum price data from Yahoo Finance. However, not all raw data features are required for this project. From this, just the closing and opening prices are extracted. This is required to establish the influence of tweets on the Ethereum market.

6.4.1 Tweets Cleaning:

The text data from tweets is challenging to decipher because these are not random news articles or statements. These are tweets from various users. We all know that tweets always contain hashtags, URLs, mentions, memorable characters, and punctuation. As we are using Python for this project, we have numerous predefined modules for locating and reducing noise. We removed the hashtags, URLs, mentions, special characters, and punctuations from the tweets. Several libraries are used to clean up the text using the code below.

```
#Extract date from the time column
eth_df['Date']= eth_df['time'].dt.date
#Cleaning the tweets
from nltk.tokenize import TweetTokenizer
from nltk.stem.wordnet import WordNetLemmatizer
lm = WordNetLemmatizer()

#Defining a funciton for cleaning the tweets.
def cleaning(tweets_data):
    #for removing the hashtags of the tweets
    tweets_without_hashtag = re.sub(r'#\w+', ' ', tweets_data)

    #remove the urls of the tweets
    tweet_without_url = re.sub(r'http\S+',' ', tweets_without_hashtag)

    #Remove mentions and characters that not in the English alphabets
    tweets_without_mentions = re.sub(r'@\w+',' ', tweets_without_hashtag)
    first_cleaned_tweets = re.sub('[^A-Za-z]+', ' ', tweets_without_mentions)


    #Tokenize the tweets
    tweet_tokens = TweetTokenizer().tokenize(first_cleaned_tweets)

    #Remove the Punctuations
    tokens_without_punctuations = [w for w in tweet_tokens if w.isalpha()]

    #Removing Stopwords
    tokens_without_sw = [t for t in tokens_without_punctuations if t not in stop_words]


    #lemmatize the tweets
    text_cleaned = [lm.lemmatize(t) for t in tokens_without_sw]

    #Joining the tweets after cleaning
    return " ".join(text_cleaned)

    return tweet
```

Multiple functions are created for each cleaning, allowing the task to be completed in a single step by calling the appropriate functions.

Regular expressions can be used to eliminate hashtags and URLs from tweets, as well as mentions and characters not found in the English alphabet. Importing re provides access to various regular expression keys. Here, the expression re.sub is performed, which is the more dependable and user-friendly Python library. By evaluating the data, we may employ the re.sub technique to replace some tweets with these noises. This is accomplished by establishing a space and replacing it with unwanted wording. Any irrelevant hashtags, URLs, punctuation, mentions, and emoticons are eliminated here. Even after removing all of the noise mentions, more strategies are required to apply the NLP algorithm to the tweets.

6.4.1.1 Removing the stopwords:

We are using the NLTK library to remove stopwords. Natural Language Toolkit (NLTK): A platform for developing Python programs that work on human language data. It comprises tokenization, parsing, classification, and stemming libraries. All English stopwords present in the NLTK library can be obtained. The stopwords provided by this library are listed below.

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

These are the available stopwords that are predefined. These may be referred to as the words that appear in every sentence. In this case, the reason for eliminating the stopword is that these words will not provide any necessary information for the project. Therefore, while applying NLP, we can eliminate these terms because they cannot be used to assess the sentiment of tweets.

6.4.1.2 Lemmatization:

Lemmatization is done to combine multiple words into one. By deleting the terms "merges," "merged," and "merging" and making them into the term "merge," each word is changed to its basic form. This library is imported from the nltk library using nltk.stem.wordnet import WordNetLemmatizer to lemmatize the words. Assigned lm as WordNetLemmatizer(); after that, we use this to lemmatize every word in the tweets.

Tokenisation is the process of dividing sentences into individual tokens so that, upon extraction, we can use these individual words to be selected for further analysis. First, the tokenization library is imported from the NLTK; specifically, import TweetTokenizer from nltk.tokenize. Then, the Tweets are tokenized using this TweetTokenizer.

6.4.1.3 Stemming

Here, the terms are reduced to their root forms. Consider, for instance, various terms similar to taste, such as tastier and tasty; the root of these words is taste. Therefore, we transform each tweet into the format described above. This procedure is known as stemming.

By performing the cleaning processes on the tweets, we can see that the table below shows that the tweets have been cleaned.

User	tweets	likes	time	retweets	cleaned_tweets
@VitalikButerin	@lexfridman 😜 \n\nhttps://t.co/JP1Ucky8h6	442	2022-08-23 20:58:21	21	JP Ucky h
@VitalikButerin	If I had to make a bear case for Anglo civiliz...	2378	2022-08-23 15:23:31	250	If I make bear case Anglo civilization would P...
@VitalikButerin	RT @VitalikButerin: @thefaketomato @palladiumm...	0	2022-08-23 15:18:40	50	RT X sound great mandatory X sound horrifying ...
@VitalikButerin	@thefaketomato @palladiummag "X sounds great, ...	367	2022-08-23 15:16:45	50	X sound great mandatory X sound horrifying man...
@VitalikButerin	@Max_Stirn @palladiummag Many kinds of technol...	15	2022-08-23 13:56:50	3	Many kind technology benefit rich people le po...
@VitalikButerin	@palladiummag In ultra-immersive fancy metaver...	6	2022-08-23 13:53:38	0	In ultra immersive fancy metaverse VR land guy...
@VitalikButerin	@palladiummag This is one of those ideas that ...	155	2022-08-23 13:51:22	7	This one idea sound appealing certain kind per...
@VitalikButerin	@ne0liberal Less a dunk and more a balanced re...	243	2022-08-22 20:20:19	11	Less dunk balanced review fascinating pointer ...
@VitalikButerin	@pourteaux Ok now I can't tell if you are maki...	124	2022-08-22 20:14:44	8	Ok I tell making fun genuinely supporting idea...

After using all the approaches for tweet cleaning, we are now able to show the outcomes. All tweets have been thoroughly cleaned by eliminating hyperlinks, lemmatizing them, and deleting stopwords. However, we are not yet satisfied with the outcome. As may be seen, the inputs contain considerable uncertainty. To determine whether cleanings are correctly performed or not. Before proceeding to the next preprocessing stage, more verifications must be performed.

Further Cleaning:

This dataset contains 29334 rows of data that may not be necessary for the project. The 29334 tweets represent all tweets posted by the selected influencers. These tweets may include general tweets, tweets

about politics, tweets about other cryptocurrencies, and other irrelevant financial tweets. However, only tweets linked to Ethereum are necessary for this project. Therefore, the Ethereum-related tweets are filtered using the following code.

```
eth_df2 = eth_df1[(eth_df1['tweets'].str.contains(r'ETH(?!$)')) | (eth_df1['tweets'].str.contains(r'eth(?!$)')) | (eth_df1['tweets'].str.contains(r'Eth(?!$)'))]
```

The phrases associated with Ethereum are selected, and all other tweets are eliminated. After applying the approach, all tweets are filtered to include the phrases Ethereum, Eth, ETH, and other influencer-related terms. Following the filtration procedure, the output data frame is displayed below.

	User	tweets	likes	time	retweets	Date
5	@VitalikButerin	In ultra immersive fancy metaverse VR land guy...	6	2022-08-23 13:53:38	0	2022-08-23
18	@VitalikButerin	Actually phrase anti civilization moral critic...	4	2022-08-20 21:44:45	1	2022-08-20
30	@VitalikButerin	None principle literally infinite weight alway...	33	2022-08-20 18:35:58	0	2022-08-20
47	@VitalikButerin	I argue moral perspective N py agent implement...	18	2022-08-19 14:35:44	0	2022-08-19
63	@VitalikButerin	Glad see Ethereum people pushing regulation pr...	3530	2022-08-17 23:43:17	533	2022-08-17
...	...	...	...	...	...	...
29009	@Cyber_FM	Welming jS vCVRX S GET PAID listening The Tu...	0	2021-02-24 09:15:02	0	2021-02-24
29010	@Cyber_FM	Positivity jS vCVRX S GET PAID listening Tog...	0	2021-02-24 08:47:03	0	2021-02-24
29132	@Cyber_FM	Welming jS vCVRX S GET PAID listening TOGETH...	0	2021-02-23 16:50:41	0	2021-02-23
29214	@Cyber_FM	Positivity jS vCVRX S GET PAID listening Tog...	1	2021-02-23 05:19:25	0	2021-02-23
29252	@Cyber_FM	Welming jS vCVRX S GET PAID listening The Tu...	0	2021-02-23 00:22:44	0	2021-02-23

6567 rows × 6 columns

Here there is a sudden decrease in the number of rows; as we can see, after the filtration process, there are only 6,567 rows remaining, following which we proceed with the exploratory analysis.

In order to determine the quality of the dataset, no repeated words are considered. This yields an ideal result for the dataframe, from which we must delete any unnecessary terms. It is necessary to count the number of words. To do this, we apply "from collections import Counter."

```
eth_df_wrd['temp_list'] = eth_df_wrd['tweets'].apply(lambda x:str(x).split())
top = Counter([item for sublist in eth_df_wrd['temp_list'] for item in sublist])
temp = pd.DataFrame(top.most_common(20))
temp.columns = ['Common_words','count']
temp.style.background_gradient(cmap='Blues')
```

We must apply the split function to extract each tweet's words into strings. These strings are stored in a temporary list, and counting is then performed on this column without affecting the original tweets. Here,

the Counter is used to tally the frequency of the words. Below is the output after counting the common words.

	Common_words	count
0	fee	11209
1	next	6583
2	block	6568
3	Ethereum	4967
4	ETH	2792
5	Bitin	2687
6	BTC	2486
7	hour	2260
8	past	2245
9	Nano	2238
10	NANO	2237
11	LTC	2230
12	Litein	2230
13	nfirmed	2187
14	fully	2184
15	Coin	2182
16	send	2051
17	It	1864
18	accumulate	1355
19	RT	1239

Table 2 Common words count

Here are the 20 most common words. The data indicates that there are no stopwords in this list, indicating that the data cleansing has been performed flawlessly. Moreover, the phrases "ETH" and "Ethereum" can be seen at the top. If we discover any irrelevant words that are not relevant to the Ethereum market, we can remove them here. Also we need to clean the tweets rows whichever tweets didn’t get any like or retweets. This is done by searching for the 0 likes and deleting the rows whichever the value of the likes and retweets becomes 0.

6.5 EXPLORATORY DATA ANALYSIS (EDA):

Exploratory data analysis was performed to provide a generic data visualisation and data-based insights. Here, for EDA, analysis is conducted using the available data. We cannot apply the numerical EDA approach to textual data. Here, the manner of approach is distinctive. First, we can conduct some generic analysis, such as identifying the most frequent words, and then we will design more precise options to make EDA more efficient. (Ahmad, 2019,)

6.5.1 The total number of tweets per influencers:

First, the number of tweets per influencer is analysed. The graph below displays the total number of tweets posted by each influencer.

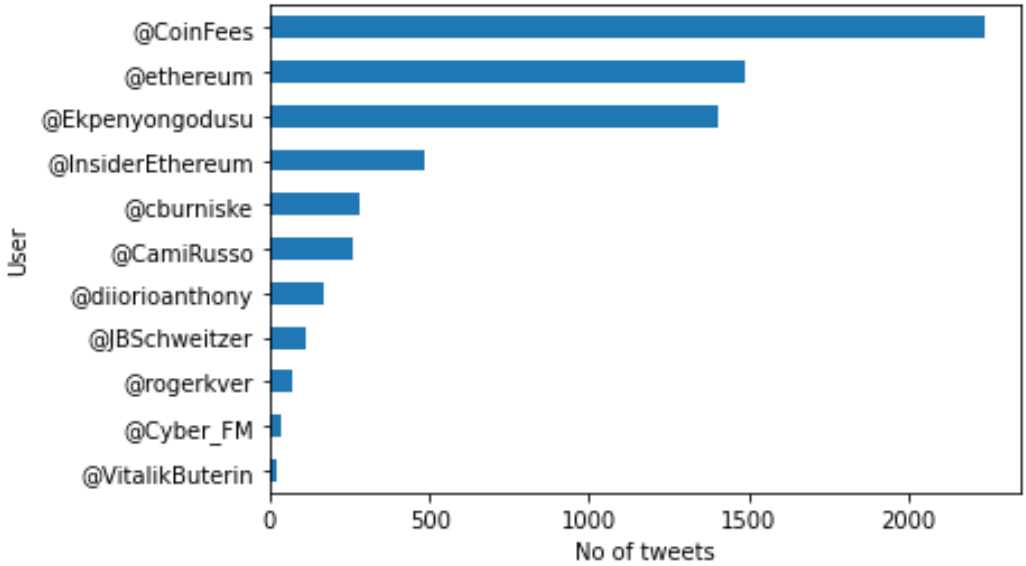


Figure 7: Total no of tweets per user

Most tweets about Ethereum are posted by @Coinfees, a fee estimator for all cryptocurrencies. Therefore, they are always on Twitter for daily tweets. The following account is @ethereum, which is the official account of Ethereum and likely tweets constantly about Ethereum. By evaluating the number of tweets, we can conclude that the data is more relevant and efficient. Vitalik Buterin, Cyber FM, also makes the fewest tweets about Ethereum. Most tweets about Ethereum are posted by @Coinfees, a fee estimator for all cryptocurrencies. Therefore, they are always on Twitter for daily tweets. The following account is @ethereum, which is the official account of Ethereum and likely tweets constantly about Ethereum. By evaluating the number of tweets, we can conclude that the data is more relevant and efficient. Vitalik Buterin, Cyber FM, also makes the fewest tweets about Ethereum.

6.5.2 Analysing top twenty common words:

The next part of the EDA is to create a visualization of the most common words that are present in the tweets. The process is the same as explained before, using the counter library from the collection in order to get the number of words. Here is the output from the most common words is given below.

