UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Cryptocurrency Price Prediction using Sentiment Analysis

Diploma Thesis

Fivos-Ioannis Tzavellos

Supervisor: Michael Vassilakopoulos

Volos 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Cryptocurrency Price Prediction using Sentiment Analysis

Diploma Thesis

Fivos-Ioannis Tzavellos

Supervisor: Michael Vassilakopoulos

Volos 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη Τιμής Κρυπτονομίσματος χρησιμοποιώντας Ανάλυση Συναισθήματος

Διπλωματική Εργασία

Φοίβος-Ιωάννης Τζάβελλος

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Βόλος 2021

Approved by the Examination Committee:

Supervisor	**Michael Vassilakopoulos** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Panagiota Tsompanopoulou** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Eleni Tousidou** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

First of all I would like to thank my supervisor Prof. Michael Vassilakopoulos for his assistance and support in developing this thesis, as well as Prof. Elias Houstis for his valuable advice. I would also like to thank Prof. Panagiota Tsompanopoulou and Prof. Eleni Tousidou for being a part of the examination committee. Last but not least I would like to express my warmest gratitude to my family and friends for supporting me throughout all these years in good times and bad times.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Abstract

In the last few years there has been an increased interest in cryptocurrencies and the blockchain technology. It is getting harder and harder to imagine a future without the world-wide adoption of cryptocurrencies. The extreme volatility that is observed in the prices of almost all the cryptocurrencies highlights the importance of predicting the future movement of the price inside a specific window in time. It could prove a very profitable investment to be able to forecast the price in a precise manner. This thesis takes steps towards this direction by extracting tweets concerning Bitcoin, by processing them and removing the ones that are considered automated messages from botted accounts through the use of a filter constructed for this purpose, by performing Sentiment Analysis with Vader while simultaneously including some cryptocurrency specific terminology in the calculations, by visualising the Pearson, Kendall and Spearman correlation between the results of the aforementioned sentiment analysis and the price of Bitcoin and by extracting real time data, feeding them to the XGBoost algorithm and using them alongside other features to predict the price in the next minute. The results displayed through graphs and measured according to certain error metrics seem quite promising, with a Root Mean Squared Error score of 27 USD for the full dataset and 24.5 USD for the dataset without bots.

Περίληψη

Τα τελευταία χρόνια παρατηρείται αυξημένο ενδιαφέρον σχετικά με τα πεδία των κρυπτονομισμάτων και της τεχνολογίας του Blockchain. Γίνεται όλο και πιο δύσκολο να φανταστούμε ένα μέλλον χωρίς την παγκόσμια υιοθέτηση των κρυπτονομισμάτων. Η ακραία μεταβλητότητα που παρουσιάζουν οι τιμές σχεδόν όλων των κρυπτονομισμάτων τονίζει την σημαντικότητα του να προβλέπουμε την μελλοντική κίνηση της τιμής μέσα σε ένα συγκεκριμένο χρονικό πλαίσιο. Θα μπορούσε να αποδειχθεί μία πολύ κερδοφόρα επένδυση να μπορεί κανείς να προγνώσει την τιμή με ακριβή τρόπο. Η διπλωματική προχωράει προς την συγκεκριμένη κατεύθυνση με το να εξαγάγει tweets σχετικά με το Bitcoin, να τα επεξεργάζεται και να αφαιρεί αυτά τα οποία θεωρούνται αυτοματοποιημένα μηνύματα από λογαριασμούς με bots μέσω της χρησιμοποίησης ενός φίλτρου ειδικά κατασκευασμένου για αυτόν τον σκοπό, να πραγματοποιεί ανάλυση συναισθημάτων με το εργαλείο Vader ενώ ταυτόχρονα συμπεριλαμβάνει στους υπολογισμούς ειδική ορολογία σχετική με τα κρυπτονομίσματα, να οπτικοποιεί την συσχέτιση Pearson, Kendall και Spearman μεταξύ των αποτελεσμάτων της προαναφερθείσας ανάλυσης συναισθημάτων και της τιμής του Bitcoin και να εξαγάγει δεδομένα σε πραγματικό χρόνο, να τα τροφοδοτεί στον αλγόριθμο XGboost και να τα χρησιμοποιεί μαζί με άλλα χαρακτηριστικά για να προβλέψει την τιμή στο επόμενο λεπτό. Τα αποτελέσματα που παρουσιάζονται μέσα από γραφήματα και μετρούνται με συγκεκριμένες μετρικές σφαλμάτων φαίνονται αρκετά ενθαρρυντικά, με ένα Root Mean Squared Error των 27 αμερικανικών δολαρίων για ολόκληρο το dataset και 24.5 δολαρίων για το dataset χωρίς τους λογαριασμούς από bots

