Decision-Based Deep Learning Model for ROI Maximization in Bitcoin Investment Using Emotion Classification

by

Jacob Sandiumenge Torres

This thesis has been submitted in partial fulfillment for the degree of Master of Science in Artificial Intelligence

in the
Faculty of Engineering and Science
Department of Computer Science

May 2025

Declaration of Authorship

This report, Decision-Based Deep Learning Model for ROI Maximization in Bitcoin Investment Using Emotion Classification, is submitted in partial fulfillment of the requirements of Master of Science in Artificial Intelligence at Munster Technological University Cork. I, Jacob Sandiumenge Torres, declare that this thesis titled, Decision-Based Deep Learning Model for ROI Maximization in Bitcoin Investment Using Emotion Classificationand the work represents substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged. I confirm that:

- This work was done wholly or mainly while in candidature Master of Science in Artificial Intelligence at Munster Technological University Cork.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University Cork or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Jacob Sandiumenge Torres

Date: 12/05/2025

MUNSTER TECHNOLOGICAL UNIVERSITY CORK

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Jacob Sandiumenge Torres

Cryptocurrencies have sparked controversy in recent years, not only as a competitor to the traditional currencies, but also as an investment due to its impressive increases in value. Consequently, people have been trying to find means to beat the market and improve its investment capabilities using statistical or deep learning-based models to predict its price fluctuations. A common approach for that has been to use Sentiment Analysis from Twitter data (currently X) due to its high volatility and corelation to its public perception, acting as a 'hype' investment. However, although there are many studies on this matter, most of them approach mimicking the price regression instead of actually attempting to beat the market generating a better Return on Investment (ROI) instead or using a very basic and oversimplified Sentiment Analysis, making its relevance towards the model minimal. Because of that, this work proposes to make a decision model with the focus on maximizing the ROI for Bitcoin investments and to solve the poor influence of the Sentiment Analysis by replacing it for a broader Emotion Classification and other inherent Twitter metrics to give the model more relevant information.

Acknowledgements

I would like to thank my master-thesis supervisor, Dr Vincent C. Emeakaroha (MTU), for his helpful guidance throughout my research project.

Contents

List of Figures

List of Tables

Abbreviations

ROI	**R**eturn **O**n **I**nvestment
LSTM	**L**ong **S**hort **T**erm **M**odel
ARIMA	**A**uto**R**egressive **I**ntegrated **M**oving **A**verage
GARCH	**G**eneralized **A**uto**R**egressive **C**onditional **H**eteroskedasticity
RNN	**R**ecurrent **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
NLP	**N**atural **L**anguage **P**rocessing
API	**A**pplication **P**rogramming **I**nterface
NFT	**N**on-**F**ungible **T**oken
TFT	**T**emporal **F**usion **T**ransformer
CAPTCHA	**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
BTC	**B**i**TC**oin
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
RoBERTa	**R**obustly **o**ptimized **BERT** **a**pproach
LLM	**L**arge **L**anguage **M**odel
IQR	**I**nter**Q**uartile **R**ange

Chapter 1

Introduction

Cryptocurrencies have slowly become an alternative to traditional centralized currencies by having a free market structure with inherent transparency and fixed coin pool size to avoid market manipulation.

However, one of the things that made them the most famous is the speculation on their price driven by their current heavy inflation. That made them quite appealing for the investing community making them raise even more in price and are today some of the most common risky investments to have, making it so that even governments hold them. For example, according to *Forbes*, the US government currently holds approximately 207,000 bitcoins, which is around a 2.3% of the total pool and about 20 billion dollars in today's exchange.

So, as with any other investment, people rushed to find patterns and prediction models to predict its price fluctuations and beat the market. There have been a lot of different approaches to get an accurate predictor, going from purely statistical approaches up to complex transformer models but for this study the one that will be used is the use of people's emotions on Twitter during the time periods and the relevance it has. This is a method that has been widely used by many people before due to the high volatility and mood relationship that cryptocurrencies are leaded by.

1.1 Motivation

Although there is quite a lot of work regarding this topic, most of it is quite poor, and mainly regression-focused, making it more of a coding exercise than an investment-viable solution. This motivated me to write my own research addressing the main focus points that I consider relevant, correcting common mistakes in the broad majority of

studies and having a clear interpretation of the data and the results gathered for a better understanding and expandability.

1.2 Contribution

The idea behind this project is to make a more robust line of focus, aiming for an actually viable investment decision model and assessing whether it is possible or not to use it to have an advantage over the market instead of aiming for a price regression prediction.

Also this project uses emotion classification instead of basic sentiment analysis, in an attempt to get a more broad understanding of the human perception. The main reason behind this is the fact that some emotions that are classified as negative have different connotations. For example, we understand fear as something that could lead people to sell the investment, while we understand anger more as a reaction after the price has already dropped, as the social psychologist Jennifer S. Lerner explains in her journals [1][2].

In addition, this project uses a new means to obtain the number of daily tweets as a new metric and specific classification models for crypto tweets for a better understanding of the fed data.

1.3 Structure of This Document

The basic structure of this document starts with the literature review, where it explains what the trend on this kind of studies is, the state of the art and what mistakes they tend to make. Next, in the design section, there is a brief explanation of the structure that follows the project. Then it gets explained and completed in the implementation section each step at the time, starting with the data extraction, followed by the pre-processing of that data and finally making the model. And it ends with the testing and evaluation of the results from the model.

Chapter 2

Literature Review

Predicting future events has always been a field of study with a broad focus due to its relevance in economics. There is a lot of work on this subject in the stock market and currencies and, with a broad growth in popularity in recent years, cryptocurrencies.

The first attempts on it were mainly statistical predictions, such as autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroskedasticity (GARCH) [3][4]. But due to the growth of AI, deep learning models became the main choice, allowing to also use Natural Language Processing (NLP) and pattern recognition when using big datasets for its training. One of the first models used was Recurrent Neural Networks (RNNs), later shifting toward Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models to avoid the gradient vanishing, which improve learning speed and handle longer time steps more effectively. [5][6][7].

Due to the high volatility in cryptocurrencies, a widely used indicator to predict its price fluctuations has been sentiment analysis, most of it done over Twitter posts [8][9]. This method usually scores tweets daily, generating a sentiment gradient as a market mood indicator and using it as an indicator to train the model.

From here, researchers have taken multiple paths to try to get an efficient predictor but their parts can mainly be broken down in three steps: data extraction, sentiment analysis and the prediction model. Despite that, there are some flaws or research gaps in some of those that can be further studied to obtain a more robust model.

2.1 Data Extraction

The main method used by almost every study is API scraping as Twitter used to give access to researchers for research-based studies. But in recent years, due to the hard

privatization of the data after Elon Musk bought the company on April 2022, the API access became private and solely for business use, making the scraping a lot more difficult and making most studies rely on previously scraped datasets.

2.2 Sentiment Analysis and Emotion Classification

Sentiment Analysis

Most of the studies regarding this topic of identifying the public feelings about cryptocurrencies using Twitter data have focused mainly on sentiment analysis, a gradient between "positive" and "negative" feelings about the topic. Although this statistic can give us some information relevant for the predictions, it is very narrow and gives barely any information to be considered for the training of the model. Consequently, most studies that use it tend to overload the training inputs with other statistics such as "the average price, opening price, highest price, lowest price, and trading volume" [10] updated daily, most of them redundant and not adding much information and having only one metric to symbolize the sentiment metric.

Emotion Classification

A more complex approach that has been used in some more recent studies has been the use of Emotion Classification to give a more broad description of the public's perception of the topic. It has been seen in many psychological and statistical studies that different emotions lead to different actions and risk taking further than the dichotomy of them being good or bad. For example, according to Dr. Jennifer Lerner "fearful people make more pessimistic risk assessments, whereas angry people make more optimistic risk assessments" [1][2]. This implies a relationship between human behavior and decision making, making it a very good candidate for further exploration. It also gives a more complex categorization of the feelings of the public, therefore making it more useful than a simple sentiment analysis.

There has been some research on the topic for stock market predictions with a study on statistics [11] and a machine learning model but only tested for short time spans as the days before an election [12]. It has also been used for NFT price predictions [13]. Finally, in the field of cryptocurrency there are also a few studies using emotion analysis [14] but they are mainly focusing on the emotion analysis evaluation alone instead of prediction models.

2.3 Decision Model

Different models

The last step is the prediction model. Because of it being a simple step to implement, many studies check the performance on multiple models to assess their performance instead of limiting themselves on just one. In most cases the best result ends up being LSTM thus making it the most used model for the latest coming studies due to its long-term forecasting properties.

Another model that has become popular just recently is Temporal Fusion Transform (TFT), which combines LSTM and attention mechanisms, providing interpretability and strong performance on temporal data. This model is outperforming the regular LSTM in many fields [15], thus becoming their current state of the art. Some work has already been done for cryptocurrencies [16] and the results are quite promising, so it is a good option to consider but it requires a lot more training data than the other models, so it only starts making sense for large scale implementations.

Results and Training Focus

A different feature to take into account is the type of the output generated. While most studies are trained to give a price prediction each day for the following day [6], some provide action recommendations like "Buy", "Sell" or "Hold".

However, a big problem that almost all those models have is that they are trained to "mimic" the current regression accounting for the amount of correct results and aiming the model into maximizing those instead of training it to maximize the Return on Investment (ROI). What this leads to is a model that, instead of predicting, adapts to the fluctuations of the currency after it changes. That isn't the expected behavior when dealing with an investment as the main goal for it is to anticipate it to get the value from holding or avoid a loss when selling.

For this reason, the logical training focus should be to maximize the ROI as it is the final goal of the prediction itself. However, this focus is barely used in most cryptocurrency predictors and in the few that consider it, most of the time the model is still trained on regression mimicking and just evaluated afterwards [6].

However, this concept is present in many studies of the stock market giving us insights of its reliability [17][11][12], and a handful of studies related to bitcoin, but with quite questionable practices and disturbingly good returns, for example with the study made

by L. Harting [18], giving returns of positive 80% even during a downtrend and over only 50 days, which implies a 10000% increase annually, which does not make any reasonable sense. And something similar happens with all the other few studies that also train toward ROI maximization [19] [20], making their methods questionable.

Chapter 3

Design

3.1 Problem Definition

The objective of this project is to make a model capable of beating the market in efficiency, giving a greater ROI by making use of the emotions that Twitter users express in regard to the matter. This works on the preamble that bitcoin is a volatile investment that relies on inflation and speculation, making it very sensitive to public perception.

So, the proposed structure for that is a decision model capable of deciding whether or not to invest in the currency by paying attention to how it behaved in the past and trying to induce patterns for the behavior of the price and its risk in the future.

3.2 Objectives

1. To design a Twitter data scraper capable of identifying and extracting the most relevant tweets and their daily distribution.
2. To train an Emotion Classification model specialized on tweets about cryptocurrencies and identify their effect on its price fluctuations.
3. To design and train a decision-based model capable of maximizing ROI while being trained off of Twitter and Bitcoin price historical data.

3.3 Data Extraction

The Twitter data extraction will consist of two steps: finding already scraped datasets to reduce the amount of scraping necessary and creating a data scraper to complete the

dataset with the necessary information and for future expansion, given that the official API for research is not available anymore.

A complete study of the datasets and their structure will be necessary to fulfill the needs of the project and optimize the scraping, as it is becoming one of the harder steps and, if scaled, quite expensive.

On the other hand, having a good understanding of the Bitcoin price structure and shape distribution is vital, as this will determine how the price behaves, which is needed to understand and correlate with the Twitter data extracted.

3.4 Emotion Classification

Taking into account all the studies about the different reactions to different emotions and sentiment analysis, emotion classification seems like a better indicator to predict fluctuations on Bitcoin prices.

Although there are already many working emotion analysis classifiers and datasets available online simplifying this step a lot, those are for general use and may not behave as expected for this particular case scenario, having a quite niche slang and being a topic that is usually identified as spam by most classifiers but not making sense here due to the fact that all the tweets talk about it. In addition, there are many studies that compare the result of using different training models[14], but those are quite old, so the model will be trained over transformers, in this case BERT-type model.

Because of all that, the Emotion classification will work in two steps, first identifying if a tweet was human made or spam and after that, identifying the emotions for the human written tweets.

3.5 Decision Model

Finally, for the decision model, the model used will be LSTM as it has been shown in many studies to be the best model in this type of situation, only beaten by transformers, but being those very data-hungry and not fitted for this research-based project.

The model will be decision-based, choosing between the decisions of buying, selling or holding and trained to maximize ROI against the "Buy and hold" in order to evaluate its performance on every situation.