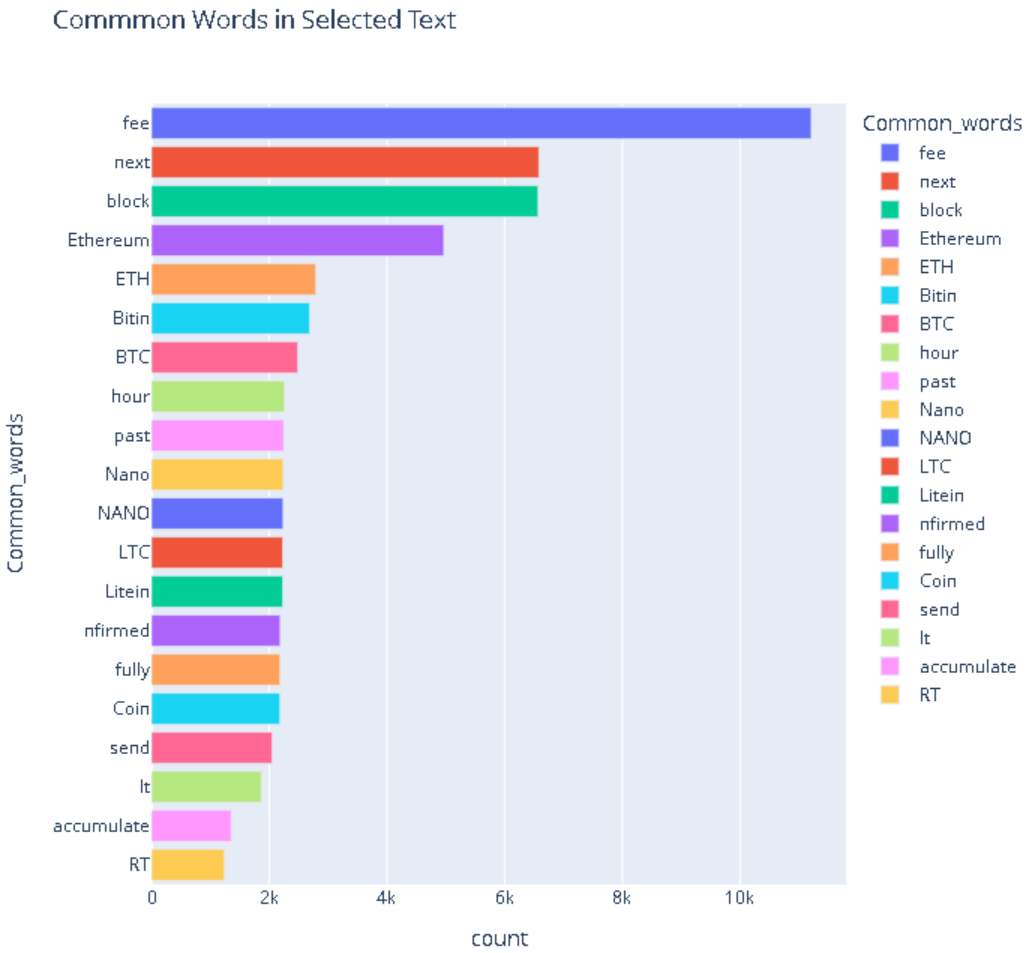


Figure 8 Common words in tweets

Here are the most common words present in the tweets. Here the ethereum also comes on the top which gives some head to the data's quality. The word fee comes above 10K and the words like Ethereum which comes around 4967 and ETH comes next with count of 2792 and Bitin with 2687 count and BTC with 2486 count. Here the BTC is a general term used for bitcoin so we can conclude that even if the influencers are based on Ethereum, there is also a change that they all plays a relevant role in the making of another crypto currency market. That is there comes the BTC as a common words in here.

6.5.3 WordCloud of the tweets

WordCloud is also created by analyzing the common words. Create the WordCloud based on the tweets using WordCloud library. It is technique to view the common words as a cluster on words. The most common words can be seen more enlarges and the least common words are seen as smaller.

The code below is used to get the word cloud

```
import wordcloud
common_words=''
for i in eth_df7.tweets:
    i = str(i)
    tokens = i.split()
    common_words += " ".join(tokens)+" "
wordcloud = wordcloud.WordCloud().generate(common_words)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

WordCloud is a library were we can print the common words as an image format. Here we use imshow plotter to plot the image.

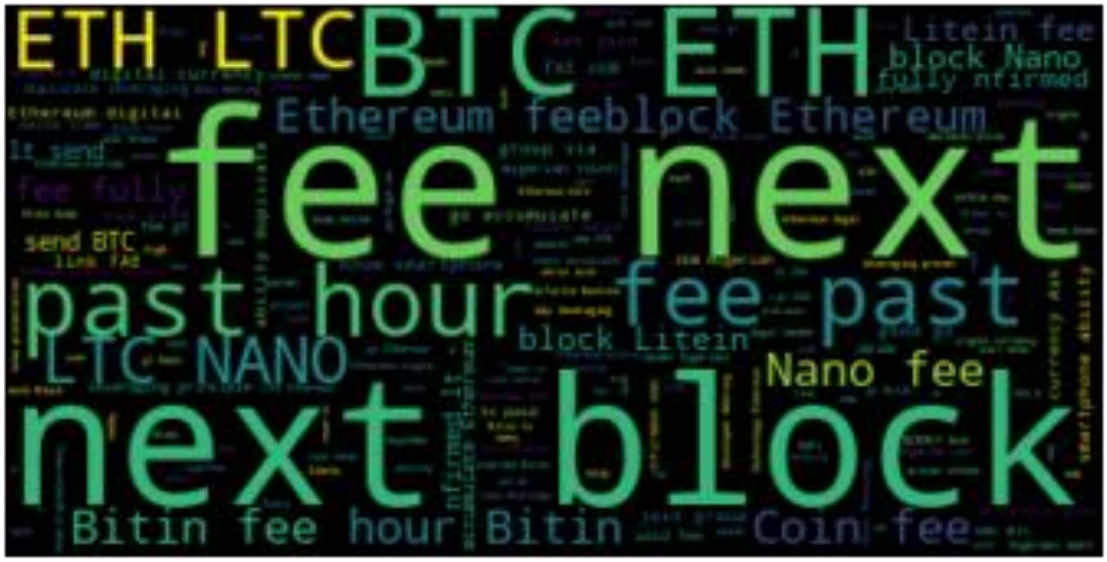


Figure 9: WordCloud of tweets

Words such as BTC, ETH, Ethereum, block, Coin, etc., can be seen clearly in the WordCloud, which is an excellent indicator of data visualisation and the correctness of the data preprocessing stage, which was performed precisely.

6.5.4 Exploratory analysis on ethereum price data:

Several numerical datasets provide the Ethereum market price. Consequently, EDA must also be performed on that dataset. When cleansing the data, only the opening and closing prices for Ethereum are considered. This data frame is later used for additional data visualisation.

	Date	Open	High	Low	Close	Adj Close	Volume
1149	2021-01-01	737.708374	749.201843	719.792236	730.367554	730.367554	13652004358
1150	2021-01-02	730.402649	786.798462	718.109497	774.534973	774.534973	19740771179
1151	2021-01-03	774.511841	1006.565002	771.561646	975.507690	975.507690	45200463368
1152	2021-01-04	977.058838	1153.189209	912.305359	1040.233032	1040.233032	56945985763
1153	2021-01-05	1041.498779	1129.371460	986.811279	1100.006104	1100.006104	41535932781
...	...	...	...	...	...	...	...
1509	2021-12-27	4064.746338	4126.001465	4033.492432	4037.547607	4037.547607	11424360002
1510	2021-12-28	4037.538086	4037.538086	3769.280029	3800.893066	3800.893066	17299472803
1511	2021-12-29	3797.436279	3827.981934	3612.795898	3628.531738	3628.531738	15722555672
1512	2021-12-30	3632.219727	3767.559814	3595.204834	3713.852051	3713.852051	12925377999
1513	2021-12-31	3713.430176	3807.288818	3636.869873	3682.632813	3682.632813	14157285268

365 rows × 7 columns

Table 3 Raw data of Ethereum price from Yahoofinance.com

6.5.5 Ethereum close price over time:

The figure is made using the plotly library, which produces more current and visually appealing plots than the matplotlib libraries. The graph below illustrates the distribution of Ethereum's closing price over time.

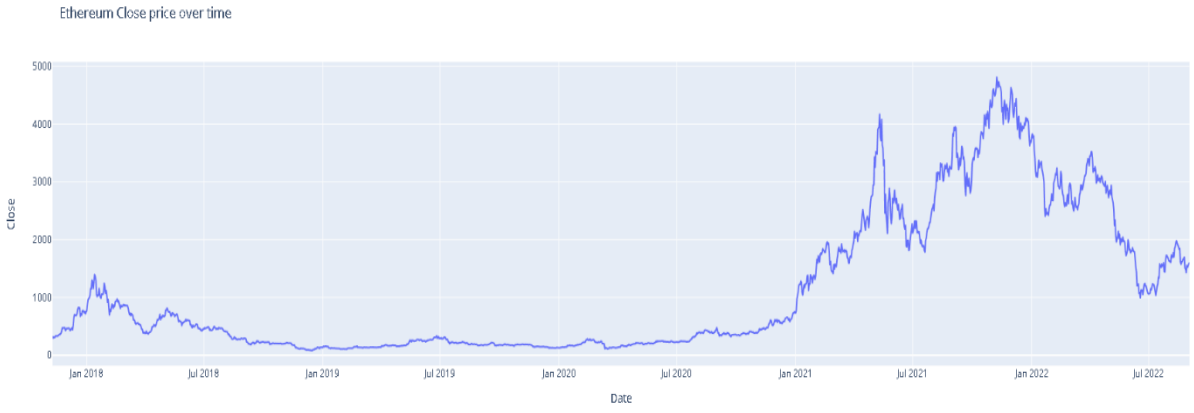


Figure 10: Ethereum close price over time

Here, the graph displays values from November 2017 to the present. Between 2018 and 2021, Ethereum's market price trend demonstrates moderate fluctuations. This may be attributed to the Covid increase in the world economy. At the time, this affected the entire market. After 2021, however, we can anticipate a little increase in the market price. Between 2021 and 2022, the market reached its peak. However, this is not a stable market; it is a dynamic market, and we cannot anticipate it now. The market is now experiencing fluctuation, but the value of Ethereum on the crypto market represents the largest economy.

6.5.6 Monthly average opening and close price

As per above analysis there is a more fluctuating market for the 2021 year so for the next analysis I am taking the period to find an excellent graph. We can see the monthly average price of the ethereum opening and closing price throughout the year 2021. The opening and closing values of each month are analysed and plotted as given below. Here blue plot represent average open price and red represent average closing price of the moth. We can see that there is more closing and opening price values come from the month November.

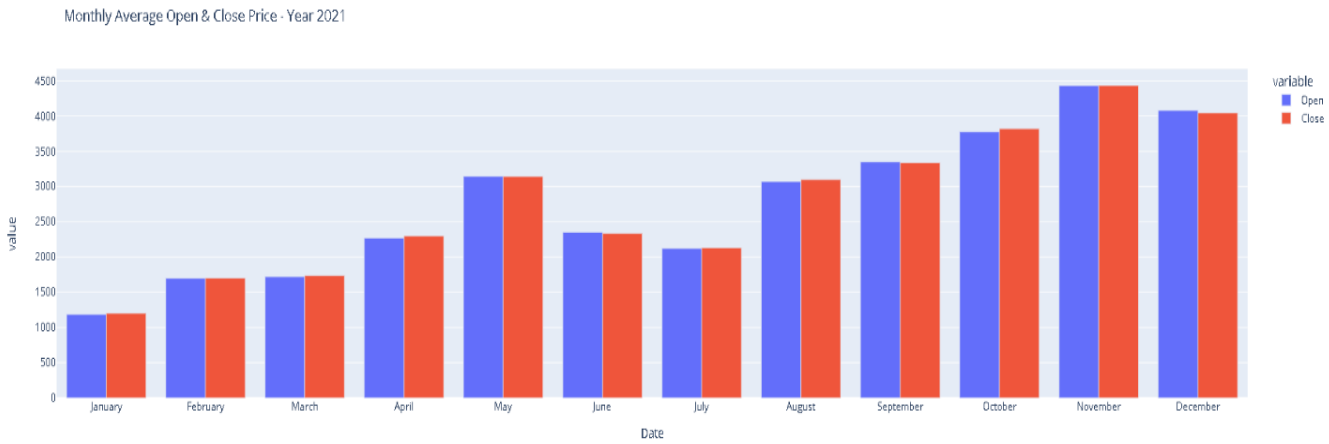


Figure 11: Monthly average Open and close price - year 2021

The EDA of both Twitter and Yahoo data is complete. These specific data must be merged in order to create an analysis and model for further analysis. The cleaned and analysed data must be contained in a single dataframe. As a result of merging the data with the regular column, we can say that the date is the characteristic shared by both data sets. However, here the format is distinct, and we must convert it to one of the standard formats.

6.5.7 Fitting of close price data using LSTM

Close price of ethereum is selected and fitted to the original price as a predictive model. Here we take (1231, 1) as training as (528, 1) as testing. Based on this x_train, y_train, x_test, y_test are calculated.