Table of contents

List of figures

Abbreviations

SA	Sentiment Analysis
e.g.	exempli gratia - for example
NaN	Not a Number
et al	et alia - and others
etc	et cetera - and the rest
USD	United Stated Dollars
BTC	Bitcoin

Chapter 1

Introduction

The field of cryptocurrencies has been growing and evolving at a very fast pace, with a rapidly increasing impact in the world both from a social and from an economical aspect. Bitcoin has been the number one cryptocurrency in terms of popularity and market capitalization (which is the amount of coins in circulation multiplied by the price of one coin - the most important metric when it comes to economic influence). In March 2021 it reached its all-time high in terms of market cap (1.1 billion USD), a 1000% increase compared to the same time in the previous year [1]. It holds the reins in the market of cryptocurrencies and acts as a trend-setter and an anchor that single-handedly drives almost every other coin's prices up and down depending on its own price. This in fact was the reason we are focusing on Bitcoin for this thesis, if you can predict its price then you can predict the movement of prices for the altcoins aswell (coins that came after Bitcoin and follow its successful model). This frenziness when it comes to Bitcoin highlights the importance of accurately prognosticating the price of certain cryptocurrencies (whether they will increase or decrease in the near future) since they can prove to be an asset that will help individuals that hold this kind of knowledge financially. The problem with the traditional economic methods of price forecasting is that they are outdated, slow and inflexible. Here is where machine learning comes and knocks on the doors of the marketplaces (virtual stores where cryptocurrency is exchanged) with some very promising tools to showcase. Another problem that emerges , when it comes to the modern methods that employ machine learning, is that there are a lot of obstacles that hinder the efforts to predict the price in a precise manner (namely concerning the data and the methods employed).

1.1 Purpose of the Thesis

In our work, we utilize tools like sentiment analysis to overcome these aforementioned problems. As far as speed is concerned, by using the XGBoost algorithm to train the model and perform the predictions we manage to get a fairly accurate prediction for the next minute price of Bitcoin in just mere seconds. We can basically say that part of the algorithm is working in real time since we gather the tweets through a stream as soon as they are posted, process them immediately and use them to aid our efforts. One of the obstacles that gets encountered when dealing with tweets is the plethora of Twitter bots in circulation. We deal with this problem by detecting the bots and removing them from our training set. Another obstacle that particularly concerns sentiment analysis methods is the limited vocabulary available for the calculation of the sentiment score of the sentences. We solve that by including cryptocurrency related terminology to our sentiment analysis lexicon.

1.1.1 Contribution

The contribution of the thesis can be summarised as:

1. We extracted tweets using the Twitter API.

2. We removed tuples by accounts that exhibited suspicious bot-like activity by using special filters constructed just for this purpose.

3. We performed sentiment analysis with VADER and by using the tools provided by VADER we added some extra cryptocurrency related jargon words to the already existing lexicon.

4. We visualised a lot of results and data segments by using graphs like WordClouds, barplots etc.

5. We observed the correlation between the prices and the sentiment scores through the use of graphs

6. We ran XGBoost for the prediction with some very promising results

1.2 Related Work

Panger [2] suggested that users' general emotional state correlated with Twitter sentiment and that Twitter had a calming effect on them, rather than amplifying their state of emotions. O'Connor et al [3] notice a correlation of around 80% between the public's polled opinion and sentiment derived from text. These papers make it clear for us that users' emotions and the opinions of the masses affect Twitter sentiment and in turn offer a good incentive for us to venture more into the field of taking advantage of sentiment analysis for financial predictions.