Chapter 4

Implementation

4.1 Data Extraction

On the occasion of the recent change on ownership of Twitter, Elon Musk decided to privatize Twitter's data, making it a lot harder to get, disabling the public API for scraping, ceasing all academic research agreements and making it extremely difficult to gather data from the site.

4.1.1 Public Datasets

Due to this new policy, instead of creating a new dataset from scratch, a much better approach is to complement the already scraped data. Looking at the main public datasets available online and comparing them by monthly amount of tweets the resulting sizes are represented in **Figure 4.1**.

Among those, two stand out: Dataset 1 and Dataset 2. Dataset 1 has a really large span, going from 2009 up to late 2019, having a huge spike of tweets scraped during this last year. However, the rest has a very small number of tweets. On the other hand Dataset 2 has a smaller time span of only one and a half years from 2021 to mid 2022 but it has a really good amount of tweets every day during that period.

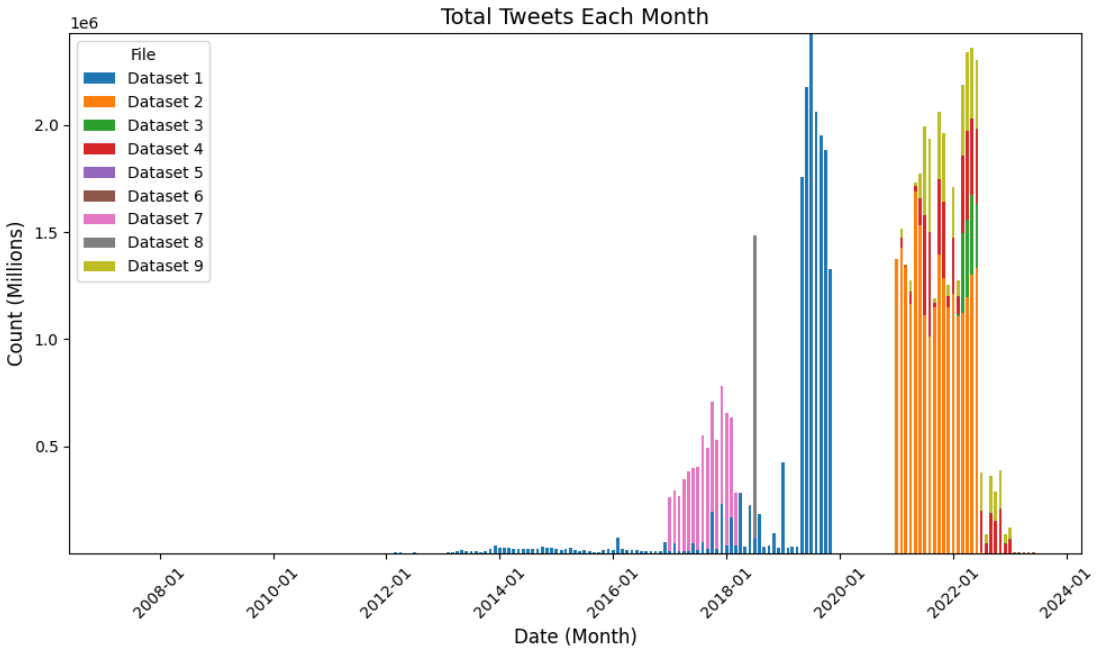


FIGURE 4.1: Histogram of Tweets per month for each public dataset

A decision factor for which datasets can be used is that, in order to get an accurate enough estimate of the Twitter's perception, a minimum amount of tweets for each step has to be reached.

For example, to provide a precision of $\pm 10\%$ with a confidence of 95% then the Cochran's formula for each emotion can be used. Each emotion has a different distribution so to get the maximum error of 10% the formula has to be used for each one of them and consider the most limiting one, so the one that requires the most Tweets/step. So in this case with 6 emotions:

Let

$$p_i \;=\; \text{real proportion of the emotion } i, \qquad r \;=\; 0.1 \quad \text{(relative margin)}$$

We wish to ensure

$$\left|\widehat{p}_i - p_i\right| \;\leq\; E_i \quad \text{with probability } 0.95, \tag{4.1}$$

where the absolute margin

$$E_i \;=\; r\, p_i \;=\; 0.1\, p_i. \tag{4.2}$$

For a 95% confidence level, the corresponding normal-approximation critical value is

$$Z \;=\; 1.96.$$

The required sample size n_i to estimate p_i within $\pm E_i$ is given by Cochran's formula:

$$n_i = \frac{Z^2\, p_i(1-p_i)}{E_i^2} \;=\; \frac{(1.96)^2\, p_i\,(1-p_i)}{(0.1\, p_i)^2}. \tag{4.3}$$

Without the real distribution of emotions the size can not be predicted exactly but an upper margin can be given by defining equiprobable emotions. So in that case:

$$E_i \;=\; r\, p_i \;=\; 0.10 \times \frac{1}{6} \;=\; 0.016667. \tag{4.4}$$

$$n_i = \frac{Z^2\, p_i(1-p_i)}{E_i^2} = \frac{(1.96)^2 \left(\frac{1}{6}\right)\left(\frac{5}{6}\right)}{(0.016667)^2} \;\approx\; 1\,921. \tag{4.5}$$

And since all emotion probabilities are identical, the required overall sample size is

$$n = 1\,921.$$

With a quick analysis, doing a daily step prediction that would be around 60.000 tweets a month, discarding all dates from Dataset 1 before 2019, making it then less than a year long.

Datasets 2, 3, 4 and 9 do have viable distributions (at least if the distribution of emotions is close to equiprobable), so Dataset 2 is the smart choice as it is simply a lot bigger than the rest combined and probably good enough for any distribution of probabilities.

Finally Dataset 7 has a decent size also but due to it being to far back from Dataset 2 and quite short it's probably not worth joining datasets (and the same for Dataset 1 during 2019).

So, the dataset that will be used for the model is Dataset 2. A problem with this is that it is quite a short span and will become tricky to train without overfitting, and the conclusions that can be extrapolated from it may not become a good representation of reality.

4.1.2 Tweet Scraping

Apart from already available datasets, scraping some data is necessary to obtain more information and make the decision model viable. As already mentioned, this practice has become increasingly more difficult, making it a very complex process and with increasing cost at scale.

There are multiple libraries already adapted for Python to make parsing the data from the site quite easy and straightforward, so the main problems with it rely on the access, the rate limitings and the bot detection.

JSON Test

Twitter, in order to prevent the user from getting access to their page through HTTP requests or headless browsers, uses a common practice which consists of sending a JSON test, which is a document that requires the user's browser to be solved. Because of that, raw requests or light headless browsers cannot perform that operation.

So in order to bypass those you are required to use a complete browser. However, in order to reduce required resources and improve the response time of the page, this test's result gets saved on the browser's cookie. Knowing this, the cookie can be retrieved and sent through a light headless browser or parametrized raw requests, making that cookie useful for at least its Time To Live (TTL), which is about a month. This also avoids the need to log into the account, as it is also saved in the cookie.

Rate Limit

Another big issue regarding tweet scraping is the extremely tight rate limits that the page offers. Those get triggered when sending multiple requests in intervals shorter than about 15 seconds each. And given that each request returns an average of 13 tweets, that makes it go to a very slow pace of less than a tweet/second.

In addition, Elon Musk added daily tweet limits to avoid scraping, **Figure 4.2**.

(a) Initial rate limit (b) Rate limit increase

FIGURE 4.2: New rate limiting constraints applied by Elon Musk

Those have slowly increased since then as their bot detection has been improving, making so that although they are not as strict anymore, they implemented other detection methods to prevent scraping.

As an example, on this project the most tweets an account has scraped in one session is 16k, followed by an account restriction being the account flagged as a bot. The obvious solution to this is obviously having multiple accounts scraping at the same time making them follow an async structure. But another problem raises from that, which is that when an account gets flagged, the IP from where it was receiving the requests gets also flagged.

Rotating Proxies

In order to prevent getting an IP banned making it possible to use more than one at the same time, is to use rotating proxies.

Those are proxy servers that act as a funnel that masks an IP sending the request from where the server is. By them being "rotating" proxies, they change every short interval making it so that you can "burn" one and use another one. For this task the type of rotating proxies needed are "sticky session" proxies which, as the name hints, do not rotate automatically letting you keep one for a whole session, in this case, for an account, and also "residential proxies", as twitter and other big websites keep track of the generic datacenter proxies (a lot cheaper).

This is necessary because otherwise Twitter detects the constant IP changes and flags the account. The only downside with rotating proxies is that they are paid services, making it increasingly costly the more tweets you scrape. The average price for the top services is around 5$/GB, which equals roughly 500k Tweets, making it quite expensive at scale.

CAPTCHA

Another concern which is a bit harder to deal with are CAPTCHA's (Completely Automated Public Turing test to tell Computers and Humans Apart).

Those come eventually when logging in or also sometimes mid-session when the site detects weird nonhuman behavior. those are extremely complicated to deal with and there are only two viable workarounds, manually solving them or using a CAPTCHA solving service, which also tend to be quite expensive and adds to the total cost of the operation.

For this project, as it ended up being a relatively small operation and just a sample project the CAPTCHA's were dealt with manually.

Search Query

Once the site bypassing is dealt with, what is left are the search methods used to find the desired data.

For what this project is about, the necessary query instructions used are the advanced search method that Twitter brings. Those let the user search for tweets containing a certain word or words. But not only that, it also lets you use certain operators like OR, NOT to chose which words to see or other useful operands like dates in which to make the search, language, etc.

Also it lets the user chose if they wants to see the tweets sorted by relevance or "Latest-first". As an example, the query:

```
(bitcoin OR btc) -ethereum lang:en until:2024-12-06 since:2024-12-05
```

returns all the english tweets containing either Bitcoin or BTC but not containing Ethereum posted between the set dates.

4.1.3 Daily Tweet Count

A very important metric that can be very helpful as an indicator is how many tweets regarding bitcoin are getting published in a certain day. There is no straight forward way to know that as twitter doesn't provide it so the number will have to be an approximation.

Scraping the whole day is not viable as there is a huge amount of tweets so a way to count them could be just sorting the tweets by latest first, and scraping a certain ammount and make an approximation to resize it as the rest of the day but there are many problems with this.

The first idea that comes through is to find the daily tweet distribution, scrape a a small set of it on "Latest first" mode and scale those by the relative amount.

So for example, given that **Figure 4.3** shows that during the first hour of the day a 3% of the tweets are posted, this gives that just by scraping the first hour of the day we could have an approximation of its size.

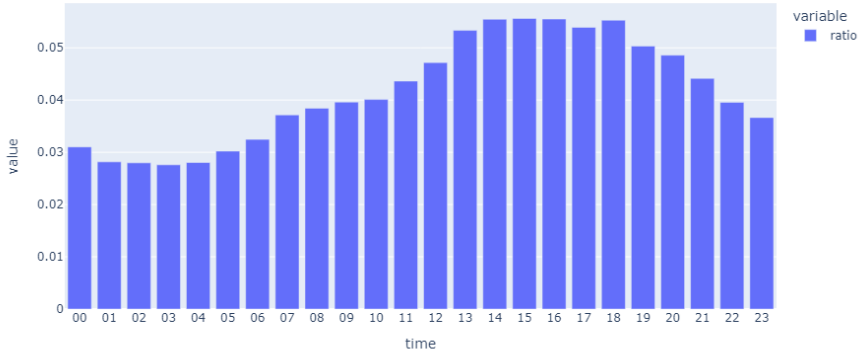


FIGURE 4.3: Daily Tweets distribution extracted from the public datasets

However, if something happened in the middle of the day triggering a huge amount of tweets it would not detect it and also, the tweets gathered would have a huge time zone bias, mainly considering the ones that have the peak hour during GMT+0.

So in order to gather better information, an evenly distributed subset should be extracted. A way to do this is using the properties of natural language.

Query Splitting

A way to split the amount of tweets into a subset is making use of the patterns in natural language. So, for example, if the word "the" appears 2% of the time when speaking in the English language, that could be used to search a query in Twitter containing "bitcoin" and "the", for example, and that would make the query already 50 times smaller. It is important to choose words wisely to avoid biased words like "profit" to affect the results, as those appear much more often on rising periods, messing up the dimentioning.