```
x_train = []
y_train = []

for i in range(100, data_training_array.shape[0]):
  x_train.append(data_training_array[i-100: i])
  y_train.append(data_training_array[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
```

```
model = Sequential()
model.add(LSTM(units=50, activation = 'relu', return_sequences = True,
               input_shape = (x_train.shape[1],1)))
model.add(Dropout(0.2))


model.add(LSTM(units=60, activation = 'relu', return_sequences = True))
model.add(Dropout(0.3))


model.add(LSTM(units=80, activation = 'relu', return_sequences = True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation = 'relu'))
model.add(Dropout(0.5))

#model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.add(Dense(units = 1))

model.summary()
```

Figure 12 LSTM Close price of Ethereum

Model is compiled by selecting optimizer as adam and loss as mean squared error.

Here 100^{th}day's prediction is found and plotted a graph regarding that is given below. Input to the output activation is kept as relu. The units selected from 50 to 120, this is a dimensionality of the output space.

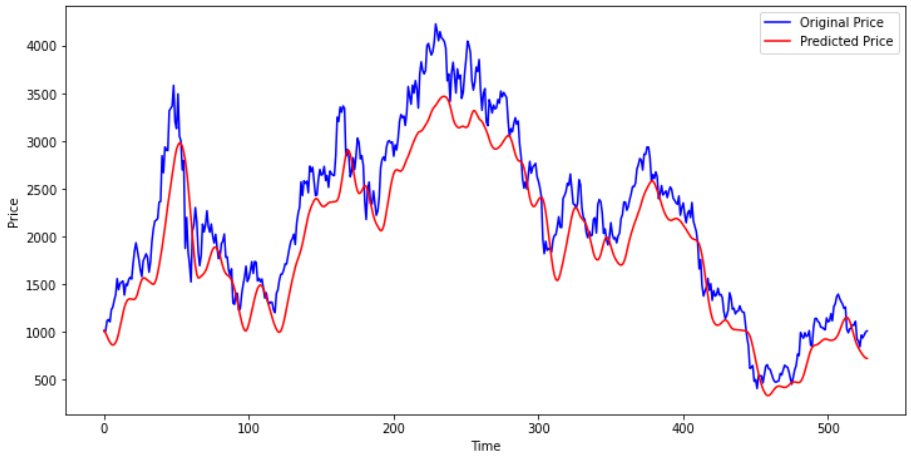


Table 4 fitting ethereum close price with original price

6.5.8 Merging both the twitter data and the ethereum price data frames.

The Ethereum price data merged with the Twitter data. Here, the data is more relevant for further analysis. Here, all the tweet data is merged in line with the Ethereum price data we get from Yahoo Finance. We converted the date format into the same as Twitter, and we merged the two dataframes by taking the date as the common factor for merging. We can see the converted date format here.

Raw data:

	Date	Open	Close	Price_diff
0	09-11-2017	308.644989	320.884003	-12.239014
1	10-11-2017	320.670990	299.252991	21.417999
2	11-11-2017	298.585999	314.681000	-16.095001
3	12-11-2017	314.690002	307.907990	6.782012
4	13-11-2017	307.024994	316.716003	-9.691009
...	...	...	...	...
1750	25-08-2022	1657.336548	1696.457031	-39.120483
1751	26-08-2022	1696.324585	1507.782837	188.541748
1752	27-08-2022	1508.156982	1491.395020	16.761962
1753	28-08-2022	1491.206787	1430.547363	60.659424
1754	29-08-2022	1432.359131	1536.331665	-103.972534

1755 rows × 4 columns

Table 5 Ethereum price data with difference

Here the date format is converted it into same format as in the tweet data. The total amount of data we receive after merging is 1755, which is increased to 3058 due to the daily influx of tweets. There could be more tweets daily, so we can obtain more tweet data after merging it with the price data. The open, close, and price difference is also displayed along with the tweets. We must search for null values and subsequently delete them.

Date	User	tweets	likes	time	retweets	Open	Close	Price_diff
2013-06-11	@rogerkver	If opposed government inflating money pay war ...	3	2013-06-11 16:04:06	11	NaN	NaN	NaN
2013-07-04	@rogerkver	http co kKww NTlEj new user day k user altoget...	1	2013-07-04 12:58:36	4	NaN	NaN	NaN
2013-07-31	@rogerkver	Bitcoin going happen whether regulator want ha...	8	2013-07-31 01:34:03	10	NaN	NaN	NaN
2013-08-14	@rogerkver	I paid bank USD conversion fee wasted hour ban...	3	2013-08-14 04:39:43	14	NaN	NaN	NaN
2013-08-18	@rogerkver	I would rather sit tub full pig feces smoke me...	3	2013-08-18 08:00:06	4	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
2022-08-23	@VitalikButerin	In ultra immersive fancy metaverse VR land guy...	6	2022-08-23 13:53:38	0	1622.939331	1662.769897	-39.830566
2022-08-23	@CamiRusso	DeFi liquidation volume v ETH price time gt ha...	61	2022-08-23 13:36:44	12	1622.939331	1662.769897	-39.830566
2022-08-23	@InsiderEthereum	Most PoW miner intend mine Ergo Ravencoin Ethe...	6	2022-08-23 11:18:20	2	1622.939331	1662.769897	-39.830566
2022-08-23	@InsiderEthereum	Bitcoin Ethereum future exerting dominance spo...	2	2022-08-23 11:18:20	0	1622.939331	1662.769897	-39.830566
2022-08-23	@InsiderEthereum	Bitfinex offer new chain split token ahead Eth...	2	2022-08-23 15:18:19	0	1622.939331	1662.769897	-39.830566

3058 rows × 8 columns

Table 6 Merged Twitter and Ethereum price data

Here, we remove the nan values from the data frame and convert all the desired items. The final dataframe that will be analysed is presented below. There are 2239 rows of data and seven features displayed here. All data must be appropriately trained in the future.

	User	Date	tweets	likes	retweets	Close	Price_diff
0	@ethereum	2017-04-12	Developer Update Ethereum Core Developers Meet...	89	35	470.204010	-4.150024
1	@rogerkver	2017-11-12	Vitalik genius class act That I sold portion B...	1643	464	515.135986	-74.777984
2	@ethereum	2017-11-17	LIVE Ethereum Core Devs Meeting http co OK bka...	126	68	332.394012	-2.227020
3	@ethereum	2017-11-17	Notes Ethereum Core Devs Meeting http co qgpgw...	198	96	332.394012	-2.227020
4	@ethereum	2017-12-15	LIVE Ethereum Core Devs Meeting http co sKGBGK...	239	135	684.447998	11.927979
...	...	...	...	...	...	...	...
2234	@VitalikButerin	2022-08-23	In ultra immersive fancy metaverse VR land guy...	6	0	1662.769897	-39.830566
2235	@CamiRusso	2022-08-23	DeFi liquidation volume v ETH price time gt ha...	61	12	1662.769897	-39.830566
2236	@InsiderEthereum	2022-08-23	Most PoW miner intend mine Ergo Ravencoin Ethe...	6	2	1662.769897	-39.830566
2237	@InsiderEthereum	2022-08-23	Bitcoin Ethereum future exerting dominance spo...	2	0	1662.769897	-39.830566
2238	@InsiderEthereum	2022-08-23	Bitfinex offer new chain split token ahead Eth...	2	0	1662.769897	-39.830566

2239 rows × 7 columns

Table 7: Final dataframe after merging

6.5.9 Sentiment calculation using polarity and Subjectivity:

After the cleaning of the tweets we need to find the polarity and subjectivity of each tweets. Polarity values are calculated using Textblob library is used to calculate this values. The calculated polarity and subjectivities are divided into categories as impact which gives the sentiment as negative, neutral and

positive sentiments. here “from textblob import TextBlob” is to access the textblob library. The sentiment is calculated as the values of polarity are taken into consideration.

If the polarity values are less than 0 then the sentiment will be negative. If polarity = 0 then sentiment will be neutral and if polarity is > 0 then sentiment will be positive. All the sentiments are generated and saved as a separate column for further analysis. The outcomes from this can be say that maybe the ethereum price not always correlated with the market Trent. The sentiment always equals to the impact which I calculated earlier.

We can see the values of polarity value and the calculated sentiment along with the subjectivity.

Subjectivity	Polarity	Sentiment
0.000000	0.000000	neutral
0.333333	0.250000	positive
0.616667	0.216667	positive
0.000000	0.000000	neutral
0.500000	0.136364	positive
0.500000	0.136364	positive
0.500000	0.318182	positive
0.000000	0.000000	neutral
0.000000	0.000000	neutral

Table 8 Sentiment based on the polarity and subjectivity

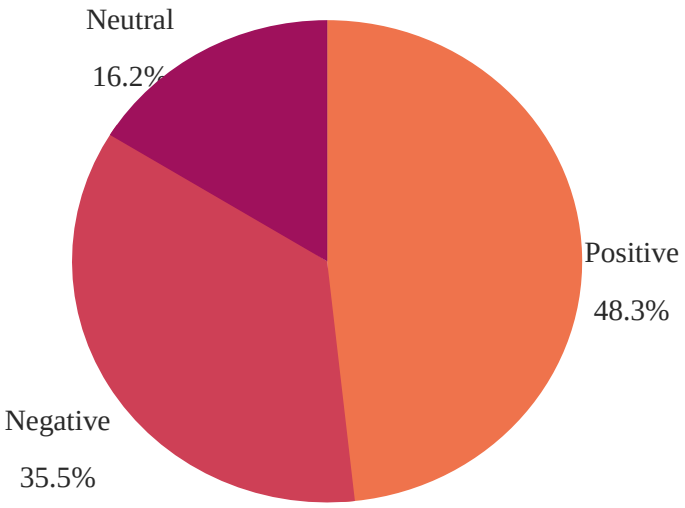


Figure 13pi-chart of Sentiments

The above chart gives the value of sentiments of positive, negative and neutral. Here about 48% of the tweets are positive and other 35% are negative and only 16 % is neutral sentiment.

6.5.10 Creating the tweet trends

The subsequent, most essential task is constructing the model according to the model's preferences. We must first employ any available method to determine a correlation between tweets and price. We can take the close price directly, but I prefer to use the price difference, representing the daily price change between the open and closed prices. The price difference may be negative or positive; a negative price difference indicates that today's closing price is lower than yesterday's.

$$Price\ difference\ =\ closing\ price\ (today)\ -\ opening\ price(today)$$

We can also take the equation as

$$price\ difference\ =\ Yesterday's\ closing\ price\ -\ todays\ closing\ price.$$

Each variation is possible, and the price difference can be viewed in the above-described dataset. This must be converted into a percentage before determining how the tweets will affect the market. Here, all price differences are converted to percentages.

$$Percentage\ daily\ change\ =\ \frac{closing\ price\ (today)-\ opening\ price(today)}{opening\ price\ (today)}$$

This equation is applied to all the price difference data then we get a percentage change of every values per day. This is given on the figure below

	User	Date	tweets	likes	retweets	Close	Price_diff	Percent
0	@ethereum	2017-04-12	Developer Update Ethereum Core Developers Meet...	89	35	470.204010	-4.150024	-0.882601
1	@rogerkver	2017-11-12	Vitalik genius class act That I sold portion B...	1643	464	515.135986	-74.777984	-14.516164
2	@ethereum	2017-11-17	LIVE Ethereum Core Devs Meeting http co OK bka...	126	68	332.394012	-2.227020	-0.669994
3	@ethereum	2017-11-17	Notes Ethereum Core Devs Meeting http co qgpgw...	198	96	332.394012	-2.227020	-0.669994
4	@ethereum	2017-12-15	LIVE Ethereum Core Devs Meeting http co sKGBGK...	239	135	684.447998	11.927979	1.742715

Table 9: Daily percentage change for Ethereum price

All values are converted to the change in percentage. This approach to model creation allows me to take the percentage or price difference directly, but the project strategy is to convert it into a market impact sentiment. The impact is categorised as negative, neutral, or positive based on the percentage value. Here we can see from the table below that