In regards to the stock market, Tetlock [4] found that high media pessimism was linked to downward movement in stock prices and extreme pessimist (either high or low) leads to high market trading volume. Yuat el al [5] collected the emotional valence (the cumulative sentiment response) of Twitter posts concerning S&P 500 companies and found that there is a relation between that and the stock price of the aforementioned companies. De Jong et al [6] measured that 87% of the minute by minute stock returns for the 30 stocks in the Dow Jones Industrial Average were influenced by lagged innovations of the tweets data for the same stocks. However when they tried to prove that the inverse is also happening they found that only 7% of tweets were being influenced by lagged innovations in stock returns.

Aside from the stock market there has also been a number of attempts to predict the price of cryptocurrencies using Twitter sentiment analysis and an extensive number of other methods. Our inspiration for this thesis and first encounter with the subject came mainly from the work of Mohapatra et al [7], where they developed a real time cryptocurrency price prediction platform based on sentiment analysis performed on tweets with an overall RMS (Root Mean Squared) error of 10 dollars between the actual and the predicted price of Bitcoin. Their innovation comes from the fact that they update their model in real time using the data as they come from the Twitter stream after some preprocessing, utilizing Apache Spark's platform to store, manage and secure the data, providing the much needed scalability and fault resistance.

Abraham et al [8] predict changes in Bitcoin and Ethereum prices using Google trends data (how popular certain search terms are compared to others) while stating that the usual method of performing sentiment analysis on tweets can prove to be inefficient (a statement that we aim to disprove), since as they say tweets are generally positive regardless of the price. They use a combination of Google Trends data and tweet volume, both of which appear to be highly correlated with the price (above 0.8 Pearson R), as input to the linear model, which appears to show promising results, following the trend of the actual price.

Lamon et al [9] use cryptocurrency related news article headlines to predict the price of three different cryptocurrencies (namely Bitcoin, Ethereum and Litecoin), by labeling the data based on price changes with a time window of one day, without needing to predict sentiment.

Stenqvist et al [10] focuses on predicting the rise and fall of Bitcoin's price based on sentiment fluctuations in tweets using different windows in time and different lag (time shift). They deployed the method of manually identifying suspicious n-grams and using them to detect bots, thus reducing the whole dataset by 55%. This in part inspired us to create part of our bot detection filter. It achieved a 79% accuracy with a certain set of parameters, but the main issue is that the prediction is of binary nature, it only predicts if the price will go up on down which is not ideal if you want to use the results obtained to invest.

Kraaijeveld et al [11] use tweets' sentiment scores derived from VADER, price data and the bivariate Granger causality test to gauge the results for 9 different cryptocurrencies. They discover that predictive power can exist when it comes to Bitcoin, Bitcoin Cash and Litecoin. They also employ bot detection using heuristically built filters and a cryptocurrency specific word list, as well as the addition of some terminology coming from a financial corpus. This paper was a point of inspiration for us when it came to developing some methods to improve our computational results.

Valencia et al [12] test multiple machine learning tools to predict the price movement of Bitcoin, Ethereum, Ripple and Litecoin. Among the tools and algorithms used there are Multi-Layer Perceptrons with around 72% accuracy when using price and Twitter sentiment data and 40% when using solely Twitter data, Support Vector Machines with 55% accuracy and Random forest with 39% (all the above as far as Bitcoin is concerned). The Neural Network architecture seems to outperform every other method on every cryptocurrency prediction tryout and it seems like we could even use Twitter data alone to predict, with a little above 50% accuracy.

The research presented here in this paper builds off the above but is unique in the sense that we combine methods taken from various papers, while also implementing some personal ideas and trying to improve the existing ones, into one with surprisingly good results.

1.3 Organisation of the Thesis

Some brief explanation and description of the basics of the topics that we will be dealing with can be found in Chapter 2. Chapter 3 is about the data, how we collected it, what it consists of, some concerns regarding privacy and some visualisations. In the next chapter, namely Chapter 4, we describe all the methods used to solve our problems, be it bot detection, data preprocessing, sentiment and correlation analysis and prediction. The results of all these methods can be found in Chapter 5, where we can clearly judge if the outcome is satisfactory. In the last chapter, Chapter 6, we conclude our thesis with some comments and some suggestions about moves that could be done in the future to further improve our work.