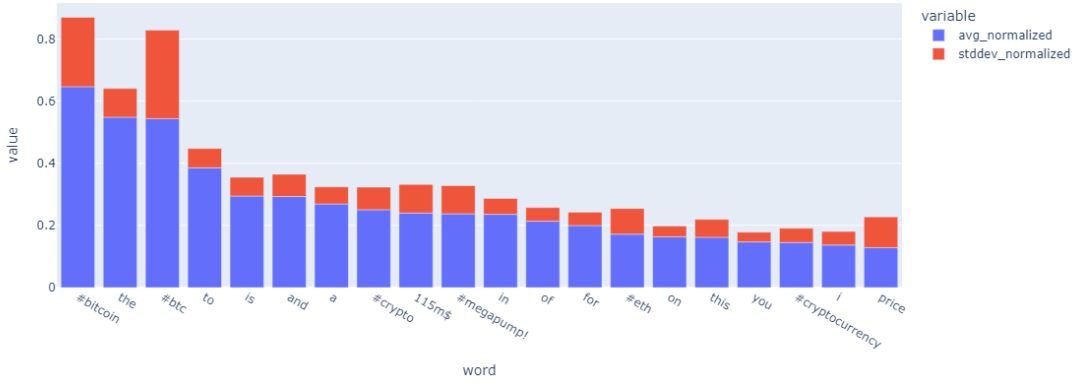


FIGURE 4.4: Histogram of probability of words from the tweets from the public datasets

The previously scraped datasets can be used to get the probabilities of words, so by tokenizing the tweets into words and checking the percentage of use that they have daily (**Figure 4.4**) the word probabilities can be approximized.

But some of the words have very noticeable biases (given by their huge standard deviations) so a more accurate attempt would be to consider only english words.

An other advantadge that gives us using only english words is that we can make use of the Zipf's law.

4.1.3.1 Zipf's Law

Zipf's law is a pattern that natural pattern sorting tends to for efficiency that can be used for this purpose.

> **Zipf's law** is an empirical law stating that when a list of measured values is sorted in decreasing order, the value of the n-th entry is often approximately inversely proportional to n.
>
> The best known instance of Zipf's law applies to the frequency table of words in a text or corpus of **natural language**.
>
> $$word\ frequency \propto \frac{1}{word\ rank}$$
>
> *(Source: Wikipedia)*

So, as it can be seen in **Figure 4.5** by only considering words in the English dictionary the distribution appears to follow a inverse proportion as predicted.

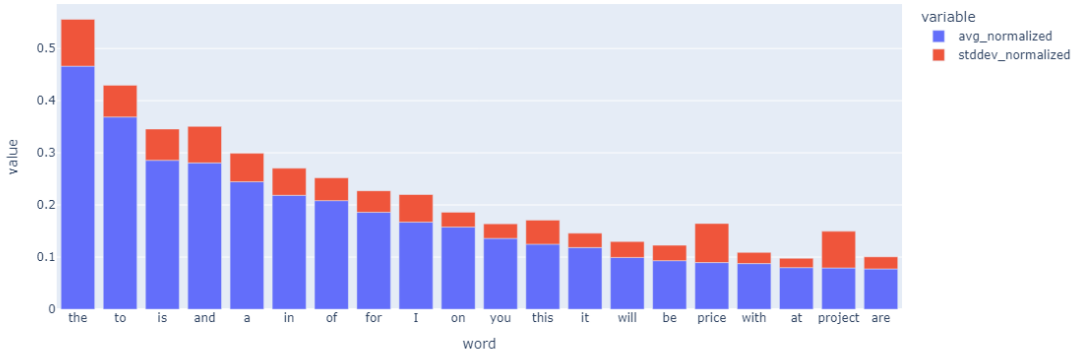


FIGURE 4.5: Histogram of probability of only english words from the public datasets

Some words as "price" or "project" are still noticeable because of their bias, making their standard deviation throughout the days be uneven.

However, the probabilities here aren't really following the probability of the word appearing in general, but the probability of it being in a tweet, but we can still consider it to follow Zipf's law as Zipf's law tells us that a word's overall "base" probability (its chance of occurring anywhere) scales roughly as

$$p \sim \frac{C}{r^s}\,, \tag{4.6}$$

So for a tweet of length L, the chance of it containing that word would be:

$$P_{\mathrm{L}} = 1 - (1-p)^L\,. \tag{4.7}$$

And a common approximation for a small p is to consider:

$$P_{\mathrm{L}} = 1 - (1-p)^L \approx Lp\,. \tag{4.8}$$

And, since the aim of this is to get a really small set, a low probability word or set of words are going to be used, so it can indeed be considered really small. So it should also follow an inverse proportion to the rank.

But in this case, there are tweets of varying length, so in order to prove that it also follows that chance for that distribution it is required to know its length probability density distribution,

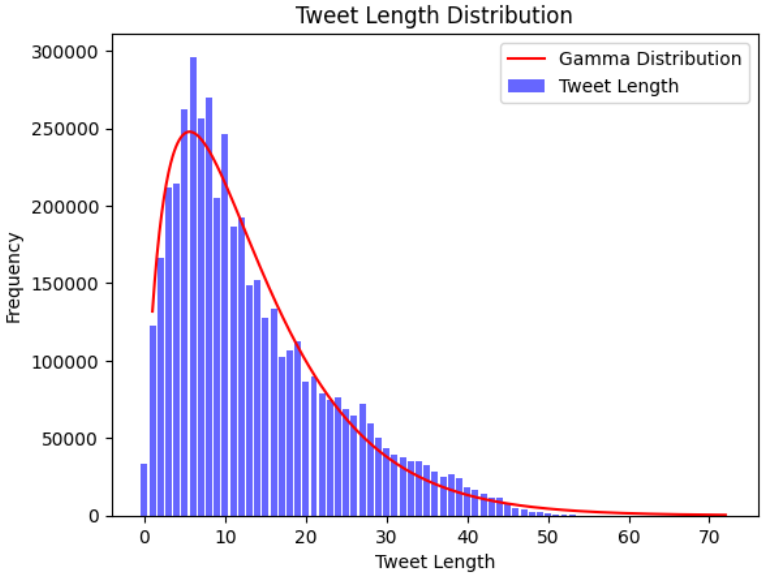


FIGURE 4.6: Histogram of the tweet length distribution from the public datasets

As it can bee seen in **Figure 4.6** it resembles a lot to a Gamma distribution, so if a Gamma distribution with shape α and scale β is considered, then the overall probability that the word appears at least once is obtained by averaging over all possible sentence lengths:

$$P = \int_0^\infty \left[1 - (1-p)^L\right] f(L)\, dL\,. \tag{4.9}$$

Which, for small p

$$(1-p)^L \approx e^{-pL}\,, \tag{4.10}$$

so using that approximation allows the use of a Laplace transform

$$P \approx 1 - \int_0^\infty e^{-pL}\, f(L)\, dL = 1 - \mathbb{E}[e^{-pL}]\,. \tag{4.11}$$

And if L is Gamma-distributed, its Laplace transform (i.e., $\mathbb{E}[e^{-pL}]$) is given by

$$\mathbb{E}[e^{-pL}] = (1+\beta\, p)^{-\alpha}\,. \tag{4.12}$$

Thus, giving

$$P \approx 1 - (1+\beta\, p)^{-\alpha}\,. \tag{4.13}$$

And again, for small p it can be rewritten as:

$$(1+\beta p)^{-\alpha} \approx 1 - \alpha\beta p\,, \tag{4.14}$$

so that

$$P \approx \alpha\beta\, p\,. \tag{4.15}$$

Giving the ratio between the probability of using the word in a tweet proportional to $\alpha\beta$ times the probability of the word, being α and β the shape and scale of the gamma distribution.

So now, having the word probabilities, allows to split the dataset but in order to know how many tweets that will be, some dimensioning is necessary. Setting a twitter account to attempt to scrape all the tweets in a day it got banned after scraping 13k tweets and it was roughly 40 minutes of the entire day.

So looking back at **Figure 4.3** a quick calculation can be made to estimate roughly the amount of tweets that were posted that day.

Knowing that the first hour is about 3% of the total:

$$N \approx N_{40\ min} * \frac{T_{00-01}}{40\ min} * f(P_{00-01}) = 13000 * \frac{60\ min}{40\ min} * 0.031 \approx 629032, \tag{4.16}$$

So it can estimated that that day around 629032 tweets were published in total. And so, in order to decide how many tweets to scrape, the same thing can be done the other way around so, in order to scrape 1000 tweets,

$$P \approx \frac{1000}{629032} = 0.0015897 \tag{4.17}$$

a query representing 0.0015897 of the dataset will be needed.

Although this approximation isn't completely precise either as when using it for scraping just looking at the first tweets each day, this scales every day by the same number, making the error rate evenly distributed and non biased. Also, given that this will anyways be scaled to be fed to the decision model, having a wrong scale factor doesn't really matter and it's only a useful tool to predict how many tweets will be scraped when using a particular query and thus size the project accordingly.

4.1.3.2 Daily Tweet Scraping

With all those calculations done, the daily scraping to predict the amount of tweets published each day can begin. Due to the already stated difficulties to scrape, the amount of tweets that can be scraped is reduced as this is just a low-scale project, and the funding and time are limited. Because of that, instead of scraping every day, only one in every three days is scraped for now, and the values in between will get average-filled.

So, setting a target of 500 tweets to scrape, the query necessary needs to have:

$$P \approx \frac{500}{629032} = 0.0007949 \tag{4.18}$$

And that can be done by using a query that combines a set of words that has a similar joined probability. The amount scraped at a time does not really matter as it can be extended in the future doing binary aggregations. So for example, if the query used is:

```
To at with bitcoin lang:en until:2021-12-06 since:2021-12-05
```

This should return about 500 tweets spread throughout that day. In order to expand that approximation to make it more accurate, using another combination would not solve it as both sets could intersect, so in order to get another set free of intersections with this one so that: $\forall x \, (x \in A \implies x \notin B)$, the easiest way is to negate one of the variables from the previous combination, so in this case:

$$\{\text{to, at, with}\} \cap \{\text{to, at, } \neg\text{with}\} = \{\text{to, at}\} \tag{4.19}$$

And repeating the same process until the whole set is covered. But a problem with this occurs, by doing ¬with the probability gets inverted, so a probability that used to be about 9% transforms into a 91%, making it 10 times bigger. So the easy fix to that is to add intermediate steps to fill it slower, for example 'you':

```
To at -with you bitcoin lang:en until:2021-12-06 since:2021-12-05
```

And that allows to get the average and have a much smoother regression (**Figure 4.7**).

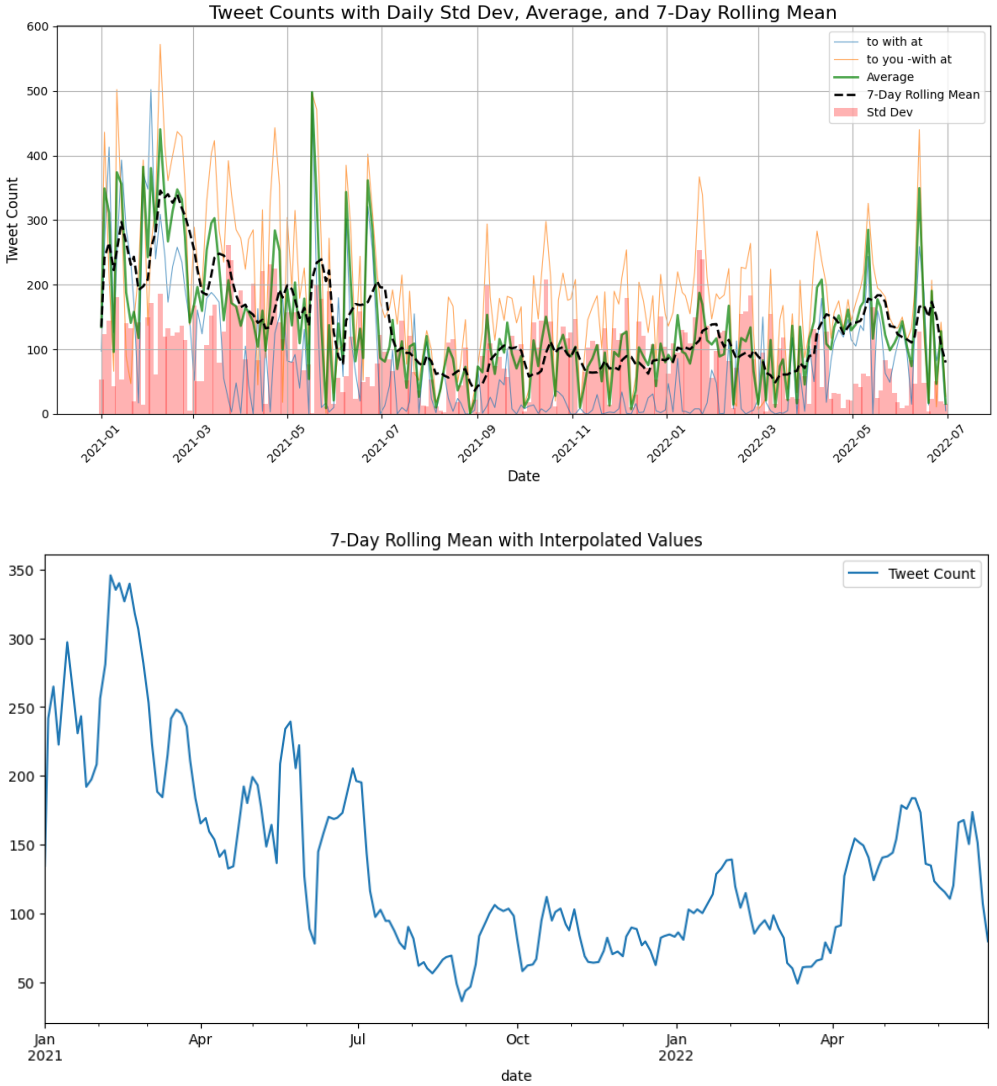


FIGURE 4.7: Prediction of daily tweet counts (unscaled)

Also, in order get a smoother trend line, given that the scraped amounts are quite low giving space for errors, a rolling window can be used. The result is not scaled to the real amount but that is not even necessary as it will later be scaled anyway, so the only thing that matters is the shape.