```
for index, row in eth_df7.iterrows():
    if eth_df7.loc[index, 'Percent'] < 0:
        eth_df7.loc[index,'Impact'] = 'Negative Impact'
    elif eth_df7.loc[index, 'Percent'] >= 0 and eth_df7.loc[index, 'Percent'] <= 2 :
        eth_df7.loc[index,'Impact'] = 'Neutral Impact'
    else:
      eth_df7.loc[index, 'Impact'] = 'Positive Impact'
eth_df7.head()
```

	User	Date	tweets	likes	retweets	Close	Price_diff	Percent	Impact
2238	@InsiderEthereum	2022-08-23	Bitfinex offer new chain split token ahead Eth...	2	0	1662.769897	-39.830566	-2.395435	Negative Impact
2235	@CamiRusso	2022-08-23	DeFi liquidation volume v ETH price time gt ha...	61	12	1662.769897	-39.830566	-2.395435	Negative Impact
2234	@VitalikButerin	2022-08-23	In ultra immersive fancy metaverse VR land guy...	6	0	1662.769897	-39.830566	-2.395435	Negative Impact
2237	@InsiderEthereum	2022-08-23	Bitcoin Ethereum future exerting dominance spo...	2	0	1662.769897	-39.830566	-2.395435	Negative Impact
2236	@InsiderEthereum	2022-08-23	Most PoW miner intend mine Ergo Ravencoin Ethe...	6	2	1662.769897	-39.830566	-2.395435	Negative Impact

Table 10 Impact calculated from percentage value of ethereum daily price change

Different types of loops are used to create functions that categorise percentage values as positive, neutral, or negative. Tweets are classified as negative when the percentage value is less than zero. If the percentage is greater than 0, the impact is positive. If the percentage is equal to zero, then there is no Impact or neutral impact. The actual difficulty lies in the neutral impact because we cannot see some counts and analyse the neutral value. The neutral value zero infrequently occurs, a significant systemic unusual case. Inconsistency and distraction in the model creation are eliminated by taking the neutral values ranging from -2 to 0 and categorized as neutral impact. Thus, the neutral values can be increased.

The sentiment impact based on the percentage approach of the ethereum price market is given below.

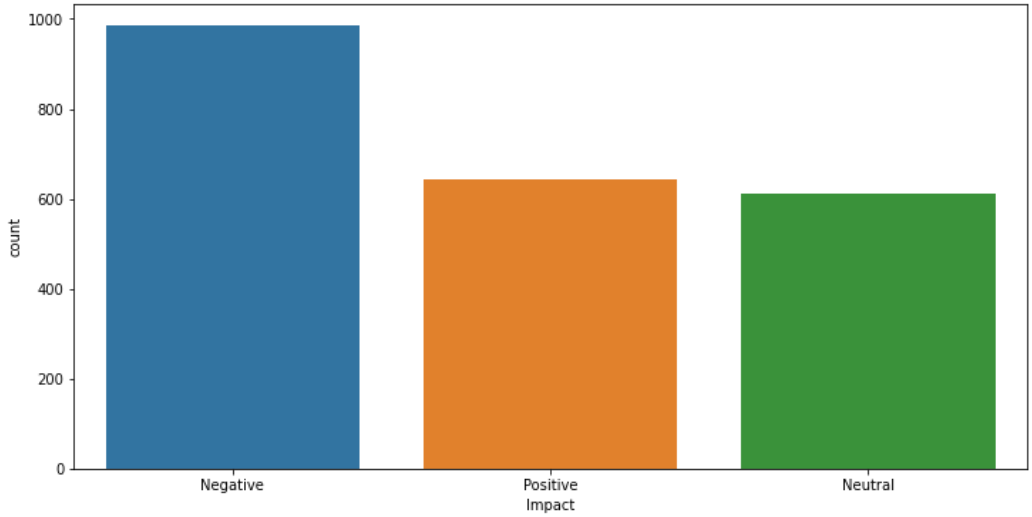


Figure 14 Sentiment Impact based on the Ethereum market price

About 1000 values has a percentage of price impact as negative. And about 700 values are impacted as positive and 600 as neutral impact.

6.6 MODEL BUILDING:

Before building the model, the data's are separated into X as input and y as output. The dataset is partitioned, with tweets serving as the input and Impact as the output. Finally, we conduct training on tweets to develop a model that generates an output based on tweets relating to the Impact.

6.6.1 Train test split:

We are splitting both the X and y according to our project objectives. Here the X_train, X_test, y_train, y_test are splitted and calculated. We take 80 percentage of data as training and 20 percentage as the test data.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1791,) (448,) (1791, 3) (448, 3)
```

There are 1791 x_train and 448 x_test data sets. Therefore, the output y_train has 17791 rows of data, and the output y_test has 448 rows of data. Because of the three impact features, the y_data is converted into three different number categories.

We can also use the sentiment as the output variable y, which is unnecessary because it provides no information about the Ethereum price. They have no relationship with one another. That is why we use Impact as the output variable.

Running the model

Here, we run 50 epochs, which produces some results but has the potential to become overfitting. The keras early stopping method can be used to avoid model overfitting.

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

earlyStop=EarlyStopping(monitor="val_loss",verbose=2,mode='min',patience=3)
history1=model.fit(X_train,y_train,epochs=10,batch_size=10,validation_data=(X_test,y_test) ,verbose=2,callbacks=[earlyStop])
```

The Keras library is used to implement the early stopping. It will prevent the model from overfitting by not running all of the epochs. This will result in the best possible output. The val loss will also be lower if early stopping is used instead of general epochs.

6.6.2 LSTM model building;

To train a neural network from a large corpus, we need a sequential method that specifies the order in which the data is obtained for training. The model is built with the TensorFlow library. Every word must be converted to a specific sequence to produce an effectual output. The following steps in the LSTM process are used to create an LSTM model: Dense, Embedding, Conv1D, MaxPooling1D, and LSTM. The code below demonstrates how the LSTM model was created.

```
np.random.seed(seed)
K.clear_session()
model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X_train.shape[1]))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

Figure 15: Code for LSTM model

Different types of methods are used in here. Different types of models are added to the sequential order in order to get an optimum output.

The output of LSTM model is given below.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 30, 100)           2000000

 conv1d (Conv1D)             (None, 30, 32)            9632

 max_pooling1d (MaxPooling1D  (None, 15, 32)           0
 )

 conv1d_1 (Conv1D)           (None, 15, 32)            3104

 max_pooling1d_1 (MaxPooling  (None, 7, 32)            0
 1D)

 lstm (LSTM)                 (None, 100)               53200

 dense (Dense)               (None, 3)                 303

=================================================================
Total params: 2,066,239
Trainable params: 2,066,239
Non-trainable params: 0
```

Figure 16: LSTM model output

- Embedding converts the positive integers into fixed size vector form.
- Cov1d: filter gives the dimensionality of output. Kernel size gives the length of 1d convolution window. Padding gives the padding to either directions.
- Max Pooling is done to get the padding from the output shape. Here we need to specify the pool size.

The accuracy value and all the parameters value we get from the final summary table of LSTM model which is given below.

Accuracy:	80.0%			
	precision	recall	f1-score	support
0	0.00	0.00	0.00	16
1	0.40	0.45	0.43	22
2	0.87	0.96	0.91	127
accuracy			0.80	165
macro avg	0.42	0.47	0.45	165
weighted avg	0.72	0.80	0.76	165

Table 11: Summary table for LSTM.

The model accuracy and loss plot is also given below.

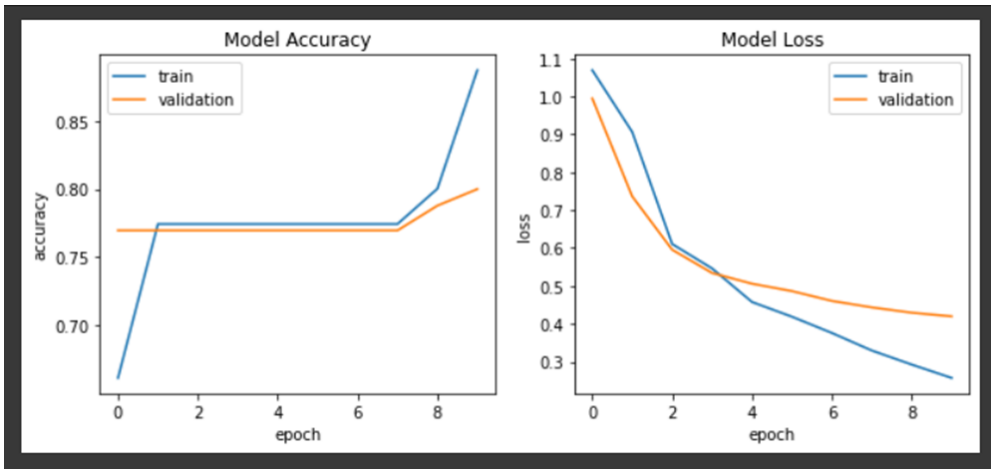


Figure 17: Model accuracy and loss plot

The models gives an accuracy of 80% which makes this model good for implementation. But inorder to find the best model we need to do some more model implementation. Here we use bidirection LSTM and the random forest regression according to my findings.

6.6.3 Bidirectional LSTM:

By using the bidirectional LSTM method we will get an accuracy more than the LSTM method. Bidirectional LSTM gives slightly upper hand in the accuracy of LSTM model. One dimensional LSTM is also good accuracy but the bidirectional LSTM gives more accuracy than the other models. In every case the whole x variables and y variables are the same.

Accuracy:	84.1%			
	precision	recall	f1-score	support
0	1.00	0.31	0.48	16
1	1.00	0.29	0.44	21
2	0.83	1.00	0.91	127
accuracy			0.84	164
macro avg	0.94	0.53	0.61	164
weighted avg	0.87	0.84	0.81	164

Table 12: summary table for Bidirectional LSTM

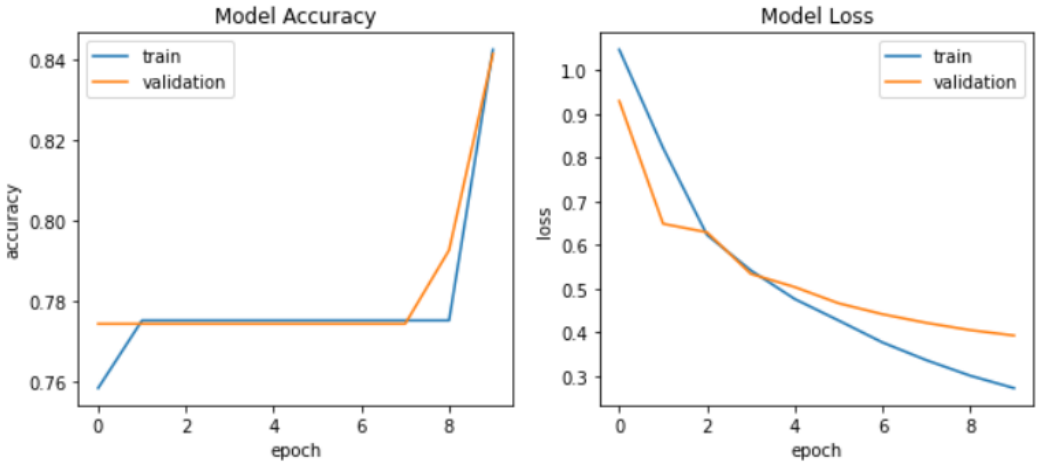


Table 13: model accuracy and loss plot of BiLSTM

Compared to the LSTM model, the BiLSTM model produces more accurate results, giving the analysis an edge over the LSTM. The accuracy is enhanced to 84.1% in this case, which is an excellent motivation to develop such a model.

Advantage of bidirectional LSTM over LSTM

This design style offers several benefits for solving real-world issues, particularly in NLP. The primary reason is that each input sequence component contains information from both the past and the present. Combining LSTM layers from both directions enables BiLSTM to provide a more relevant result.

6.6.4 Random Forest Regressor model:

To find and optimum calculation of both the model we need to find out which model is more efficient than the other two model. I have done some analysis on the topic and selected the random forest

regression model for further model building. Here it is a tree based model which can be created using the library.

Random Forest is a well-known Supervised Machine Learning Algorithm for Classification and Regression. First, individual decision trees are generated for each sample from n random records. Then, for each decision tree, the output will be generated. We are using sklean's default parameters. We trained the model using RandomForestClassifier().

Random forest classifier is build using the code below.

```
from sklearn.ensemble import RandomForestClassifier


rfc=RandomForestClassifier(n_estimators= 100, random_state= seed)
rfc.fit(X_train, y_train)
predictions = rfc.predict(X_test)
```

	precision	recall	f1-score	support
0	0.88	0.28	0.42	25
1	0.67	0.61	0.64	49
2	0.94	0.94	0.94	374
micro avg	0.91	0.87	0.89	448
macro avg	0.83	0.61	0.67	448
weighted avg	0.91	0.87	0.88	448
samples avg	0.87	0.87	0.87	448

Accuracy : 86.83 %

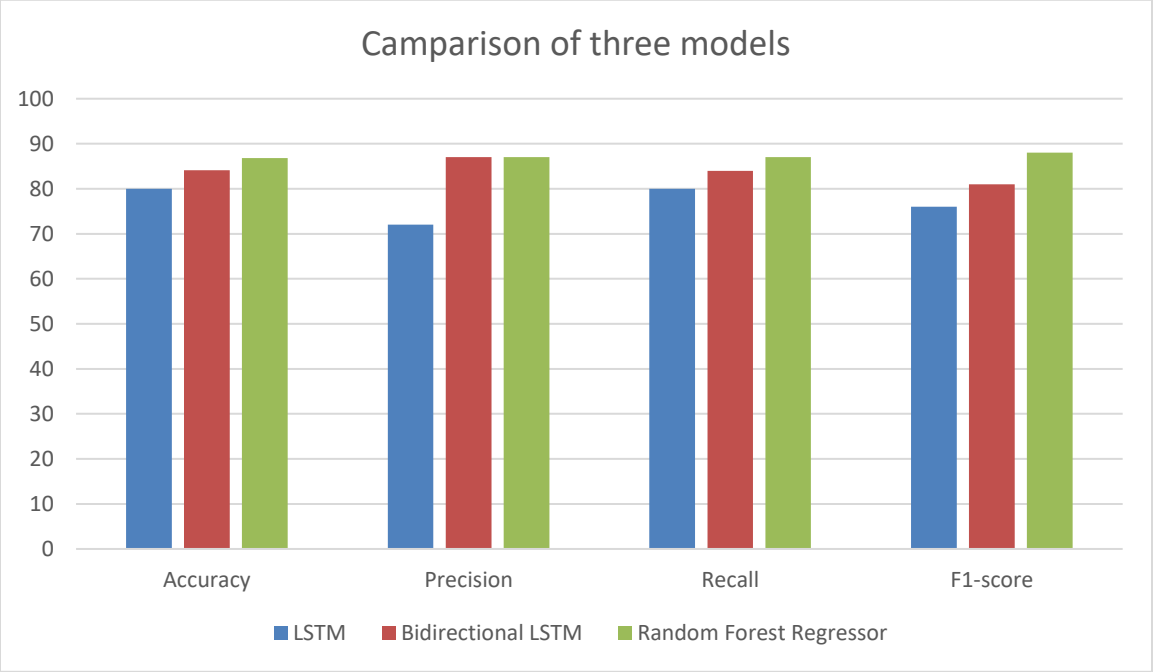
Table 14: Summary table of Random forest regressor

Here the random forest regressor gives more accuracy than other models with an accuracy of about 86.3%. Next step is to compare all the models with the effective way and make a final decision of which model is the best to implement for the future work. The random forest model gives more reliable and effective output compare to the other models. The model building works best on the categorise data compared to other data's.

6.7 COMPARISON OF THREE MODELS.

Parameters	LSTM	Bidirectional LSTM	Random Forest Regressor
Accuracy	80	84.1	86.83
Precision	72	87	87
Recall	80	84	87
F1-score	76	81	88

Table 15 Comparison of different model approaches.



Three alternative model parameters are examined, and it is discovered that the Random forest regressor surpasses all other models in terms of accuracy, precision, recall and F1-score. All of the parameter values are listed above. It is obvious that this model has an accuracy of about 86%.

6.8 DASHBOARD:

Created the dashboard using the streamlit library from python. Streamlit is a Python-based open-source app framework. It enables us to develop web applications for data science and machine learning quickly. It works with important Python libraries, including scikit-learn, Matplotlib, Keras, NumPy, pandas, and others. We use the visual studio code for the dashboard creation