Chapter 2

Background

In this chapter we will set the basis of our research by explaining in detail the terms and the concepts that we will be dealing with. In order to fully understand what comes next, we first need to talk about Twitter and the Cryptocurrency and Blockchain technology, with our focus shifter towards Bitcoin, after that we need to talk broadly about Machine Learning and specifically about Sentiment Analysis and then we will explain in detail the specific tools used, which are Jupyter Notebook and Google Colab, VADER and XGBoost. We will provide a background for these terms in order to provide the reader with an understanding as to why we chose this topic, why we chose to use these specific tools and how it all works.

2.1 Twitter

Twitter is a social media platform founded in March 2006 in San Francisco California USA by a group of entrepreneurs – coders. It is an American based micro-blogging (i.e. the concept of posting very short status updates [13]) and social networking service with a some-to-many style (the vast majority of tweets is written by a small minority of users) . The original idea that the creators had and the intended use of the platform was as a medium to broadcast thoughts, feelings, and ideas. It was essentially a free SMS service with a social networking aspect. However, it naturally shifted towards a more conversational style. The thing that helped Twitter evolve and pivot from its original concept was its use as an amateur journalism tool, playing a big role in the American elections in the coming years, as well as helping document and broadcast major political events in the whole world.

Twitter at the time of writing has around 320 million active users [14], with 192 million of them being active daily [15], a 27% increase over the previous year, mainly due to the pandemic and some development efforts. In comparison, Facebook boasts over 1.84 billion daily active users. Breaking down the user-base, Twitter is mostly used by males, aged 35 to 65 and predominantly from the US (30% of the user-base).

The messages used in Twitter, called tweets, can be up to 140 characters long and can include hyperlinks and types of media like photos, videos and GIFs. Hashtags can be added to the tweet (the symbol '#' followed by a word) to refer to a specific topic and identify it and to make it searchable. The limitations in the length of the tweets add to the snappy nature of Twitter, allowing users to refrain from reading long content items on their smartphone or PC screens. Tweets are fully public and can be accessed by anyone, they are permanent and searchable, a feature that proves really useful for researchers that want to document human thoughts and emotions or are interested in polling opinions and recording historical events through the eyes of the people that are witnessing them.

The platform offers a vast selection of tools for developers that are interested in using Twitter data, called the Twitter Developer Platform [16]. It enables developers to extract tweets, look up users and statistics, record trends, geographically locate conversation topics and a lot more. The data could be obtained in real time or historically through the Twitter archive. Researchers can mine tweets based on specific hashtags, obtain information like the author, the author's follower count, the tweet's Retweet and Like count, the time, the geographic location, a unique identification number for the tweet and much more. All this, along with the classifying nature of tweet hashtags and the restrictions to message length enables us to make use of the resources that Twitter provides and collect textual opinionated data in a semi-structured form referring to a topic (in this case Bitcoin) and to utilize them to obtain certain results. Ultimately, we chose Twitter over other social media platforms for our source of data because of the plethora of polarizing text that gets published there , which can be easily obtained through the easy-to-use API.

2.2 Cryptocurrency and Blockchain

The first time the term 'cryptocurrency' emerged was in 1998 [17]. The term refers to an asset of the digital world that is used as a means of exchange. All records are stored in a ledger with the use of cryptography to secure the transactions. This ledger is often called a blockchain and most cryptocurrencies are based on this technology. Essentially, blockchain is a type of database that uses blocks to store information and those blocks are chained together. Every transaction exists on one specific block and the way to validate the blocks and add them to the chain differs based on the cryptocurrency we are talking about. We will explain Bitcoin's method in detail, as it is the most popular cryptocurrency and the one we will be basing our thesis on. The potential power of blockchain technology lies in a form of distribution associated with a technically valid equivalent of 'inter-subjective agreement'. Just as in spoken and written language the meaning of a word remains stable because of the agreement of multiple users for the validity of that word, blockchain 'democratises' agreement that a certain state of affairs exists.

The appeal of cryptocurrencies in general exists due to the fact that they possess a certain set of qualities, as well as some drawbacks which will be discussed extensively. These qualities derive from the nature of the cryptocurrencies, from the way in which they were formed and based on the source code that entails their use.