4.2 Tweet Pre-Processing, Labeling and Classification

Once the tweets are extracted, the next step is to retrieve the relevant data from them. Given that the final goal of this is to get an emotion classification, a good idea would be to see what the public datasets provide.

Some of them contain the user's name, the amount of likes, retweets, comments that the post has, its ID, etc. But two thing that all of them have is the Tweet's text and date.

In particular, Dataset 2 from **Figure 4.1**, which is the main dataset for this project, only has date and text. It would have been nice to also have the tweet's likes or any means to size the impact of a tweet but because of that, each tweet will have to be taken into account as equal.

In order to understand what emotion is perceived from each tweet, the models that usually gives the best results are transformers, in particular Long Language Models (LLM) like Bidirectional Encoder Representations from Transformers (BERT).

4.2.1 BERTweet

> **BERT** is a language model introduced in October 2018 by researchers at Google.
>
> It is trained by masked token prediction and next sentence prediction. As a result of this training process, BERT learns contextual, latent representations of tokens in their context, similar to ELMo and GPT-2.
>
> It was originally implemented in the English language at two model sizes, BERTBASE (110 million parameters) and BERTLARGE (340 million parameters). Both were trained on the Toronto BookCorpus (800M words) and English Wikipedia (2,500M words).
>
> *(Source: Wikipedia)*

The BERT structure is the current state of the art in open source pre-trained language models and has had multiple models derived from it either trained on larger datasets or using more or less tokens.

It works as a base for training other models as it has a great understanding of natural language and capable of understanding the connections between tokens.

One of the most famous ones, and one of the most advanced is the more recent Robustly Optimized BERT Pretraining Approach (RoBERTa), which builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training

with much larger mini-batches and learning rates, which makes it more accurate and efficient.

But considering that tweets are quite a singular structure of text, with capped length, keywords like hashtags and mentions and specific slang, understanding it reading from mainly well structured text makes it not a great option.

But fortunately there are models like BERTweet [21] which have been built over RoBERTa trained with tweets. In the particular the corpus used to pre-train BERTweet consists of 850M English Tweets (16B word tokens 80GB) and it has proved to perform better than multiple other models for sentiment analysis and emotion classification as seen in the tests from **Figure 4.8**:

	Model	F_1^{pos}	Accuracy
Our results	RoBERTa-large	73.2	76.5
	XLM-R-large	70.8	74.2
	BERTweet-large	**76.3**	**80.3**
	RoBERTa-base	71.0	74.0
	XLM-R-base	66.6	70.8
	BERTweet-base	**74.6**	**78.2**
Wu et al. (2018)		70.5	73.5
Baziotis et al. (2018)		67.2	73.2

Performance scores on the SemEval2018-Task3A irony detection test set.

(a) Text Classification Test (Irony detection)

	Model	AvgRec	F_1^{NP}	Accuracy
Our results	RoBERTa-large	72.5	72.0	70.7
	XLM-R-large	71.7	71.1	70.7
	BERTweet-large	**73.3**	**73.1**	**72.2**
	RoBERTa-base	71.6	71.2	71.6
	XLM-R-base	70.3	69.4	69.3
	BERTweet-base	**73.2**	**72.8**	**71.7**
Cliche (2017)		68.1	68.5	65.8
Baziotis et al. (2017)		68.1	67.7	65.1

Performance scores on the SemEval2017-Task4A sentiment analysis test set.

(b) Sentiment Analysis Test

FIGURE 4.8: Results of BERTweet model, outperforming other models in tweet tests

There are already multiple models that use BERT for emotion or spam detection (to avoid having noise on the data) but, although they all claim to have quite high accuracy rates, they perform extremely poorly on the Bitcoin tweets dataset.

That is mainly because, for the general use, when a tweet refers to Bitcoin or any other cryptocurrency, usually tends to be spam or phishing scams, and on the other hand, the fact of talking about currencies or investments gets detected as greed. But that changes when the whole dataset refers to Bitcoin. Because of that, that bias should be removed so that the model focuses on the rest of the traits.

Also, emotion classification makes sense if it is aimed at relevant emotions from an investment perspective, so making new custom models is the smart move here.

To do so, the hardest part is to get a labeled dataset with the data. BERT models have a really robust structure so they don not need to have a very extensive fine-tuning dataset to create classification models.

So in order to train the model, a dataset with 100k labeled tweets will be used. The problem now is how to viably label that many tweets.

Although it may look redundant, the cutting edge LLM's are extremely good at classifying text and give very precise and customizable results. The problem with them is that they are extremely slow due to their huge amount of tokens and also you need to pay for their API services at scale. Fortunately, classifying 100k tweets is time and economically viable, and Deepseek has one of the cheapest and best API's at the moment, and with a few tweaks can make the classification extremely cheap, so that's the go-to in this case.

4.2.2 Pre-processing Tweets

First of all, in order to get an unbiased subset, 200k tweets get extracted from the whole Dataset pool in **Figure 4.1**. That dataset is likely going to have useless data or in other languages, so some pre-processing is needed.

This pre-processing will have to be applied to the entire chosen dataset (which is 22M tweets long), so it has to be fast and close to O(n).

FastText

One of the main concerns is language, as BERTweet is only trained over an english corpus of tweets. To achieve that, fastText can be used.

FastText is a library for learning of word embeddings and text classification created by Facebook's AI Research (FAIR) lab known for its lightning fast speed. One of its resource models is Language Identification, which can be used for this purpose.

Surprisingly, it is capable of detecting it for the whole 200k tweets dataset and filtering them out in under 3 seconds, removing around 13% of the tweets.

Minimum Sentence Length

After removing the non-English words, only the English tweets are left. But some of those still lack semantic meaning. A good way to remove those is to count how many "valid" words are on each tweet and filter them out if there are less than 3. This removes an additional 22% of the tweets.

MinHash Duplicates removal

On the full dataset having duplicate tweets does not necessarily mean something bad, as some tweets make sense to repeat briefly. However, for the model training, those would not add much information to it. So for that reason, a duplicate removal is applied.

A good approach to do that is to hash the tweets and then comparing them to each other with a certain threshold. But a problem with those is that some tweets (mainly bot written ones) that write the exact same thing but changing dates, prices, hashtags or user tags. That makes it so that some of them don't get caught by the MinHash comparator so some normalization of the data is needed.

A really good library for that is Spacy, which can detect what each token (in this case words) represents, so this way the dates, numbers, etc. can be parsed as so, making the MinHash's job easier.

On the other hand, for the tweet related items such as hashtags, links or user tags the BERTweet library itself is already trained over normalized text.

Tweet Normalization

In order to train BERTweet wisely, the developers decided to normalize the text replacing all the user tags, URL's and emojis by a place-hanger. This way the model can understand that there is a user to whom it is refering or an URL but without the need for the model to induce the internal connection that all texts that start with http are URL's and that all names that start with @ are usernames.

So in the end the text gets converted into something this:

```
DHEC confirms HTTPURL via @USER :crying_face:
```

They provide this normalization function within a library, making the fine-tuning easier and also giving the MinHash an easier hash comparison. So using this with the MinHash and all the previous filters, the dataset gets reduced to 109k.

4.2.3 Labeling Dataset

With that smaller but better dataset it can proceed to getting labeled. To do so a this prompt followed by 100 tweets (in order to reduce the amount of requests needed) classifies the whole dataset in three cathegories:

```
system_prompt = """
You are a classifier for bitcoin related tweets. For each tweet below:
1. Classify it as "spam", "bot" or "human".
   - Classify as "spam" the spam, phishing, promotion, "join this" or "
    play this" or redirecting tweets
   - Classify as "bot" for automated price updates, news bots, or
    repetitive patterns
   - Classify as "human" if the tweet seems like a tweet written by a
    human
2. Assign one emotion representing how a bitcoin investor or fan would
    feel reading it from [Fear, Greed, Neutral, Optimism, Pessimism, Anger
    , Excitement].
3. Provide sentiment score (-1 to 1) regarding how a bitcoin investor or
    fan would feel about this tweet.

Return JSON in format for each ID:
{
    "1":["human","Anger",0.8]
    "2":["bot","Greed",-0.6]
    ...
}
"""
```

By having this start of the prompt sent always, it can be saved into cahce, which is a method that Deepseek uses to reduce processing time and cost, making it cheaper. Also, asking for a compact and simple output, the output tokens needed get reduced, which are the most expensive being about 5-10x the cost of input tokens for cache hit and miss respectively.

Each response takes about 1 minute, so in order to label it in a reasonable time, multiple async requests at the same time are required.

Also, although it may sound surreal, another pre-processing step necessary is to modify all the tweets to avoid words like 'Taiwan', 'Xi Jinping' or other sensible Chinese topics as Deepseek would return a default response stating that that topic is over its scope, instead of the requested format messing with the response parsing (learned the hard way).

And the result is the following, which had a total cost of 2$ worth of Deepseek API tokens for the 100k tweets **Table 4.1**.

Text	Spam	Emotion	Sentiment
Want to put your #bitcoin company at the end of every #game played? Sponsor a day for...	spam	Greed	-0.3
This young Kenyan chess champion wants to take her moves abroad. @newstwiteafrica...	bot	Neutral	0.0
#Bitcoin is the best money that humanity has ever had.	human	Optimism	0.8
@Lukewearechange Can you guys talk about the woke left mob attacking Bitcoin on the next...	human	Anger	0.6
Man Sells Everything He Owns To Buy Bitcoin! https://goo.gl/fb/zw5LG9	human	Excitement	0.5
The man who photobombed Janet Yellen with a Buy #bitcoin sign has received nearly $16000...	human	Excitement	0.6
When you buy #Bitcoin with dirty fiat https://t.co/UBpsvCcntL	human	Neutral	0.1
@Leoalenkar_1 @kinas98 Hi, have you been exposed to the trading investment on BINARY...	spam	Greed	-0.5

TABLE 4.1: Classifications made by Deepseek on Spam/Human, Emotions and Sentiment Analysis

The sentiment is not required, but given that the cost of including it to the prompt and the solution is extremely low, being less than 5 cents, it was included in case the emotion classification did not work as expected.

4.2.4 Model Fine-tuning

Spam/Human Classifier

Instead of making only a distinction between spam and human, the classification also creates a "bot" category as a lot of accounts are bots that follow events or price variations, so they behave slightly different from spam, as it can give negative news and is usually informative, while spam is usually non-related to crypto and just use it as a currency for gambling, NFTs or scams.

This difference can be seen if the emotions distributions are taken separately as in **Figure 4.9**.