```python
from sklearn.preprocessing import MinMaxScaler
from turtle import st
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pandas_datareader as data
from keras.models import load_model
import streamlit as st

# Describe
close_eth = pd.read_csv("ETH-USD.csv")
st.title('Ethereum Price Prediction')
user_input = st.text_input('Enter a user', 'vitalik')
close_eth.reset_index()
st.subheader('Ethereum')
st.write(close_eth.describe())

st.subheader('Closing Price vs Time chart')
fig = plt.figure(figsize=(12, 6))
plt.plot(close_eth.Close)
st.pyplot(fig)

st.subheader('Closing Price vs Time chart with 100MA')
ma100 = close_eth.Close.rolling(100).mean
fig = plt.figure(figsize=(12, 6))
plt.plot(ma100)
plt.plot(close_eth.Close)
st.pyplot(fig)

st.subheader('Closing Price vs Time chart with 100MA and 200MA')
ma100 = close_eth.Close.rolling(100).mean
ma200 = close_eth.Close.rolling(200).mean
fig = plt.figure(figsize=(12, 6))
```

Figure 18: code example for Dashboard

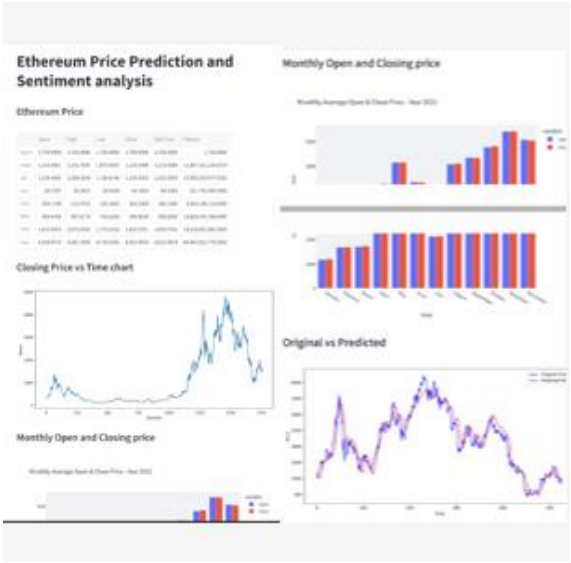


Figure 19 Dashboard sample

7 CONCLUSION & VALUE STATEMENT

The dissertation's primary aim is to set up a correlation between influencer tweets and the impact of each tweet on the Ethereum market price. The impact of each tweet on the market price is analysed, and logical conclusions are drawn from these findings.

The evolution of online technologies generates a large quantity of data, which is then made accessible to internet users. Based on the information offered by these tweets, other sentiment analysis applications, such as reviews, elections, and marketing, may be built. The dissertation subject, "Sentiment analysis on cryptocurrencies," focuses on sentiment analysis for a specific coin using Natural Language Processing (Ethereum). The sentiment analysis uses tweets to identify Ethereum influencers.

Subjective textual elements representing sentimental content could have some demerits. The same term might be considered subjective in one context and objective in another. Domain dependency refers to the fact that the same statement may have various meanings in different domains. Sarcastic phrases unusually convey a negative opinion about a target by employing positive terms. There is a need to examine tweets with and without emotion properly.

Influencer marketing may raise a crypto currency's image, revenue, or reputation in a short period. Therefore, choosing the correct influencer for the right platform and speciality is critical to getting favourable outcomes from marketing efforts. (Shubham Agarwal, 2018)

A dependable source of information must be provided for the project, and numerous feature extraction approaches must be used. Long Short-Term Memory networks, often referred to as "LSTMs," are a specific kind of RNN that can learn long-term dependencies. Bidirectional LSTM is a recurrent neural network primarily used for natural language processing. This project focuses on researching sentiment analysis methods to identify the most critical tweets about the impact of influencer in the trend of ethereum market data sets. We built the LSTM, Bi-directional LSTM and Random Forest Classification Algorithm to classify impacted tweets from a particular Twitter dataset. Our model achieved an efficiency of about 86.3%, corresponding to an F1 score of 0.87. The LSTM model, which achieves 80% accuracy, and the Bidirectional LSTM model, which achieves roughly 84% accuracy, were also built for the best outcomes.

This work may be expanded by creating a Twitter API application that retrieves tweets in real-time using a specified keyword. The Random Forest classifier is based on trees. Multiple classification trees are used to estimate the class label. Using the Tweepy library, we collected daily tweets using selected hashtags.

Limitations:

Compared to a simpler model, such as a 1D conv net, LSTMs have the following disadvantages: LSTMs take longer to train and require more memory. In addition, LSTM easily overfits, and the dropout implementation is complex. However, overfitting can be overcome by the early stopping method. Also, LSTMs have more parameters than other models. These drawbacks can be resolved to an extend by bidirectional LSTM.

All given and discussed works have the common drawback of focusing on improving forecasting performance by using more complex models and approaches while disregarding the construction of a complex training dataset containing more meaningful information.

Here, just the Ethereum currency is examined; however, we must deploy the model generation on additional coins and stocks before determining how the model performs on the various currencies. Cryptocurrencies like bitcoin, Solana, and Tether can also be taken to examine. Each influencer may only be an expert in a single currency; thus, we must examine other currencies for optimal results.

Future scope:

This model can be used to assist ordinary individuals in determining whether or not to invest in the cryptocurrency market. Due to their knowledge in this industry, market influencers and dealers can comprehend the market. However, the average person cannot explain how the market operates. By developing these kinds of neural network models, we may train the machine and do predictive analysis to gain market knowledge. The project focuses mostly on laying the groundwork for future work; if we can update and train this model further, we can obtain a decent result that predicts how tweets will affect the market. According to traders, they know whose tweet will cause a market crash. However, by training the models using various NLP techniques, we can create a good algorithm that can measure the Impact of each influencer's tweets. As a future project, we can use this work to construct a model with data from sources other than Twitter. We cannot assert that Twitter alone influences the market. If we analyse all the data from many sources, we will have a more dependable and valuable input source, allowing us to create a model with the most accurate predictions.

8 LINKEDIN POST

I am adding the LinkedIn post here;

Post 1: https://www.linkedin.com/feed/update/urn:li:activity:6976303665460486144/

Post2: https://www.linkedin.com/feed/update/urn:li:activity:6978062248317460480/

Post3: https://www.linkedin.com/posts/arshad-mk-20b7ba74_eda-dataanalysis-datacleaning-activity-6979156446076641280-hVLh?utm_source=share&utm_medium=member_desktop

9 REFERENCES

Ahmad, H. (2019,, December 5). Exploratory Data Analysis of Cryptocurrency Historical Data | by Hamza Ahmad |. Retrieved from Medium. https://medium.com/@hamzaahmad86/exploratory-data-analysis-of-cryptocurrency-historical-data-d8ec719641e7

Bahrawi. (2019). SENTIMENT ANALYSIS USING RANDOM FOREST ALGORITHM-ONLINE SOCIAL MEDIA BASED. *JOURNAL OF INFORMATION TECHNOLOGY AND ITS UTILIZATION, VOLUME 2, ISSUE 2,*.

Batra, S., & D. Rao. (n.d.). Entity Based Sentiment Analysis on Twitter.

Bekiros, Stelios, & Georgoutsos, Dimitris.;. (2008). Direction-of-change forecasting using a volatility-based recurrent neural network. *Journal of Forecasting.*, 407 - 417.

Breiman L. (2001). Random forests. Mach. Learn.

Crestani, A. G. (Jun. 2016.). *"Like It or Not: A Survey of Twitter Sentiment Analysis Methods," ACM Comput. Surv., vol. 49, no. 2, .*

Deebadi, A. (2020). *Understanding Impact of Twitter Feed on Bitcoin Price and Trading Patterns.* San José State University: The Faculty of the Department of Computer Science.

Demuth, H., Beale, M., De Jess, O., & Hagan, M. (2014.). Neural Network Design; . USA. : Oklahoma State University: Stillwater, OK,.

DEQUAN. (2022). Bitcoin price analysis by Tweets. Bitcoin Price Analysis by Tweets | Kaggle. . Retrieved from https://www.kaggle.com/code/erdeq1024/bitcoin-price-analysis-by-tweets

Derbentsev, V., Matviychuk, A., & Soloviev, V. (2020). Forecasting of Cryptocurrency Prices Using Machine Learning. In Advanced Studies. Springer: Berlin/Heidelberg, Germany.

DUTTA, G. (2020). Bitcoin Price Prediction using Different Models. Bitcoin Price Prediction Using Different Models |. Retrieved from Kaggle. https://www.kaggle.com/code/gauravduttakiit/bitcoin-price-prediction-using-different-models/notebook

Franco Valencia, Alfonso Gómez-Espinosa, & Benjamín Valdés-Aguirre. (2019). Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning. *Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias*.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio. (2016). Y. Deep Learning; Volume 1. In I. Goodfellow, Y. Bengio, A. Courville, & Bengio. USA: MIT Press: Cambridge, MA.

Goyal, G. (2021, June 11). Twitter Sentiment Analysis | Implement Twitter Sentiment Analysis Model. Analytics Vidhya. . Retrieved from https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/

J. Akilandeswari, & G. Jothi. (2018). “Sentiment classification of tweets with non-language features,”. Procedia Comput. Sci.

Kamps, J., M. Marx, R. J. Mokken, & M. De Rijke. (2004.). Using wordnet to measure semantic orientations of adjectives.

Kumar, A. (2020). Twitter Sentiment Analysis. Understanding the sentiments of people. *Analytics Vidhya*. Retrieved from Medium. https://medium.com/analytics-vidhya/twitter-sentiment-analysis-b9a12dbb2043

Livieris, I., E., Kiriakidou, , N., Stavroyiannis, , & S., & Pintelas. (2021). Electronics | Free Full-Text | An Advanced CNN-LSTM Model for Cryptocurrency Forecasting |. Retrieved from https://www.mdpi.com/2079-9292/10/3/287/htm

Livieris, I., Pintelas, E., Stavroyiannis, S., & Pintelas, P. (2020). *Ensemble Deep Learning Models for Forecasting Cryptocurrency.*

Livieris, I., Stavroyiannis, S., Pintelas, E., Kotsilieris, T., & Pintelas, P. A. (2020). dropout weight-constrained recurrent neural network model for forecasting the price of major cryptocurrencies . CCi30 index. Evol. Syst.

M. Bouazizi , & T. Ohtsuki, . (2016.). “Sentiment analysis in twitter: From classification to quantification of sentiments within tweets,”. *IEEE Glob. Commun. Conf. GLOBECOM 2016 - Proc.*

Michael Crosby, N. P. (October 16, 2015). *BlockChain Technology.* Sutardja Center for Entrepreneurship & Technology Technical Report: Beyond Bitcoin.

Pablo Gamallo, & Marcos Garcia, . (2014). Citius: A Naive-Bayes Strategyfor Sentiment Analysis on English Tweets. *8th InternationalWorkshop on Semantic Evaluation* (pp. 171-175.). Dublin, Ireland: SemEval 2014.

Patel, M., Tanwar, S., Gupta, R., & Kumar, N. (2020). A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial. *Institutions. J. Inf. Secur. Appl*.

Pintelas, E., Livieris, I., Stavroyiannis, S., Kotsilieris, T., & Pintelas, P. (2020). Investigating the Problem of Cryptocurrency Price: A Deep Learning Approach. *IFIP International Conference on Artificial Intelligence Applications and Innovations;*, (pp. 99–110). Springer: Berlin/Heidelberg, Germany.

Shubham Agarwal, M. D. (2018). Sentiment Analysis to Evaluate Influencer. *Palarch’s Journal Of*.

Wilson, T., J. Wiebe, , & P. Hoffmann. (Oct 2005). Recognizing contextual polarityin phrase-level sentiment analysis. *in Proceedings of the conferenceon Human Language Technology and Empirical Methods in Natural Language Processing*, (pp. 347-354,).

Zvornicanin, E. (2022). Differences Between Bidirectional and Unidirectional LSTM.

10APPENDIX

The code used for the data-preprocessing and model building are given below.