First of all they are **secure**, due to being based on cryptographic algorithms that would take years to decipher. Each cryptocurrency relies on a different algorithm to secure its records. For instance Bitcoin - our point of focus in the next section - uses SHA-256 (Secure Hashing Algorithm 256) to verify transactions. It is very difficult, nearly impossible, to double-spend or to counterfeit this type of currency. In addition, the disaggregation of data across a distributed network of nodes (i.e., computers) provides security against attempts to destroy or change the record of transactions.

Secondly, they are **decentralised**, which means they are not governed by a single entity that acts based on profit - for example a bank or a government. The network's type is peer to peer, i.e. information and assets are exchanged between parties without the involvement of a central authority. In other words the middle-man is being cut out of the transactions, with the parties placing mutual trust on one-another. Using direct transactions, blockchain technology can streamline processes by cutting out unnecessary intermediaries and process steps, as well as reduce the risk of errors that usually come with extra transactions in a system.

Transactions become **transparent**, **traceable** and **verifiable**. It is a known fact that the complete record of Bitcoin transactions can be verified. At its core, blockchain is a digital recording system. It offers anyone that requests it entire visibility into the history of every transaction ever made using the specific cryptocurrency. There exists the possibility to track transactions made by anyone, even by the so called 'whales', accounts that accumulate currency and transact money in the scale of millions of dollars.

Another redeeming quality is that it is **immutable**. The transactions recorded on the blockchain cannot be changed or removed. This, working together with strict governance rules and cryptography provides strong security for individuals interacting directly on a distributed network without a central trusted authority, making the system **trustworthy**.

Pseudonymity offered by the network means that by using public and private key systems, participants have a public-facing digital "address" that is not publicly associated to them, but over which they exercise unique control, without revealing their true personal data.

All this is achieved by removing the risk and the friction that arises out of dependence on just one or a few nodes of authority. The character of blockchain is inherently anti-authoritarian and it could revolutionize everything for billions of people around the globe, from worldwide financial markets and the distribution of humanitarian assistance to the very way we recognize human identity.

Of course there is also the other side of the coin. There are some drawbacks and risks involved in the use of cryptocurrencies. Namely, cryptocurrency has been used to fuel illegal activity, due to the anonymity it provides [18]. The fact that users are able to execute transactions without leaving a trace becomes extremely beneficial to activities on the dark web. It can also be used by governments or other entities to exert and consolidate power over people and information. On the one hand the crypto-economic systems can increase financial inclusion and create innovative microeconomies, on the other hand these structures could also create exploitative systems with perverse incentives or undermine existing payment and monetary systems that have the virtue of being understood and accepted within formal financial markets. In addition, when every information is transparent and available to everyone, there is the risk of getting prosecuted or exploited, based on the knowledge of someone's race, religion or sexual identity. Another problem posed is that as computational techniques and computer power continue to evolve at a rapid pace, it becomes increasingly difficult for encryption algorithms to stay ahead of the technology to break through encryption. If immutable and distributed information on a blockchain is encrypted with outdated algorithms, that information may become vulnerable to exposure. Blockchains built for long-term applications, such as land registries, must also consider the possible effects of quantum computing to amplify this threat through its projected ability to break through any non-quantum-proof digital signatures used on blockchains and forge transactions [19].On another note, losing your private key would mean there is no way to access the information you have stored in the blockchain and thus you lose access to your assets. Lastly, one of the most pressing issues that cryptocurrencies pose is that they are taking an increasing toll on the environment. The electrical energy consumption required to mine (create) these type of currencies is very big. Research suggests that Bitcoin consumes around 121.36 Terawatt-hours (TWh) a year - a figure higher than the consumption of Argentina for the same duration [20].

We can consider these risks as an opportunity for cryptocurrencies to alleviate the problems and find room to grow even more. The promise of blockchain to have an impact on millions of people is real. Its key attributes of transparency, trust, and immutability have the potential to improve lives across the globe. By increasing efficiency, security, and verifiability in the way that social impact organizations operate, access to services is delivered, data is stored and controlled, and assets are tracked, blockchain's potential can literally change the world.