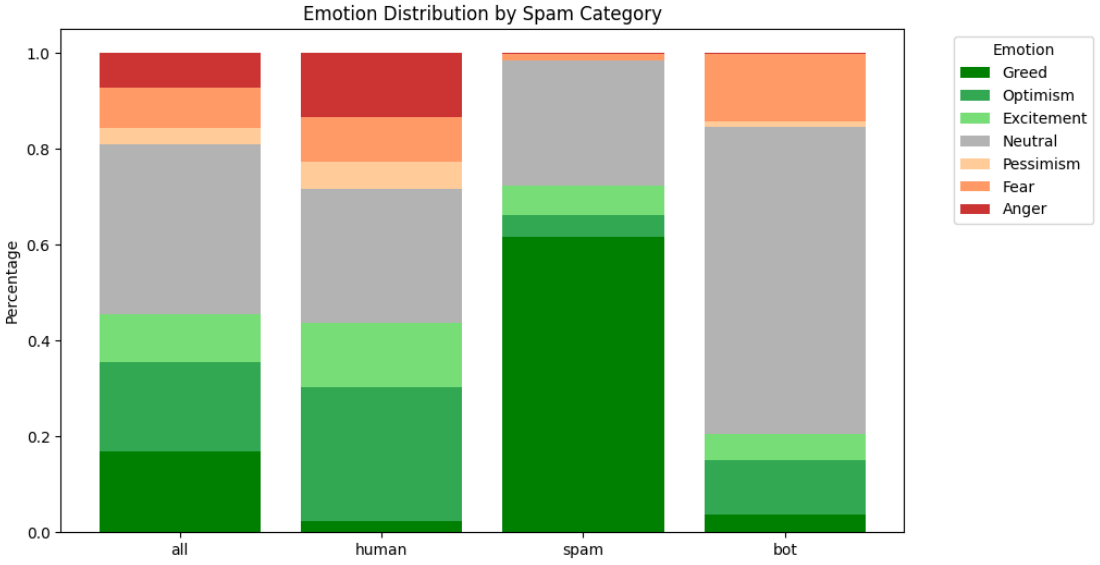


FIGURE 4.9: Emotion distribution for each different Spam/Bot/Human category

And shows the importance of separating them into categories. As explained earlier, the Bot category has an almost even distribution between positive and negative as they usually reflect the reality of the currency, whereas on the Spam category they are almost always positive and greedy. Anyway, as with the sentiment category, they got separated during the training as the circumstances allowed it due to the great performance of the LLM used (Deepseek) but, if required, they can be joined together, even allowing for a regression type model.

But for this case, the go to was a classification model, which in order to avoid misunderstandings penalizes more the cross classifications between human and spam or bot than between spam or bot following a cost matrix **Table 4.2a**.

(a) Misclassification cost matrix

	Human	Bot	Spam
Human	0.0	2.0	2.0
Bot	2.0	0.0	0.5
Spam	2.0	0.5	0.0

(b) Class distribution

Class	Percentage (%)
Human	52.986082
Spam	24.284490
Bot	22.729429

TABLE 4.2: Misclassification costs (a) and label class distribution (b).

Also, the fact that the dataset is not evenly distributed and has more human written tweets than bots or spam, the classification would have a bias so, in order to solve that, either the dataset has to be truncated or the loss function balanced so that the model

does not tilt the distribution to a certain class. And although the first one is much easier to implement, it loses information so it is much better to use the second one, scaling the results for the inverse proportion of the distribution, in this case **Table 4.2b**.

These modifications allow for a much cleaner classification and end up giving an accuracy of 88% (**Figure 4.10**).

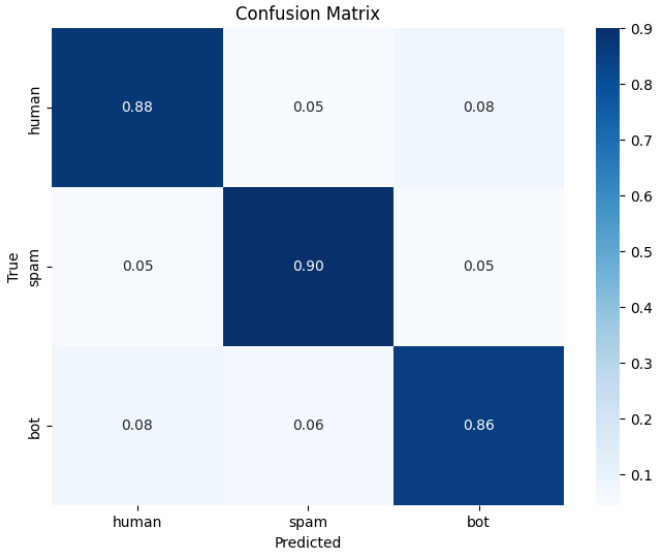


FIGURE 4.10: Confusion matrix of the spam/bot/human classification

Emotion Classifier

Similarly to the spam classifier, the emotion classifier also uses a confusion matrix and a class scaler to improve the prediction shapes using the following matrix **Table 4.3** and scaling by the inverse of the distribution **Table 4.4**.

True\Pred	Greed	Optimism	Excitement	Neutral	Pessimism	Fear	Anger
Greed	0.0	0.5	0.5	1.0	4.0	4.0	4.0
Optimism	0.5	0.0	0.5	1.0	6.0	4.0	4.0
Excitement	0.5	0.5	0.0	1.0	4.0	4.0	4.0
Neutral	2.0	2.0	2.0	0.0	2.0	2.0	2.0
Pessimism	4.0	6.0	4.0	1.0	0.0	0.5	1.0
Fear	4.0	4.0	4.0	1.0	0.5	0.0	2.0
Anger	4.0	4.0	4.0	1.0	1.0	2.0	0.0

TABLE 4.3: Misclassification cost matrix between the emotion classes.

A cool thing is that now, with the true distributions acquired from the dataset, a more accurate prediction of the minimum tweets per step required can be achieved.

Emotion	**Percentage (%)**
Neutral	35.630507
Optimism	18.527572
Greed	16.955181
Excitement	9.952391
Fear	8.503128
Anger	7.130479
Pessimism	3.278853

TABLE 4.4: Distribution of emotions in the dataset.

So applying Cochran's formula as in **Equation 4.3** again with the lowest percentage distribution (Pessimism):

$$E_i \;=\; r\,p_i \;=\; 0.10 \times 0.03278853 \;=\; 0.003278853. \tag{4.20}$$

$$n_i = \frac{Z^2\,p_i(1-p_i)}{E_i^2} = \frac{(1.96)^2\,\big(0.03278853\big)\big(1-0.03278853\big)}{(0.003278853)^2} \;\approx\; 11\,332. \tag{4.21}$$

At least 11k tweets/step are required for a solid representation. So, the only one that gets remotely close to that is the Dataset 2 from **Figure 4.1** with around 250k tweets/week (**Figure 4.11**).

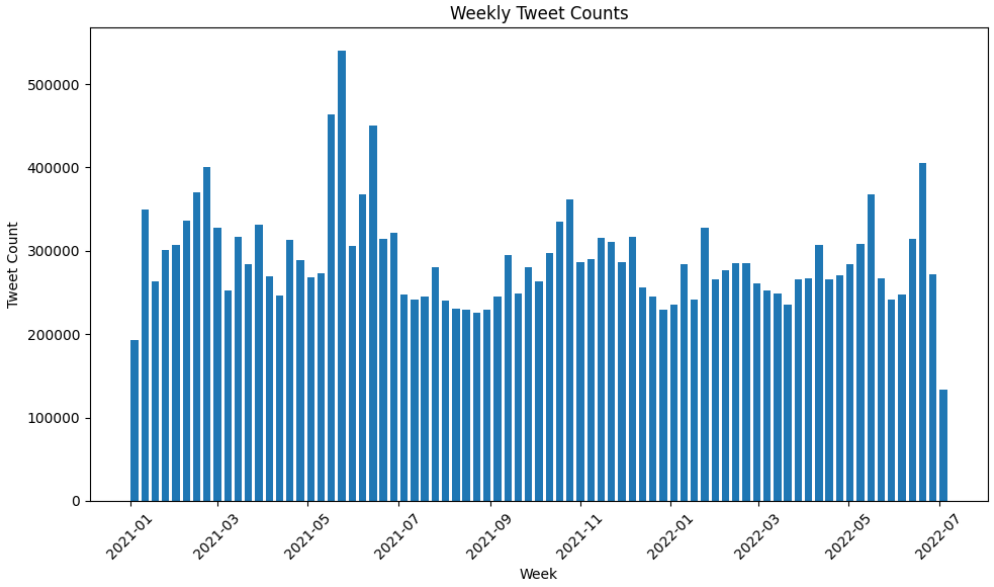


FIGURE 4.11: Weekly distribution of the tweets on Dataset 2

But going back to the classification model, this time the accuracy isn't as great mainly due to the fact that it needs to classify between 7 classes this time, increasing the entropy of the results, giving up to a 65% accuracy (**Figure 4.12**).

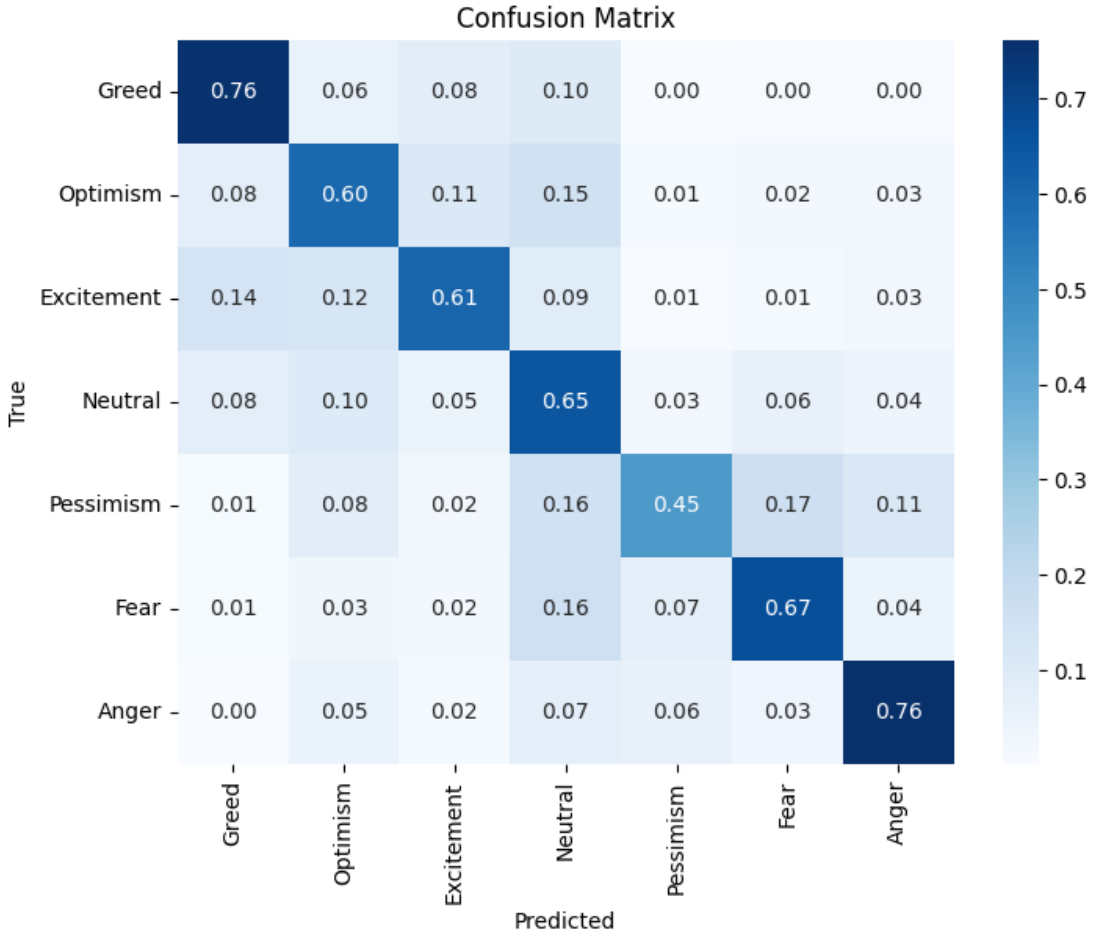


FIGURE 4.12: Confusion matrix of the emotion classification

Here, the cost matrix is a lot more clear, so although the accuracy is lower, the confusions almost always occur between similar emotions and almost never between opposite ones.

Both models and the labeled dataset are now publicly accessible on huggingface, and as of May 2025 the spam detection dataset has 100+ monthly downloads.

Category	**Link**
Labeled Dataset	`sandiumenge/bitcoin-tweets-spam-emotion-sentiment`
Spam Classifier Model	`sandiumenge/twitter-bitcoin-spam-detection`
Emotion Model	`sandiumenge/twitter-bitcoin-emotion-classification`

TABLE 4.5: Datasets and models on huggingface with their paths.