```
[ ]  #Importing all the essential libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     import tweepy
     from sklearn import preprocessing
     #All the tweets are cleaned and the extracted as csv file and the further cleaning data processing and modelling building are done in
     import re
     from collections import Counter
     import string
     import random
     import seaborn as sns
     from plotly import graph_objs as go
     import plotly.express as px
     import plotly.figure_factory as ff
     #from PIL import Image
     from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
     import nltk
     from nltk.corpus import stopwords
     from tqdm import tqdm
     import os
     import spacy
     from spacy.util import compounding
     from spacy.util import minibatch
```

```
[ ]  #Importing the extracted data from the twitter as csv
     eth_df = pd.read_csv("/content/drive/MyDrive/Dabi_Project/Ethereum_data_24_08.csv")
     eth_df
```

```
[ ]  eth_df1 = eth_df.drop(eth_df.columns[0], axis=1, inplace=True)
     #Looking for the info of time column
     eth_df['time'] = pd.to_datetime(eth_df['time'])
     eth_df.info()
     eth_df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29334 entries, 0 to 29333
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   User       29334 non-null  object
 1   tweets     29334 non-null  object
 2   likes      29334 non-null  int64
 3   time       29334 non-null  datetime64[ns]
 4   retweets   29334 non-null  int64
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 1.1+ MB
```

	User	tweets	likes	time	retweets
0	@VitalikButerin	@lexfridman 😜 \n\nhttps://t.co/JP1Ucky8h6	442	2022-08-23 20:58:21	21
1	@VitalikButerin	If I had to make a bear case for Anglo civiliz...	2378	2022-08-23 15:23:31	250
2	@VitalikButerin	RT @VitalikButerin: @thefaketomato @palladiumm...	0	2022-08-23 15:18:40	50
3	@VitalikButerin	@thefaketomato @palladiummag "X sounds great, ...	367	2022-08-23 15:16:45	50
4	@VitalikButerin	@Max_Stirn @palladiummag Many kinds of technol...	15	2022-08-23 13:56:50	3
...	...	...	...	...	...
29329	@Cyber_FM	Welcome 2021! https://t.co/jS8vCVRX5S GET PAID...	0	2021-02-22 14:30:35	0
29330	@Cyber_FM	Positivity for 2021 from https://t.co/jS8vCVRX...	1	2021-02-22 14:30:34	0
29331	@Cyber_FM	Ahora sonando en CyberFM Latino: Manantial de ...	1	2021-02-22 14:26:24	0
29332	@Cyber_FM	An amazing 2021 w/ https://t.co/jS8vCVRX5S GET...	1	2021-02-22 14:20:27	0
29333	@Cyber_FM	Kick off 2021 w/ https://t.co/jS8vCVRX5S GET P...	0	2021-02-22 14:05:17	0

29334 rows × 5 columns

```
[ ]  #Extract date from the time column
     eth_df['Date']= eth_df['time'].dt.date
     #Cleaning the tweets
     from nltk.tokenize import TweetTokenizer
     from nltk.stem.wordnet import WordNetLemmatizer
     lm = WordNetLemmatizer()

     #Defining a funciton for cleaning the tweets.
     def cleaning(tweets_data):
         #for removing the hashtags of the tweets
         tweets_without_hashtag = re.sub(r'#\w+', ' ', tweets_data)

         #remove the urls of the tweets
         tweet_without_url = re.sub(r'http\S+',' ', tweets_without_hashtag)

         #Remove mentions and characters that not in the English alphabets
         tweets_without_mentions = re.sub(r'@\w+',' ', tweets_without_hashtag)
         first_cleaned_tweets = re.sub('[^A-Za-z]+', ' ', tweets_without_mentions)


         #Tokenize the tweets
         tweet_tokens = TweetTokenizer().tokenize(first_cleaned_tweets)

         #Remove the Punctuations
         tokens_without_punctuations = [w for w in tweet_tokens if w.isalpha()]

         #Removing Stopwords
```

```
        tokens_without_sw = [t for t in tokens_without_punctuations if t not in stop_words]


        #lemmatize the tweets
        text_cleaned = [lm.lemmatize(t) for t in tokens_without_sw]

        #Joining the tweets after cleaning
        return " ".join(text_cleaned)

        return tweet

    #Created a function to clean the tweets
    def edaclean(tweet):
      tweet = re.sub('http', '', tweet) # Remove any hyperlinks
      tweet = re.sub('co', '', tweet)
```

```
[ ] #Further Cleaning for tweets
    # Import nltk package and download the stopwords
    import nltk
    nltk.download('stopwords')
    # We filter out the english language stopwrds
    from nltk.corpus import stopwords
    stop_words = stopwords.words('english')
    print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himsel
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[ ] #Imoporting the nltk library
    nltk.download('wordnet')
    nltk.download('omw-1.4')

    eth_df.tweets = eth_df.tweets.apply(lambda x: cleaning(x))
    eth_df['tweets']=eth_df['tweets'].apply(edaclean)
    eth_df
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

	User	tweets	likes	time	retweets	Date
0	@VitalikButerin	JP Ucky h	442	2022-08-23 20:58:21	21	2022-08-23
1	@VitalikButerin	If I make bear case Anglo civilization would P...	2378	2022-08-23 15:23:31	250	2022-08-23
2	@VitalikButerin	RT X sound great mandatory X sound horrifying ...	0	2022-08-23 15:18:40	50	2022-08-23
3	@VitalikButerin	X sound great mandatory X sound horrifying man...	367	2022-08-23 15:16:45	50	2022-08-23
4	@VitalikButerin	Many kind technology benefit rich people le po...	15	2022-08-23 13:56:50	3	2022-08-23
...	...	...	...	...	...	...
29329	@Cyber_FM	Welme jS vCVRX S GET PAID listening Always F...	0	2021-02-22 14:30:35	0	2021-02-22
29330	@Cyber_FM	Positivity jS vCVRX S GET PAID listening We ...	1	2021-02-22 14:30:34	0	2021-02-22
29331	@Cyber_FM	Ahora sonando en CyberFM Latino Manantial de K...	1	2021-02-22 14:26:24	0	2021-02-22
29332	@Cyber_FM	An amazing w jS vCVRX S GET PAID listening G...	1	2021-02-22 14:20:27	0	2021-02-22
29333	@Cyber_FM	Kick w jS vCVRX S GET PAID listening The Gro...	0	2021-02-22 14:05:17	0	2021-02-22

29334 rows × 6 columns

Cleaning the data based on the ethereum search terms

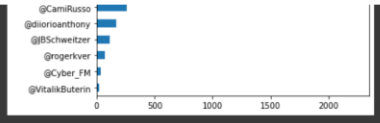
```
[ ] eth_df1 = eth_df.copy()
    #Collecting all the data related to the ethereum search terms
    eth_df2 = eth_df1[(eth_df1['tweets'].str.contains(r'ETH(?!$)')) | (eth_df1['tweets'].str.contains(r'eth(?!$)')) | (eth_df1['tweets'].str.contains(r'Eth(?!$)'))]
    eth_df2
```

	User	tweets	likes	time	retweets	Date
5	@VitalikButerin	In ultra immersive fancy metaverse VR land guy...	6	2022-08-23 13:53:38	0	2022-08-23
18	@VitalikButerin	Actually phrase anti civilization moral critic...	4	2022-08-20 21:44:45	1	2022-08-20
30	@VitalikButerin	None principle literally infinite weight alway...	33	2022-08-20 18:35:58	0	2022-08-20
47	@VitalikButerin	I argue moral perspective N py agent implement...	18	2022-08-19 14:35:44	0	2022-08-19
63	@VitalikButerin	Glad see Ethereum people pushing regulation pr...	3530	2022-08-17 23:43:17	533	2022-08-17
...	...	...	...	...	...	...
29009	@Cyber_FM	Welming jS vCVRX S GET PAID listening The Tu...	0	2021-02-24 09:15:02	0	2021-02-24
29010	@Cyber_FM	Positivity jS vCVRX S GET PAID listening Tog...	0	2021-02-24 08:47:03	0	2021-02-24
29132	@Cyber_FM	Welming jS vCVRX S GET PAID listening TOGETH...	0	2021-02-23 16:50:41	0	2021-02-23
29214	@Cyber_FM	Positivity jS vCVRX S GET PAID listening Tog...	1	2021-02-23 05:19:25	0	2021-02-23
29252	@Cyber_FM	Welming jS vCVRX S GET PAID listening The Tu...	0	2021-02-23 00:22:44	0	2021-02-23

6567 rows × 6 columns

```
[ ] #The distribution of tweet counts based on influencers
    eth_df2.User.value_counts().sort_values().plot(kind = 'barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9947782610>
```




```
[ ] #Sort values by date
    eth_df2.sort_values(by='Date', ascending = False, inplace=True)
    eth_df3 = eth_df2[eth_df2['likes'] != 0]
    eth_df4 = eth_df3[eth_df3['retweets'] != 0]
    eth_df4
```

	User	tweets	likes	time	retweets	Date
14187	@CamiRusso	DeFi liquidation volume v ETH price time gt ha...	61	2022-08-23 13:36:44	12	2022-08-23
20601	@InsiderEthereum	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2022-08-23 11:18:20	2	2022-08-23
20629	@InsiderEthereum	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	2022-08-20 03:18:19	1	2022-08-20
18	@VitalikButerin	Actually phrase anti civilization moral critic...	4	2022-08-20 21:44:45	1	2022-08-20
10983	@cburniske	Cobble lot people together around valuable res...	22	2022-08-18 22:52:46	4	2022-08-18
...	...	...	...	...	...	...
3387	@rogerkver	I would rather sit tub full pig feces smoke me...	3	2013-08-18 08:00:06	4	2013-08-18
3396	@rogerkver	I paid bank USD nversion fee wasted hour bank ...	3	2013-08-14 04:39:43	14	2013-08-14
3402	@rogerkver	Bitin going happen whether regulator want happ...	8	2013-07-31 01:34:03	10	2013-07-31
3437	@rogerkver	kKww NTiEj new user day k user altogether tr...	1	2013-07-04 12:58:36	4	2013-07-04
3464	@rogerkver	If opposed government inflating money pay war ...	3	2013-06-11 16:04:06	11	2013-06-11

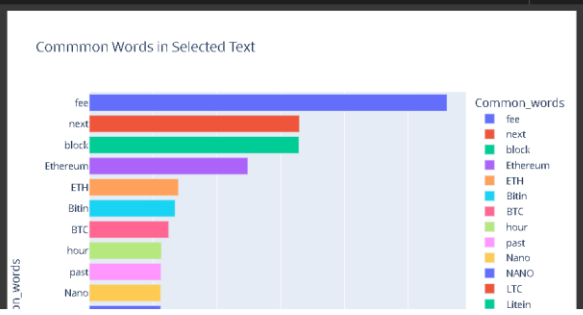
1592 rows × 6 columns

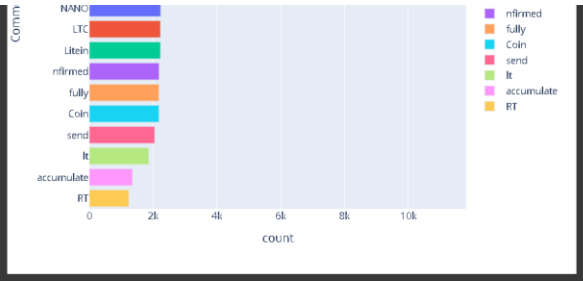
```
[ ] eth_df_wrd = eth_df2.copy()
```

```
[ ] eth_df_wrd['temp_list'] = eth_df_wrd['tweets'].apply(lambda x:str(x).split())
    top = Counter([item for sublist in eth_df_wrd['temp_list'] for item in sublist])
    temp = pd.DataFrame(top.most_common(20))
    temp.columns = ['Common_words','count']
    temp.style.background_gradient(cmap='Blues')
```

	Common_words	count
0	fee	11209
1	next	6583
2	block	6568
3	Ethereum	4967
4	ETH	2792
5	Bitin	2687
6	BTC	2486
7	hour	2260
8	past	2245
9	Nano	2238
10	NANO	2237
11	LTC	2230
12	Litein	2230
13	nfirmed	2187
14	fully	2184
15	Coin	2182
16	send	2051
17	It	1864
18	accumulate	1355
19	RT	1239

```
[ ] fig = px.bar(temp, x="count", y="Common_words", title='Commmon Words in Selected Text', orientation='h',
                 width=700, height=700,color='Common_words')
    fig.show()
```






```
[ ] #Create a function to get the sentiment text
    def getSentiment(score):
      if score < 0:
        return 'negative'
      elif score == 0:
        return 'neutral'
      else:
        return 'positive'
```

```
[ ] #function for subjectivity
    from textblob import TextBlob, Word, Blobber

    def getSubjectivity(twt):
      return TextBlob(twt).sentiment.subjectivity
    #Create a function to get the polarity
    def getPolarity(twt):
      return TextBlob(twt).sentiment.polarity

    #Add teo column as subjectivity and Polarity
    eth_df4['Subjectivity'] = eth_df4['tweets'].apply(getSubjectivity)
    eth_df4['Polarity'] = eth_df4['tweets'].apply(getPolarity)

    # Create a column to store the text sentiment
    eth_df4['Sentiment'] = eth_df4['Polarity'].apply(getSentiment)

    eth_df4.head()
```

	User	tweets	likes	time	retweets	Date	Subjectivity	Polarity	Sentiment
14187	@CamiRusso	DeFi liquidation volume v ETH price time gt ha...	61	2022-08-23 13:36:44	12	2022-08-23	0.541667	-0.291667	negative
20601	@InsiderEthereum	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2022-08-23 11:18:20	2	2022-08-23	0.333333	0.333333	positive
20629	@InsiderEthereum	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	2022-08-20 03:18:19	1	2022-08-20	0.000000	0.000000	neutral
18	@VitalikButerin	Actually phrase anti civilization moral critic...	4	2022-08-20 21:44:45	1	2022-08-20	0.175000	0.000000	neutral
10983	@cburniske	Cobble lot people together around valuable res...	22	2022-08-18 22:52:46	4	2022-08-18	1.000000	1.000000	positive

```
[ ]
```

Close price analysis and prediction of Ethereum

```
[ ] #Importing the current market values inline with the tweets
    close_eth = pd.read_csv("/content/drive/MyDrive/Dabi_Project/ETH-USD_24.csv")
    close_eth_full = pd.read_csv("/content/drive/MyDrive/Dabi_Project/ETH-USD.csv")
    close_eth_full1 = close_eth_full.copy()
    close_eth_full
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
1	2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2	2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
3	2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
4	2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984
...	...	...	...	...	...	...	...
1754	2022-08-29	1430.439453	1556.309570	1427.728394	1553.037354	1553.037354	17965837488
1755	2022-08-30	1553.188965	1600.461182	1480.831787	1523.838867	1523.838867	21835784470
1756	2022-08-31	1524.286499	1612.358887	1524.286499	1553.684937	1553.684937	20591680941
1757	2022-09-01	1553.756348	1593.082764	1520.188354	1586.176758	1586.176758	16434276817
1758	2022-09-02	1583.881592	1602.590088	1571.791260	1596.431030	1596.431030	16078590976

1759 rows × 7 columns

```
[ ] close_eth_full.reset_index()
    close_eth_full = close_eth_full.drop(['Date', "Adj Close"], axis = 1)
    close_eth_full.head()
```

	Open	High	Low	Close	Volume
0	308.644989	329.451996	307.056000	320.884003	893249984
1	320.670990	324.717987	294.541992	299.252991	885985984
2	298.585999	319.453003	298.191986	314.681000	842300992
3	314.690002	319.153015	298.513000	307.907990	1613479936
4	307.024994	328.415009	307.024994	316.716003	1041889984

```
plt.plot(close_eth_full.Close)
plt.title('Ethereum close Price')
plt.xlabel('Quantity')
plt.ylabel('Values')
plt.show()
```

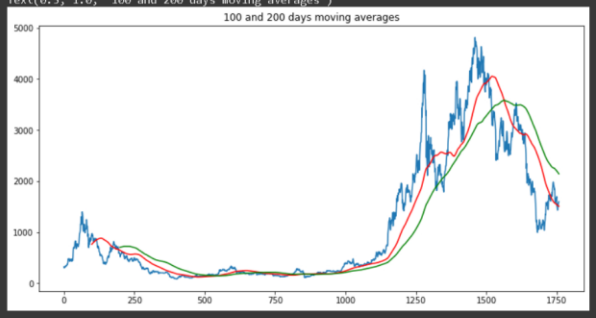
```
#Creating the moving averages for next 100 days
ma100 = close_eth_full.Close.rolling(100).mean()
ma100
```

```
0              NaN
1              NaN
2              NaN
3              NaN
4              NaN
            ...
1754    1522.001680
1755    1516.808367
1756    1512.623398
1757    1508.695338
1758    1505.211369
Name: Close, Length: 1759, dtype: float64
```

```
plt.figure(figsize = (12,6))
plt.plot(close_eth_full.Close)
plt.plot(ma100, 'r')
plt.title('100 days moving averages')
```

```
#Creating the 200 days moving averages
ma200 = close_eth_full.Close.rolling(200).mean()
ma200
plt.figure(figsize = (12,6))
plt.plot(close_eth_full.Close)
plt.plot(ma100, 'r')
plt.plot(ma200, 'g')
plt.title('100 and 200 days moving averages')
```

```
Text(0.5, 1.0, '100 and 200 days moving averages')
```




```
close_eth_full1['Date'] = pd.to_datetime(close_eth_full1['Date'], format='%Y-%m-%d')
year_2021 = close_eth_full1.loc[(close_eth_full1['Date'] >= '2021-01-01')
                        & (close_eth_full1['Date'] < '2022-01-01')]
year_2021.drop(year_2021[['Open']],axis=1)
months_2021 = year_2021.groupby(year_2021['Date'].dt.strftime('%B'))[['Open','Close']].mean()
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
               'September', 'October', 'November', 'December']
months_2021 = round(months_2021.reindex(month_order, axis=0),2)
months_2021 = months_2021.reset_index()
```

```
#Taking only the 2021 data of ethereum
year_2021
```

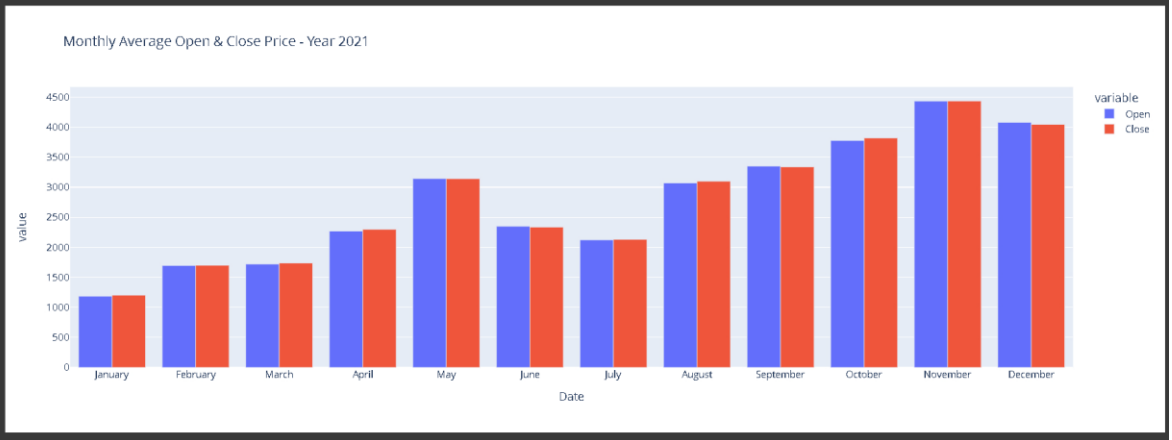
	Date	Open	High	Low	Close	Adj Close	Volume
1149	2021-01-01	737.708374	749.201843	719.792236	730.367554	730.367554	13652004358
1150	2021-01-02	730.402649	786.798462	718.109497	774.534973	774.534973	19740771179
1151	2021-01-03	774.511841	1006.565002	771.561646	975.507690	975.507690	45200463368
1152	2021-01-04	977.058838	1153.189209	912.305359	1040.233032	1040.233032	56945985763
1153	2021-01-05	1041.498779	1129.371460	986.811279	1100.006104	1100.006104	41535932781
...	...	...	...	...	...	...	...
1509	2021-12-27	4064.746338	4126.001465	4033.492432	4037.547607	4037.547607	11424360002
1510	2021-12-28	4037.538086	4037.538086	3769.280029	3800.893066	3800.893066	17299472803
1511	2021-12-29	3797.436279	3827.981934	3612.795898	3628.531738	3628.531738	15722555672
1512	2021-12-30	3632.219727	3767.559814	3595.204834	3713.852051	3713.852051	12925377999
1513	2021-12-31	3713.430176	3807.288818	3636.869873	3682.632813	3682.632813	14157285268

365 rows × 7 columns

```
#Printing Ethereum price over period by date
fig = px.line(close_eth_full1, x="Date", y="Close", title='Ethereum Close price over time')
fig.show()
```

Ethereum Close price over time


```
[ ] #Monthly Average Open & Close Price - Year 2021
    import plotly.express as px
    fig = px.bar(months_2021, x='Date', y=['Open','Close'], barmode='group', title = 'Monthly Average Open & Close Price - Year 2021')
    fig.show()
```



Model building for Ethereum price

```
[ ] #Spliting data to training and testing
    data_training = pd.DataFrame(close_eth_full['Close'][0:int(len(close_eth_full)*0.70)])
    data_testing = pd.DataFrame(close_eth_full['Close'][int(len(close_eth_full)*0.70): int(len(close_eth_full))])
    print(data_training.shape)
    print(data_testing.shape)
```

```
(1231, 1)
(528, 1)
```

```
[ ] #For data scaling
    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler(feature_range=(0,1))
    #After scaling the data
    data_training_array = scaler.fit_transform(data_training)
    data_training_array
```

```
array([[0.1261161 ],
       [0.11458483],
       [0.12280934],
       ...,
       [0.9083364 ],
       [0.85668903],
       [0.84992741]])
```

```
[ ] x_train = []
    y_train = []

    for i in range(100, data_training_array.shape[0]):
      x_train.append(data_training_array[i-100: i])
      y_train.append(data_training_array[i, 0])

    x_train, y_train = np.array(x_train), np.array(y_train)
```

```
[ ] #Importing  model
    from keras.layers import Dense, Dropout, LSTM
    from keras.models import Sequential
```

```
[ ] model = Sequential()
    model.add(LSTM(units=50, activation = 'relu', return_sequences = True,
                   input_shape = (x_train.shape[1],1)))
    model.add(Dropout(0.2))
```

```
model.add(LSTM(units=60, activation = 'relu', return_sequences = True))
model.add(Dropout(0.3))


model.add(LSTM(units=80, activation = 'relu', return_sequences = True))
model.add(Dropout(0.4))

model.add(LSTM(units=120, activation = 'relu'))
model.add(Dropout(0.5))

#model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.add(Dense(units = 1))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 60)           26640

 dropout_1 (Dropout)         (None, 100, 60)           0

 lstm_2 (LSTM)               (None, 100, 80)           45120

 dropout_2 (Dropout)         (None, 100, 80)           0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178,761
Trainable params: 178,761
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs = 50)
```

```
#Saved the model after model building
model.save('keras_model.h5')
```

```
data_testing.head()
```

	Close
1231	1593.413452
1232	1595.359253
1233	1702.842041
1234	1716.494629
1235	1691.355957

```
past_100_days = data_training.tail(100)
final_df = past_100_days.append(data_testing, ignore_index=True)
```

```
input_data = scaler.fit_transform(final_df)
input_data
```

```
input_data.shape
```

```
(628, 1)
```

```
x_test = []
y_test = []

for i in range(100, input_data.shape[0]):
  x_test.append(input_data[i-100: i])
  y_test.append(input_data[i, 0])

x_test, y_test = np.array(x_test), np.array(y_test)
print(x_test.shape)
print(y_test.shape)
```

```
(528, 100, 1)
(528,)
```

```
y_predicted = model.predict(x_test)
y_predicted.shape
y_predicted
```

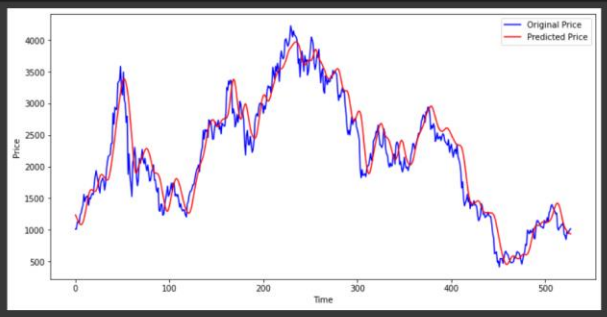
```
scaler.scale_
```

```
array([0.0002365])
```

```
scale_factor = 1/0.0002365
y_predicted = y_predicted * scale_factor
y_test = y_test * scale_factor
```

```
plt.figure(figsize=(12,6))
plt.plot(y_test, 'b', label = 'Original Price')
```

```
plt.plot(y_predicted, 'r', label = 'Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```




```
[ ] close_eth2 = pd.read_csv("/content/drive/MyDrive/Dabi_Project/ETH-USD_2015_with_diff.csv")
    close_eth2
```

	Date	Open	Close	Price_diff
0	09-11-2017	308.644989	320.884003	-12.239014
1	10-11-2017	320.670990	299.252991	21.417999
2	11-11-2017	298.585999	314.681000	-16.095001
3	12-11-2017	314.690002	307.907990	6.782012
4	13-11-2017	307.024994	316.716003	-9.691009
...	...	...	...	...
1750	25-08-2022	1657.336548	1696.457031	-39.120483
1751	26-08-2022	1696.324585	1507.782837	188.541748
1752	27-08-2022	1508.156982	1491.395020	16.761962
1753	28-08-2022	1491.206787	1430.547363	60.659424
1754	29-08-2022	1432.359131	1536.331665	-103.972534

1755 rows × 4 columns

```
[ ] #Converting the date column into date time
    close_eth2['Date'] = pd.to_datetime(close_eth2['Date'])
    close_eth2.sort_values(by='Date', ascending = False, inplace=True)
    close_eth2
```

```
[ ] eth_df5 = eth_df4.copy()
```

```
[ ] #Joining the market value closed price inline with the tweets
    eth_df5 = eth_df5.set_index('Date').join(close_eth2.set_index('Date'))
    eth_df5
```

Date	User	tweets	likes	time	retweets	Subjectivity	Polarity	Sentiment	Open	Close	Price_diff
2013-06-11	@rogerkver	If opposed government inflating money pay war ...	3	2013-06-11 16:04:06	11	0.500000	0.250000	positive	NaN	NaN	NaN
2013-07-04	@rogerkver	kKww NTiEj new user day k user altogether tr...	1	2013-07-04 12:58:36	4	0.454545	0.136364	positive	NaN	NaN	NaN
2013-07-31	@rogerkver	Bitin going happen whether regulator want happ...	8	2013-07-31 01:34:03	10	0.000000	0.000000	neutral	NaN	NaN	NaN
2013-08-14	@rogerkver	I paid bank USD nversion fee wasted hour bank ...	3	2013-08-14 04:39:43	14	0.400000	0.100000	positive	NaN	NaN	NaN
2013-08-18	@rogerkver	I would rather sit tub full pig feces smoke me...	3	2013-08-18 08:00:06	4	0.550000	0.350000	positive	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
2022-08-18	@cburniske	Cobble lot people together around valuable res...	22	2022-08-18 22:52:46	4	1.000000	1.000000	positive	1833.715576	1847.007813	-13.292237
2022-08-20	@InsiderEthereum	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	2022-08-20 03:18:19	1	0.000000	0.000000	neutral	1612.650635	1577.003784	35.646851
2022-08-20	@VitalikButerin	Actually phrase anti civilization moral critic...	4	2022-08-20 21:44:45	1	0.175000	0.000000	neutral	1612.650635	1577.003784	35.646851
2022-08-23	@CamiRusso	DeFi liquidation volume v ETH price time gt ha...	61	2022-08-23 13:36:44	12	0.541667	-0.291667	negative	1622.939331	1662.769897	-39.830566
2022-08-23	@InsiderEthereum	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2022-08-23 11:18:20	2	0.333333	0.333333	positive	1622.939331	1662.769897	-39.830566