4.3 Bitcoin Historical Data

Now, with the tweet datasets already labeled, the only thing missing is to pre-process the Bitcoin historical price data. For it, for maximum precision a 1 minute precision dataset comparing BTC to USDT (which is a token that resembles the USD valuation).

The first and most important thing to do is to change the price for its percentage gain regarding the last step (in this case, last minute). What this does is making it an easier format to understand when dealing with ROI maximization. Because of this, what will matter for the model is not the actual price gain but its increase. This helps as, considering that what is being done is a decision model between holding and selling periods, the way to calculate the gain is to multiply all the steps where the decision returned hold. Otherwise, it would have to do $\frac{\text{HoldEnd}-\text{HoldStart}}{\text{PriceAtHoldStart}}$ for each buy/sell period. Having that simplified does not only help the loss calculation, but it also makes it easier to understand for the model as having a step-by-step gain it is fed a lot easier by the model as it understands the data this way.

Having this data in minute precision is not really helpful, given that the steps that are going to be used are a lot bigger than that, but what this allows is to know the distribution of the data and how it changed during the hourly or daily period. For example, a really good metric to have apart from the price itself is the deviation of the price the last hours, as it gives a rough estimate of how volatile the price is becoming. This can be very useful, as it gives an idea of magnitude, which together with the emotion bias and the price trend can give an idea of how much the price will change and how risky it can get.

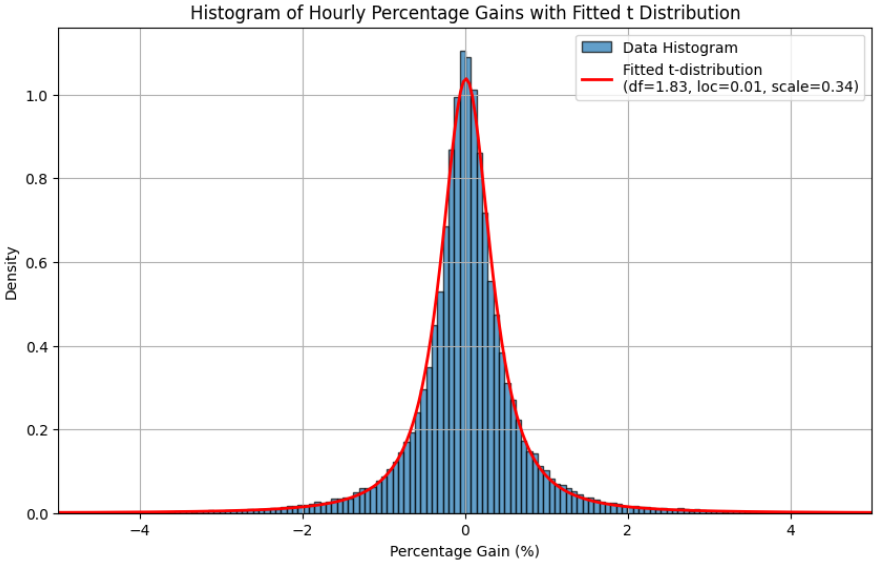


FIGURE 4.13: Distribution of price percentage gain with minute precision of bitcoin prices with a fitted T-Student distribution

When looking at its distribution (**Figure 4.13**), it follows almost perfectly a T-Student, which makes sense as that is also the shape that most investments follow.

But that presents a problem as the deviation for a T-Student is only defined for degrees of freedom $v > 2$ as when it is smaller than that it diverges, so another approximation must be done. A finite approximation is to clip the tails and define a deviation from that. A good way to do that is to use the Interquartile range (IQR) which, as the name suggests, describes the values between 25% (Q1) and 75% (Q3) of the distribution. This gives an idea of the dispersion of the data and does not have problems with $v < 2$.

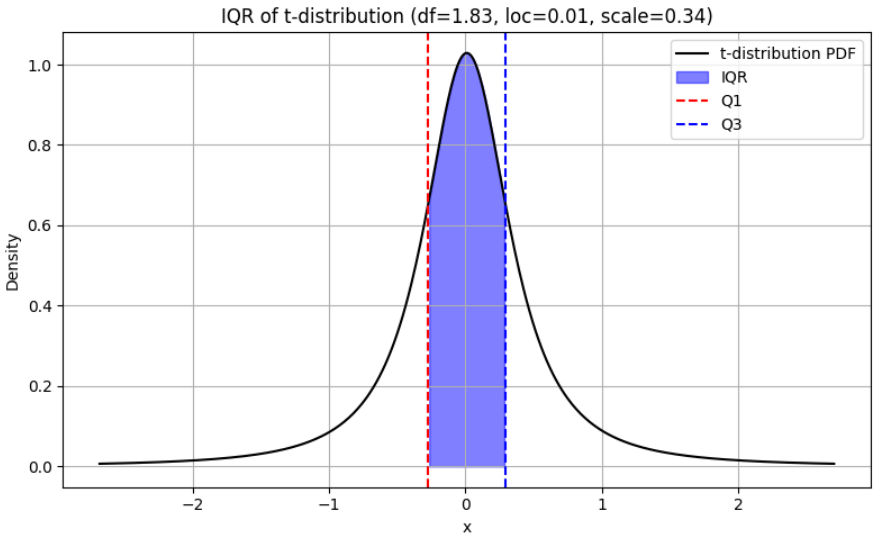
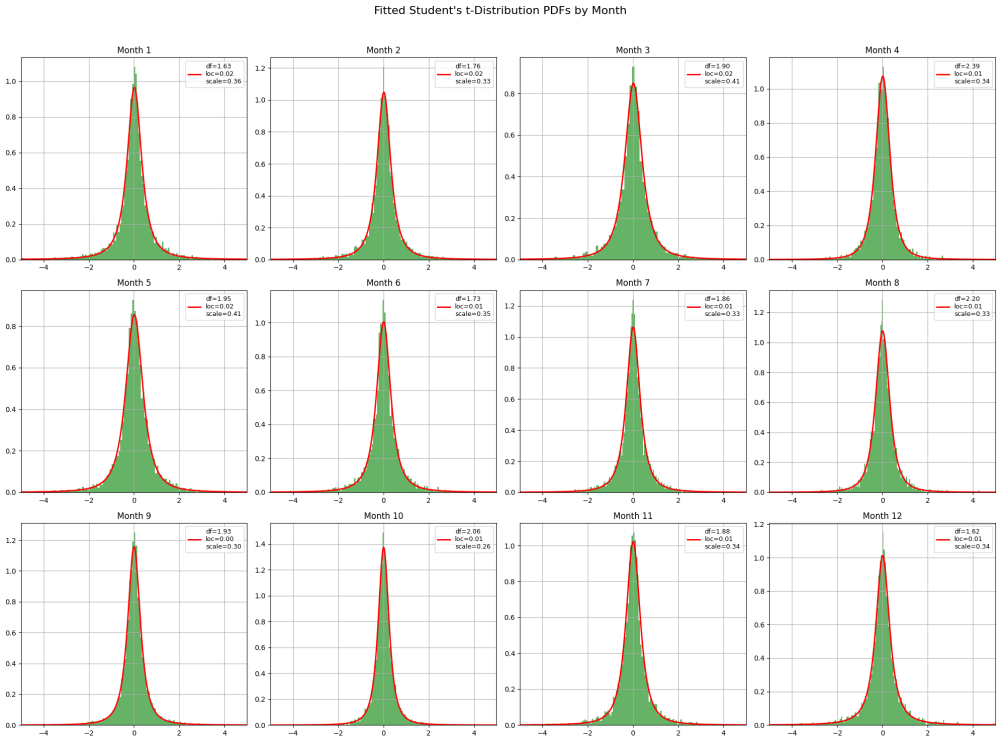


FIGURE 4.14: IQR of a T-Student distribution

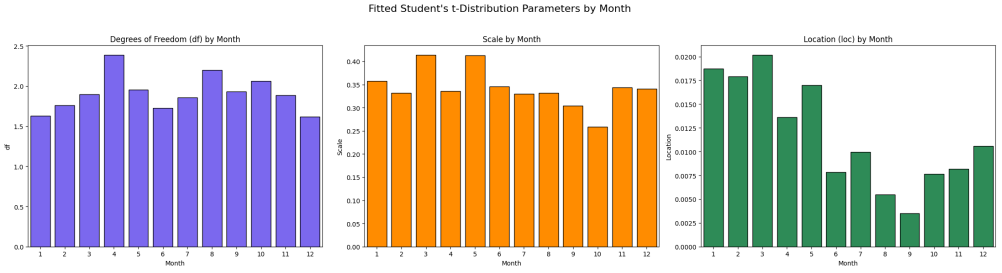
Data patterns

Now, although none of these will be used as parameters, they can give an idea of the patterns that the distributions follow, so let's look at the distributions throughout the cycles.

To begin with, when looking at the prices gain on a monthly scale (**Figure 4.15**) there's a visible drop on the gain during the summer, probably due to people spending more money on vacations and not paying as much attention on investments, which is also supported by the lower scale and higher degrees of freedom, which indicate a lower deviation, so less action an less aggressive.



(a) Distribution Shape



(b) Distribution parameters

FIGURE 4.15: Distribution of price percentage gain throughout the months

On the other hand, although when looking at the distribution it is possible to expect higher price during the start of the month dates (due to the paychecks coming then), there does not seem to bee a pattern as clear as with the months and looks a lot more erratic (**Figure 4.16**).

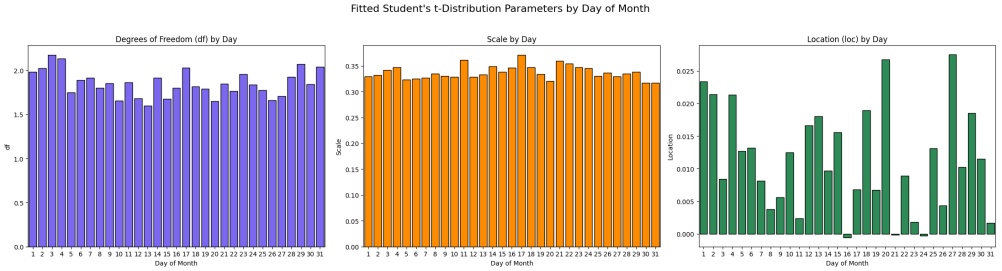


FIGURE 4.16: Distribution of price percentage gain throughout the day of the month

Finally, although with only 8 years it is not very representative, the yearly distribution is the following (**Figure 4.17**):

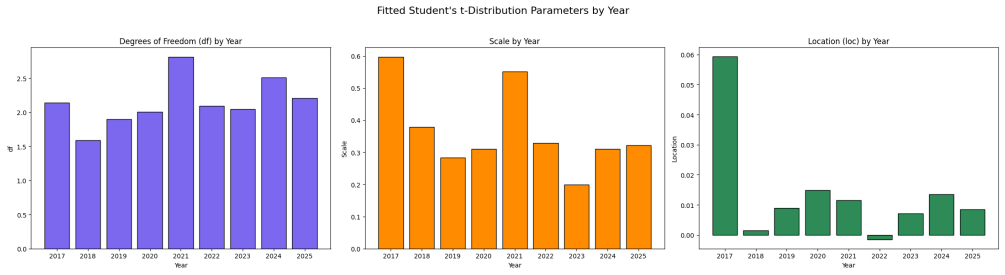


FIGURE 4.17: Distribution of price percentage gain throughout the years

4.4 Decision Model

With all the parameters already isolated, the next step is training the decision model. The parameters that will be used for it are the following:

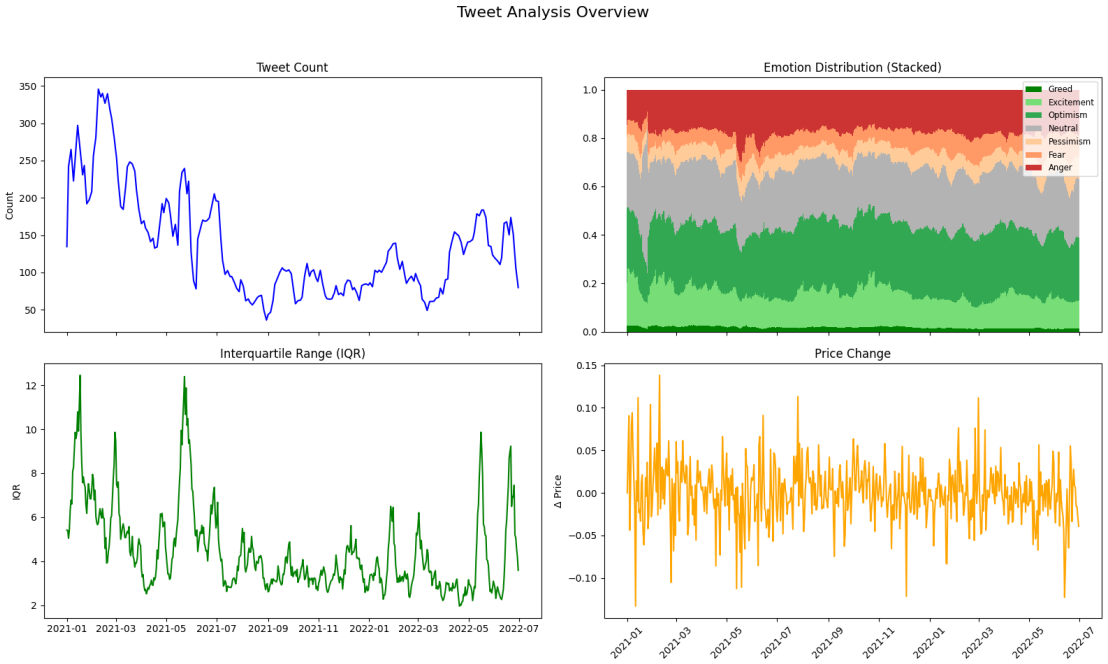


FIGURE 4.18: Parameters fed to the model

Where the emotions are used as independent variables. A cool visualization is to plot it with the price chart, making the inherent relationship between them clear, as seen in **Figure 4.19**, where there is a visible correlation between the increase of positive emotions when the value is rising and the increase of negative emotions when it decreases:

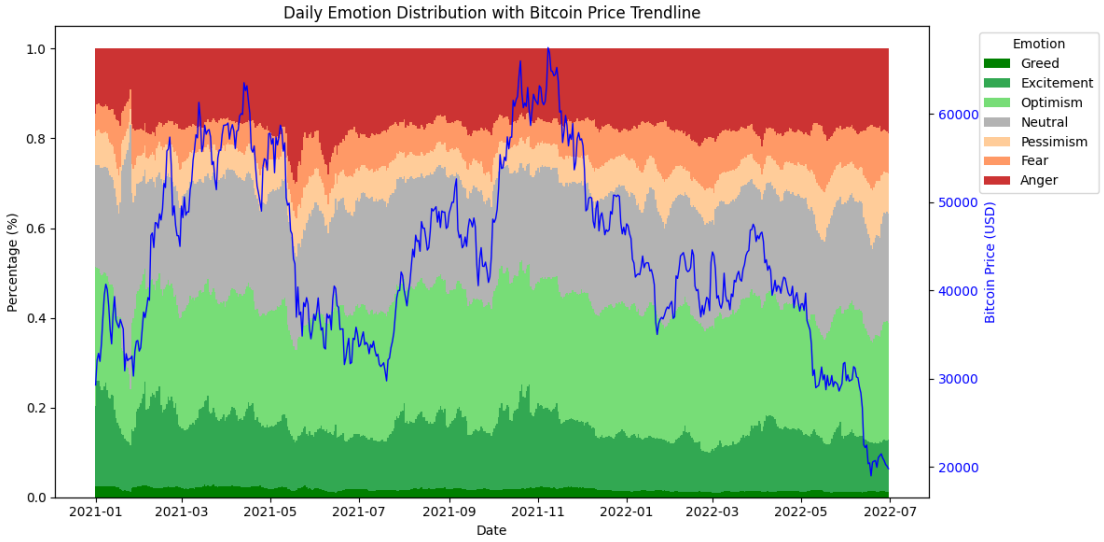


FIGURE 4.19: Daily emotions and bitcoin price relationship

One last step that will be applied to the emotions to make them easier for the model to digest is smoothing them by applying a rolling window with inverse log weights to still give more relevance to the newest values, which prevents the model from struggling with abrupt changes.

4.4.1 Model Structure

Although this is a time series prediction, making it a good fit for an LSTM-based model, having only about 2000 steps and those representing only 500 days makes it a questionable choice, as it is a model that requires a lot more data than a simpler CNN. But even if using a simpler CNN could give a slightly better prediction in this case than an LSTM, given that this project is just a sample to assess the viability of using this over a broader amount of data, using a simpler model would be misleading and would not give a real scale of the scalability of the project. Because of that, an LSTM-based model is the choice for it. However, some tricks and manipulations will need to be done to prevent overfitting or memorizing the data.

The model structure consists on two LSTM layers, a big one (64 dimensions) to ingest a window of the last previous days features to imply a sense of regression, letting the second LSTM layer (much simpler with 32 dimensions) see the day-to-day evolution of those abstract features and understanding what has been happening and the momentum that the currency has been following the last N days.

In order to prevent the model from overfitting and memorizing events, two dropout layers will be applied, one after each LSTM. What those do is randomly dropping 20%

of the hidden units during training to further encourage the network to build a more robust and redundant representation that generalizes better from the samples. This makes the model learn robust concepts that can be generalized instead of finding small loopholes that work extremelly well but on just a small batch of the data.

As another means to prevent overfitting, a kernel L2 regularization is applied to both LSTM penalizing overly large weights and preventing the model from "memorizing" random noise from the data instead of true leading indicators.

Finally, as the last step, since this is a decision model, a dense layer with a sigmoid activation function gives a direct signal of which decision to make, either buying or selling.

Chapter 5

Testing and Evaluation

With all the previous steps completed and the model defined, the last step is to run and train the model. In order to have a good visualization of the results, the test set must be quite large, so in this case it takes 25% of the data, with a 60-15% train-validation split. Also, in order to remove noise and focus mainly on the human reaction to the bitcoin price, the LSTM structure will only take into account the human written tweets.

And after training the model, a quite good looking result is obtained as seen in **Figure 5.1**.

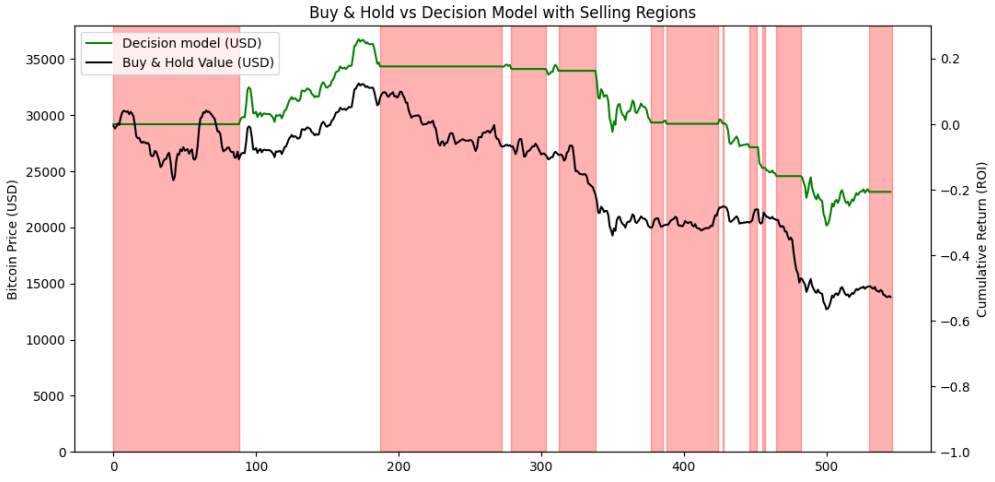


FIGURE 5.1: Price gain comparison between the Decision Model and Buy & Hold

But, although this could be the end of the research, this is not a robust result because of the high variability of the resulting models due to the complexity of the context, making it diverge easily to local minimums, and not giving the full scope of the paradigm, so this cannot be used as a generalization.

On the other hand, giving more patience to the model or letting it do random jumps in order to try to give it more means to find a deeper minimum is a bit dangerous in this situation given that the dataset is quite limited and would probably lead to overfitting.

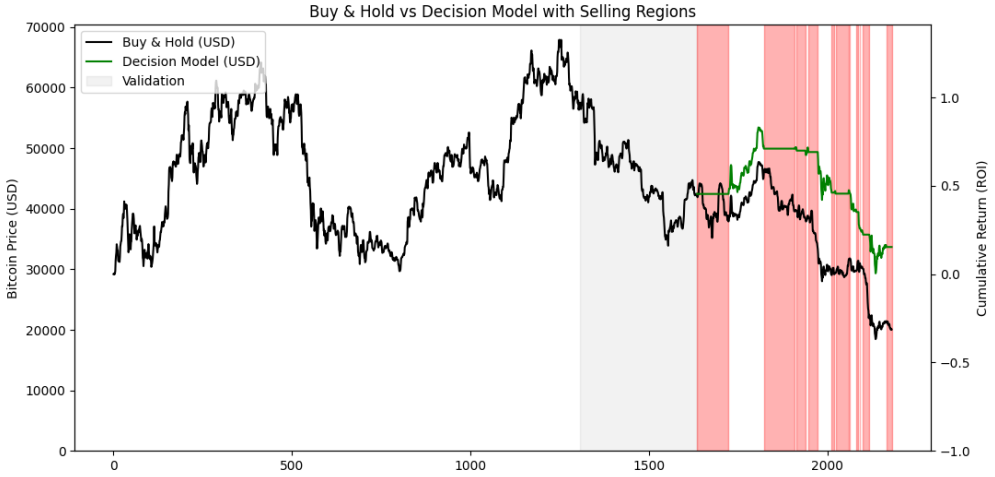


FIGURE 5.2: Price gain comparison between the Decision Model and Buy & Hold with training and validation dates

So, the only way to improve that and allow for better generalizations without overfitting would be to get more data. However, given that this is just a test to assess the viability of this kind of models, this is good enough and a more robust generalization would be to get the average gain for different trainings of this model.

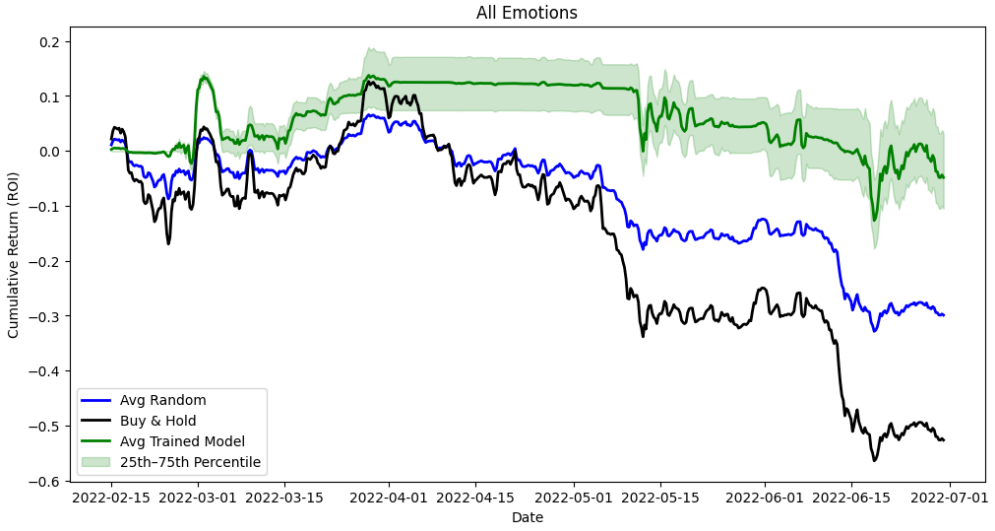


FIGURE 5.3: Comparison between Buy & Hold, Average Random and Average Model decisions

Furthermore, given that the test set follows a downstream, a more accurate way to compare the viability of the models would be to, instead of solely comparing them to

the Buy & Hold, also compare it to the average of random decisions. So in this case, this will make for the possibility of having a biased distribution of decisions, in this case towards 'sell'. So, as can be seen in **Figure 5.3**, the model seems quite robust and, in this case, it would be able to hold quite reliably during a downstream period.

By comparing these results with the results when only using Sentiment Analysis (in this case adding all the positive and negative emotions and making an average) as in **Figure 5.4**, the difference is noticeable, proving the thesis preamble that having only one metric to represent the Twitter's perception is not giving enough information.