1592 rows × 11 columns

```
[ ] eth_df6 = eth_df5.dropna()
    eth_df6.reset_index(inplace = True)
```

```
[ ] column_names2 = ['User', "Date", "tweets", "likes", 'retweets', 'Close', 'Price_diff']

    eth_df7 = eth_df6.reindex(columns=column_names2)
```

```
eth_df7.head()
```

	User	Date	tweets	likes	retweets	Close	Price_diff
0	@ethereum	2017-04-12	Developer Update Ethereum Core Developers Meet...	89	35	470.204010	-4.150024
1	@rogerkver	2017-11-12	Vitalik genius class act That I sold portion B...	1643	464	515.135986	-74.777984
2	@ethereum	2017-11-17	LIVE Ethereum Core Devs Meeting OK bkahmiR	126	68	332.394012	-2.227020
3	@ethereum	2017-11-17	Notes Ethereum Core Devs Meeting qgpgwfn cf	198	96	332.394012	-2.227020
4	@ethereum	2017-12-15	LIVE Ethereum Core Devs Meeting sKGBGKjRKJ	239	135	684.447998	11.927979

```
[ ] eth_df7['Percent']=eth_df7[['Price_diff']].div(eth_df7.Close, axis=0)*100
    eth_df7.sort_values(by='Date', ascending=False, inplace=True)
    eth_df7.head()
```

	User	Date	tweets	likes	retweets	Close	Price_diff	Percent
820	@InsiderEthereum	2022-08-23	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2	1662.769897	-39.830566	-2.395435
819	@CamiRusso	2022-08-23	DeFi liquidation volume v ETH price time gt ha...	61	12	1662.769897	-39.830566	-2.395435
818	@VitalikButerin	2022-08-20	Actually phrase anti civilization moral critic...	4	1	1577.003784	35.646851	2.260416
817	@InsiderEthereum	2022-08-20	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	1	1577.003784	35.646851	2.260416
816	@cburniske	2022-08-18	Cobble lot people together around valuable res...	22	4	1847.007813	-13.292237	-0.719663

```
[ ] for index, row in eth_df7.iterrows():
        if eth_df7.loc[index, 'Percent'] < 0:
            eth_df7.loc[index,'Impact'] = 'Negative Impact'
        elif eth_df7.loc[index, 'Percent'] >= 0 and eth_df7.loc[index, 'Percent'] <= 2 :
            eth_df7.loc[index,'Impact'] = 'Neutral Impact'
        else:
          eth_df7.loc[index, 'Impact'] = 'Positive Impact'
    eth_df7.head()
```

	User	Date	tweets	likes	retweets	Close	Price_diff	Percent	Impact
820	@InsiderEthereum	2022-08-23	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2	1662.769897	-39.830566	-2.395435	Negative Impact
819	@CamiRusso	2022-08-23	DeFi liquidation volume v ETH price time gt ha...	61	12	1662.769897	-39.830566	-2.395435	Negative Impact
818	@VitalikButerin	2022-08-20	Actually phrase anti civilization moral critic...	4	1	1577.003784	35.646851	2.260416	Positive Impact
817	@InsiderEthereum	2022-08-20	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	1	1577.003784	35.646851	2.260416	Positive Impact
816	@cburniske	2022-08-18	Cobble lot people together around valuable res...	22	4	1847.007813	-13.292237	-0.719663	Negative Impact

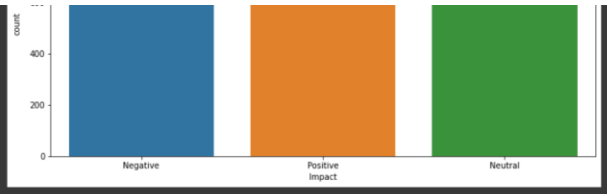
```
[ ] #function for subjectivity
    def getSubjectivity(twt):
      return TextBlob(twt).sentiment.subjectivity
    #Create a function to get the polarity
    def getPolarity(twt):
      return TextBlob(twt).sentiment.polarity

    #Add teo column as subjectivity and Polarity
    eth_df7['Subjectivity'] = eth_df7['tweets'].apply(getSubjectivity)
    eth_df7['Polarity'] = eth_df7['tweets'].apply(getPolarity)


    #Create a function to get the sentiment text
    def getSentiment(score):
      if score < 0:
        return 'negative'
      elif score == 0:
        return 'neutral'
      else:
        return 'positive'

    # Create a column to store the text sentiment
    eth_df7['Sentiment'] = eth_df7['Polarity'].apply(getSentiment)
    eth_df7.sort_values(by='Date', ascending=False, inplace=True)
    eth_df7.head()
```

	User	Date	tweets	likes	retweets	Close	Price_diff	Percent	Impact	Subjectivity	Polarity	Sentiment
820	@InsiderEthereum	2022-08-23	Most PoW miner intend mine Ergo Ravenin Ethere...	6	2	1662.769897	-39.830566	-2.395435	Negative Impact	0.333333	0.333333	positive
819	@CamiRusso	2022-08-23	DeFi liquidation volume v ETH price time gt ha...	61	12	1662.769897	-39.830566	-2.395435	Negative Impact	0.541667	-0.291667	negative
818	@VitalikButerin	2022-08-20	Actually phrase anti civilization moral critic...	4	1	1577.003784	35.646851	2.260416	Positive Impact	0.175000	0.000000	neutral
817	@InsiderEthereum	2022-08-20	Price analysis BTC ETH BNB XRP ADA SOL DOGE DO...	2	1	1577.003784	35.646851	2.260416	Positive Impact	0.000000	0.000000	neutral
816	@cburniske	2022-08-18	Cobble lot people together around valuable res...	22	4	1847.007813	-13.292237	-0.719663	Negative Impact	1.000000	1.000000	positive

```
[ ]
```

```
[ ] plt.figure(figsize=(12,6))
    sns.countplot(x='Impact',data=eth_df7)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f33a2cc5690>
```




```
[ ] import wordcloud
    common_words=''
    for i in eth_df7.tweets:
        i = str(i)
        tokens = i.split()
        common_words += " ".join(tokens)+" "
    wordcloud = wordcloud.WordCloud().generate(common_words)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
```


```
[ ]
```

Model Building

```
[ ] import tensorflow as tf
    from tensorflow.keras.preprocessing.text import Tokenizer
    from tensorflow.keras.preprocessing.sequence import pad_sequences
    import tensorflow.keras.layers as Layers
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.losses import SparseCategoricalCrossentropy
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM, Conv1D, GlobalMaxPooling1D, Bidirectional, SpatialDropout1D
    from tensorflow.keras.models import load_model


    from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
```

```
[ ] X = eth_df7['tweets']
    y = pd.get_dummies(eth_df7['Impact']).values
    num_classes = eth_df7['Impact'].nunique()
```

```
[ ] seed = 38
    np.random.seed(seed)

    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                        test_size=0.2)
    print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
    (656,) (165,) (656, 3) (165, 3)
```

```
[ ] max_features = 20000
    tokenizer = Tokenizer(num_words=max_features)
    tokenizer.fit_on_texts(list(X_train))
    X_train = tokenizer.texts_to_sequences(X_train)
    X_test = tokenizer.texts_to_sequences(X_test)
```

```
[ ] from tensorflow.keras.preprocessing import sequence
    max_words = 30
    X_train = sequence.pad_sequences(X_train, maxlen=max_words)
    X_test = sequence.pad_sequences(X_test, maxlen=max_words)
    print(X_train.shape,X_test.shape)
```

```
    (656, 30) (165, 30)
```

LSTM Model

```
[ ] import tensorflow.keras.backend as K
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense,Embedding,Conv1D,MaxPooling1D,LSTM
    from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

    batch_size = 128
    epochs = 10

    max_features = 20000
    embed_dim = 100

    np.random.seed(seed)
    K.clear_session()
    model = Sequential()
    model.add(Embedding(max_features, embed_dim, input_length=X_train.shape[1]))
    model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
```

```
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 30, 100)           2000000

 conv1d (Conv1D)             (None, 30, 32)            9632

 max_pooling1d (MaxPooling1D  (None, 15, 32)           0
 )

 conv1d_1 (Conv1D)           (None, 15, 32)            3104

 max_pooling1d_1 (MaxPooling  (None, 7, 32)            0
 1D)

 lstm (LSTM)                 (None, 100)               53200

 dense (Dense)               (None, 3)                 303

=================================================================
Total params: 2,066,239
Trainable params: 2,066,239
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] sgd = SGD(learning_rate=learning_rate, momentum=momentum, decay=decay_rate, nesterov=False)
    # Build model
    model= Sequential()
    model.add(Embedding(vocab_size, embedding_size, input_length=max_len))
    model.add(Conv1D(filters=32, kernel_size=1, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Bidirectional(LSTM(32)))
    model.add(Dropout(0.4))
    model.add(Dense(3, activation='softmax'))
```

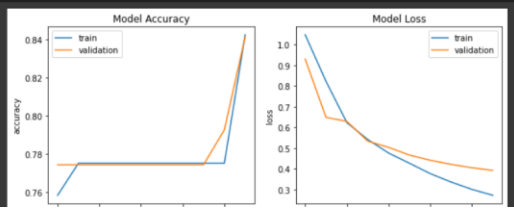
```
[ ] from keras.callbacks import ModelCheckpoint, EarlyStopping
    earlyStop=EarlyStopping(monitor="val_loss",verbose=2,mode='min',patience=3)
    history1=model.fit(X_train,y_train,epochs=10,batch_size=10,validation_data=(X_test,y_test) ,verbose=2,callbacks=[earlyStop])
```

```
Epoch 1/10
66/66 - 7s - loss: 0.9981 - accuracy: 0.5046 - val_loss: 0.9024 - val_accuracy: 0.5636 - 7s/epoch - 113ms/step
Epoch 2/10
66/66 - 2s - loss: 0.8264 - accuracy: 0.6113 - val_loss: 0.9002 - val_accuracy: 0.5515 - 2s/epoch - 36ms/step
Epoch 3/10
66/66 - 2s - loss: 0.6797 - accuracy: 0.6738 - val_loss: 1.0426 - val_accuracy: 0.4970 - 2s/epoch - 35ms/step
Epoch 4/10
66/66 - 2s - loss: 0.4052 - accuracy: 0.8338 - val_loss: 1.2957 - val_accuracy: 0.4364 - 2s/epoch - 35ms/step
Epoch 5/10
66/66 - 2s - loss: 0.0972 - accuracy: 0.9787 - val_loss: 2.1674 - val_accuracy: 0.4364 - 2s/epoch - 37ms/step
Epoch 5: early stopping
```

Bidirectional LSTM

```
[ ] sgd = SGD(learning_rate=learning_rate, momentum=momentum, decay=decay_rate, nesterov=False)
    # Build model
    model= Sequential()
    model.add(Embedding(vocab_size, embedding_size, input_length=max_len))
    model.add(Conv1D(filters=32, kernel_size=1, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Bidirectional(LSTM(32)))
    model.add(Dropout(0.4))
    model.add(Dense(3, activation='softmax'))
```

```
def plot_training_hist(history):
    '''Function to plot history for accuracy and loss'''

    fig, ax = plt.subplots(1,2, figsize=(10,4))
    # first plot
    ax[0].plot(history.history['accuracy'])
    ax[0].plot(history.history['val_accuracy'])
    ax[0].set_title('Model Accuracy')
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('accuracy')
    ax[0].legend(['train', 'validation'], loc='best')

    # second plot
    ax[1].plot(history.history['loss'])
    ax[1].plot(history.history['val_loss'])
    ax[1].set_title('Model Loss')
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('loss')
    ax[1].legend(['train', 'validation'], loc='best')

plot_training_hist(history)
```




```
[ ] # predict class with test set
    y_pred_test =  np.argmax(model.predict(X_test), axis=1)
    print('Accuracy:\t{:0.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_test)*100))
    print(classification_report(np.argmax(y_test,axis=1), y_pred_test))
```

```
Accuracy:       84.1%
              precision    recall  f1-score   support

           0       1.00      0.31      0.48        16
           1       1.00      0.29      0.44        21
           2       0.83      1.00      0.91       127

    accuracy                           0.84       164
   macro avg       0.94      0.53      0.61       164
weighted avg       0.87      0.84      0.81       164
```

Random Forest Classifier

```
[ ] from sklearn.ensemble import RandomForestClassifier


    rfc=RandomForestClassifier(n_estimators= 100, random_state= seed)
    rfc.fit(X_train, y_train)
    predictions = rfc.predict(X_test)
```

```
    print(classification_report(y_test, predictions))
    #print(confusion_matrix(y_test, predictions))



    #rfc_f1 = round(f1_score(y_test, predictions, average = 'weighted'), 3)
    rfc_accuracy = round((accuracy_score(y_test, predictions) * 100), 2)



    print("Accuracy : " , rfc_accuracy , " %")
    #print("f1_score : " , rfc_f1)
```

```
              precision    recall  f1-score   support

           0       0.88      0.28      0.42        25
           1       0.67      0.61      0.64        49
           2       0.94      0.94      0.94       374

   micro avg       0.91      0.87      0.89       448
   macro avg       0.83      0.61      0.67       448
weighted avg       0.91      0.87      0.88       448
 samples avg       0.87      0.87      0.87       448

Accuracy :  86.83  %
```