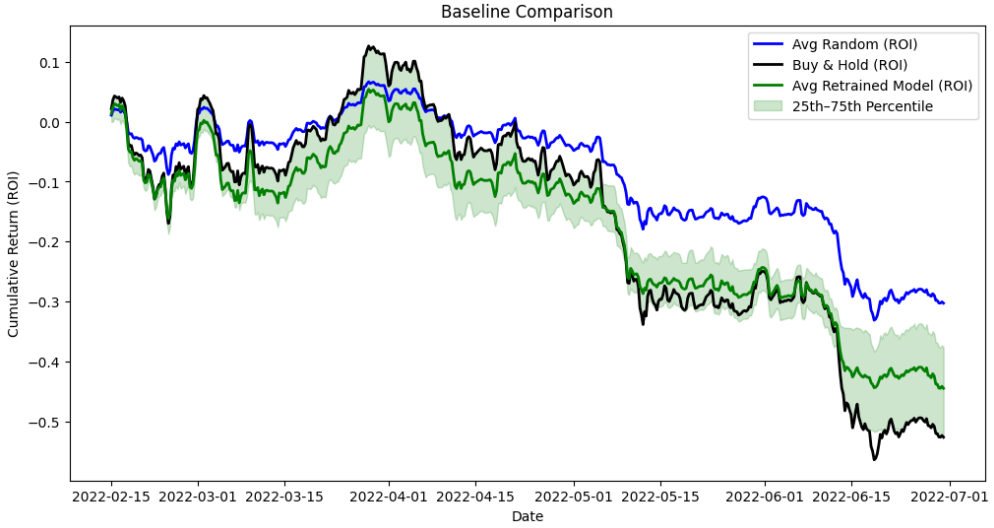


FIGURE 5.4: Comparison between Buy & Hold, Average Random and Average Model decisions for just Sentiment Analysis

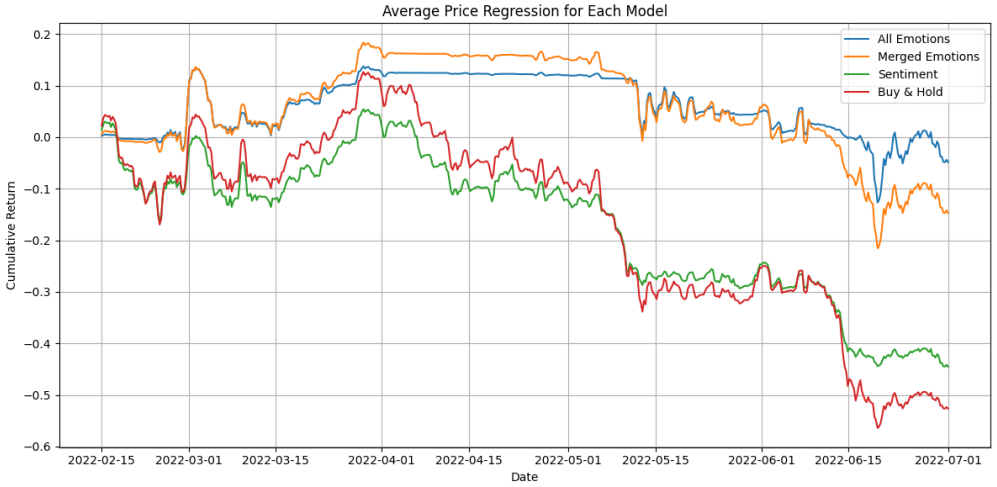


FIGURE 5.5: Comparison between the different model parameter inputs

And the same happens when joining emotions into clusters, in this case joining 'Fear' and 'Pessimism' into one emotion and the same with 'Greed' and 'Excitement', making it visibly noticeable that it is making it lose information (**Figure 5.5**).

Also, when testing only with the bitcoin data (so only previous day's price and IQR) it follows the Buy & Hold exactly (in **Figure 5.6** they perfectly overlap), probably due to the fact that, as can be seen in **Figure 5.2**, the training set is an upstream during almost all the way, motivating the model to make a "hold all" decision, given that due to the lack of data, it has to make an easy LSTM structure and that diverges fast to an all-hold.

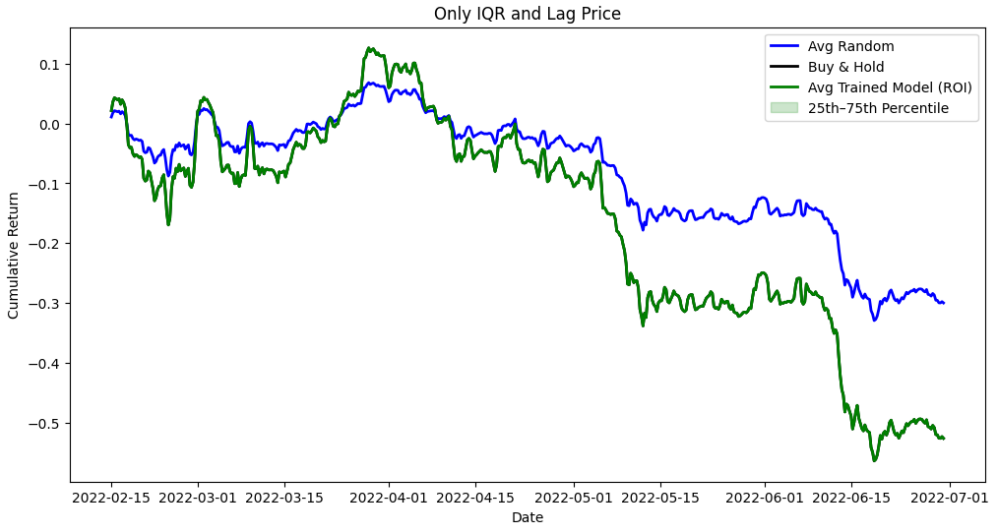


FIGURE 5.6: Comparison between Buy & Hold, Average Random and Average Model decisions (Buy & Hold not visible as it overlaps perfectly with the model decision)

So, as a numerical solution, the model with all the emotions, tweet count and IQR, holds quite well during the downstream period, losing only about 5% of the price, while the Buy & Hold loses more than 50% and the average random decision about 30% making that the best model. In comparison, merging the emotions as seen in **Figure 5.5** or reducing them to a mere sentiment value only reduces precision, ending up with a 15% and 45% loss respectively.

Chapter 6

Discussion and Conclusions

6.1 Discussion

As mentioned multiple times, although the results are extremely positive, this is not conclusive, as given the size of the data used and the clear positive skew of the train set compared to the negative one in the verification and test sets, gives a lot of uncertainty so further testing with more data could maybe give different results, but it gives a good starting point for a future scaled project.

It also proves (to an extent, taking into account the short evaluation span) that the emotion classification appears to be giving a more reliable prediction indicator than a simple sentiment value.

Furthermore, the two custom evaluated parameters, Tweet Count and IQR, show a direct correlation between price variability and the number of tweets as visible in **Figure 4.18**, which have a very similar trend and this is very good because it makes it viable to use it as a risk indicator.

6.2 Conclusion

Because of that, the main conclusion that can be drawn is that indeed, the emotions extracted from Twitter do have a correlation with the bitcoin price trend and risk, making it seem as a good indicator worth investing some money to exploit and study forward, in case someone wanted to make an investing decision model.

So, this is a very good first step and a proof of concept to assess the viability of this structure for a future scaled model with more information (which would come with a

higher cost of scraping and training). It also debunks the thought that having almost 50% better returns than Buy & Hold doesn't mean that this would be the average gain of the model, as a great part of that is given due to the bias due to the training and testing sets, one being positively skewed and the other very negative, making it so that even a random decision (as pictured in **Figure 5.3**) produces a better result.

Finally, referencing the research objectives, they were all fulfilled. To begin with, a functional Twitter scraper was created and used to make the Tweet count during the working period. The second one also got completely fulfilled making two very reliable models, one for emotion classification and the other for spam detection (made to clean the data before the emotion detection), and as of the last objective, a very robust decision model has been developed, open to be trained with more data, but already giving quite promising improvements.

6.3 Future Work

This leads to a clear room for improvement for future work. First of all, as clearly stated, an increase on the fed data and the time span would give a more broad diversity of price trends, letting the model learn from them without falling into memorizing patterns.

Another improvement that could be used for multiyear studies would be to add some year cycle indicators (usually done by adding a sine and cosine functions with year long cycles) to indicate a year-long distribution, which could help the model learn the yearly patterns such as the increase and decrease in variability between summer and winter periods as seen in **Figure 4.15b**.

Also, in case of future scraping, paying attention to a tweet's scope can help determine how much it impacts its audience, as a tweet written by someone with just a few followers has a completely different reach compared to the one of a celebrity with a great following base and effect on the crypto market as of Elon Musk.

The dataset used for this project was an already scraped public dataset and it did not provide anything other than the text, making it impossible to use it, but for future scrapings it would be a great magnifier.

The use of a Transformer-based model could also be beneficial (given a dataset with enough data to make it viable), as this has been proved to give better, although it requires a lot more training data.

And finally, adding some more bitcoin indicators as trade volume could have given a good idea of movement and variation but, given that the bitcoin dataset used was coming

solely from one broker, it could have added some bias so it did not get added, but for future projects, by using multiple datasets or more relevant brokers, that bias could be reduced to a point where it could become reliable to use.

Bibliography

[1] J. S. Lerner and D. Keltner, "Beyond valence: Toward a model of emotion-specific influences on judgement and choice," *Cognition and Emotion*, vol. 14, no. 4, pp. 473–493, 2000.

[2] ——, "Fear, anger, and risk." *Journal of Personality and Social Psychology*, vol. 81, no. 1, p. 146, 2001.

[3] N. Abu Bakar and S. Rosbi, "Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction," *International Journal of Advanced Engineering Research and Science*, vol. 4, no. 11, pp. 130–137, 2017.

[4] C. Conrad, A. Custovic, and E. Ghysels, "Long- and short-term cryptocurrency volatility components: A garch-midas analysis," *Journal of Risk and Financial Management*, vol. 11, no. 2, 2018.

[5] H. Sebastião and P. Godinho, "Forecasting and trading cryptocurrencies with machine learning under changing market conditions," *Financial Innovation*, vol. 7, pp. 1–30, 2021.

[6] M. H. Bin Mohd Sabri, A. Muneer, and S. M. Taib, "Cryptocurrency price prediction using long short-term memory and twitter sentiment analysis," in *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pp. 1–6.

[7] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1419–1426.

[8] A. Yousaf, M. Umer, S. Sadiq, S. Ullah, S. Mirjalili, V. Rupapara, and M. Nappi, "Emotion recognition by textual tweets classification using voting classifier (lr-sgd)," *IEEE Access*, vol. 9, pp. 6286–6295, 2021.

[9] Y. Chandra and A. Jana, "Sentiment analysis using machine learning and deep learning," in *2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1–4.

[10] J. Park and Y.-S. Seo, "Twitter sentiment analysis-based adjustment of cryptocurrency action recommendation model for profit maximization," *IEEE Access*, vol. 11, pp. 44 828–44 841, 2023.

[11] D. Lazeski, "Stock market prediction: A multiclass classification on emotions and sentiment analysis for tweets and news headlines," 2020.

[12] C. H. Liu, "Applications of twitter emotion detection for stock market prediction," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[13] M. Othman, E. Nasr, J. Nasr, and L. Karam, "Impact of crypto art sentiment on art valuation," in *2023 IEEE 4th International Multidisciplinary Conference on Engineering Technology (IMCET)*, pp. 259–263.

[14] N. Aslam, F. Rustam, E. Lee, P. B. Washington, and I. Ashraf, "Sentiment analysis and emotion detection on cryptocurrency related tweets using ensemble lstm-gru model," *IEEE Access*, vol. 10, pp. 39 313–39 324, 2022.

[15] S. Rasiya Koya and T. Roy, "Temporal fusion transformers for streamflow prediction: Value of combining attention with recurrence," *Journal of Hydrology*, vol. 637, p. 131301, 2024.

[16] A. J. Amadeo, J. G. Siento, T. A. Eikwine, Diana, and I. H. Parmonangan, "Temporal fusion transformer for multi horizon bitcoin price forecasting," in *2023 IEEE 9th Information Technology International Seminar (ITIS)*, pp. 1–7.

[17] J. K. Chiang and R. Chi, "A novel stock price prediction and trading methodology based on active learning surrogated with cyclegan and deep learning and system engineering integration: A case study on tsmc stock data," *FinTech*, vol. 3, no. 3, pp. 427–459, 2024.

[18] L. Harting and N. Åkesson, "A machine learning approach leveraging technical-and sentiment analysis to forecast price movements in major crypto currencies," 2022.

[19] P. M. Fung, A. C. Leung, A. W. Ng, B. C. Wan, and I. T. Yim, "Bitcoin trading strategies using news and tweets with sentiment analysis," 2019.

[20] S. Otabek and J. Choi, "Multi-level deep q-networks for bitcoin trading strategies," *Scientific Reports*, vol. 14, no. 1, p. 771, 2024.

[21] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.