2.3 Bitcoin

Bitcoin is undoubtedly the pioneer of cryptocurrencies, the first decentralised one. A person (or group of people) with the alias Satoshi Nakamoto published Bitcoin's whitepaper in 2009 [21] and wrote its software which is publicly available for anyone who wishes to study it (it is considered open source). Bitcoin is a peer-to-peer network using cryptography to verify transactions through network nodes and a blockchain to record them in a public distributed ledger. Bitcoins are created through a process called mining, where different miners compete who will be the fastest to perform a mathematical operation to solve a problem that has a certain cost. The first one to solve this problem gets to validate the block and gets rewarded in bitcoins (6.25 bitcoins as of 2021). This happens once every 10 minutes (the creation time of the block), and it puts a bottleneck on Bitcoin's transaction processing capacity. So in reality, miners trade computational power and electric energy for bitcoins. This operation is called proof of work, and it also helps the network guard itself towards an attack. Whoever wants to forge or alter a transaction has to be in charge of 51% of the network's nodes. If he is successful in controlling such a big part of the network, it is more beneficial for him to join the other miners in validating blocks than to go against them and harm the network.

Bitcoin mining used to be treated as a hobby, an opportunity for self-employment for some computer-inclined youngsters. Now it is being dominated by mining farms with high-end asic cards. In fact around 54% of the world's bitcoin computed power is located in Sichuan, China [22] The real world value of Bitcoin is extremely volatile, reaching an all-time high of around 65.000$ in mid April 2021 and down to 32.000$ in mid July of the same year. Bitcoin's price acts like an anchor, pulling down all other cryptocurrency prices with it as it falls. Since it is the cryptocurrency with the highest market capitalisation as of July 2021, it has the influence to drive the market up and down, and that is part of the reason why we chose it to be the point of focus of this study, along with the fact that it has the most mentions out of every cryptocurrency on Twitter. Despite its volatility it has been coined the digital version of gold, a very solid investment for large purchases in the long run.

2.4 Machine Learning

Machine Learning is a buzzword that has been going around a lot in the last few years, for a good reason. It is a branch of artificial intelligence and it is based on the idea that the machines (systems) can learn from actual data fed to them, get trained, and produce results, sometimes in the form of predictions. In the field of cryptocurrency price prediction, methods from several machine learning fields can be deployed in order to get the desired result. The choice falls on the researcher to make based on the data and resources available and the direction he/she chooses to follow.

2.5 Sentiment Analysis

Sentiment analysis is a machine learning tool, falling in the category of Natural Language Processing (a collection of methods for computers to analyze and understand text [23]). It is the computational study of people's opinions, attitudes and emotions towards individuals, events or topics [24]. Polarity, either positive, negative or neutral is extracted from text after a certain analysis and used in a way that leads to certain results. It has wide application fields and until recently it was used extensively in the field of marketing. The challenges [25] that sentiment analysis creates are:

- The fact that tone is hard to convey through the means of written text. This is alleviated through the use of high level software that detects and properly handles tone.

- The fact that some words are polarised - either fully positive or full negative - while some others are in the middle somewhere, e.g. less positive than the full positive words. The solution is provided by the use of a lexicon that holds the sentimental value of certain words

- The fact that sarcasm is hard to detect. We were contemplating a bit on creating a sarcasm detection tool to be used in this thesis but we decided against it due to the small number of tweets in our dataset that are actually sarcastic.

- The fact that emojis exist in large number in textual data. Vader specifically cares for emojis and assigns a sentiment score to each one of them.

- The fact that idioms exist and make it hard for algorithm to separate the literal sense of the expression from the simile. This is solved by adding these special expressions to the lexicon of already existing words of the sentiment analysis tool.

There are three main classification categories in Sentiment Analysis: document-level, sentence-level, and aspect-level SA. **Document-level SA** is about classifying whether an opinion document expresses a positive or negative opinion or sentiment. **Sentence level SA** focuses on each sentence individually and expresses its sentiment. Firstly it identifies whether the sentence is subjective or objective. In the case of the sentence being subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions. If the sentence is objective, then it merely states facts, providing information without a positive or negative connotation to them. Documents are basically short sentences so there is not much of a difference between document-level and sentence-level SA. **Aspect-level SA** classifies the sentiment with respect to the specific aspects of entities.

For our research we use VADER (Valence Aware Dictionary and sEntiment Reasoner) [26], a system which we will describe in detail in a following section. What we aim for is to apply sentence-level sentiment analysis on textual data gathered from Twitter in order to extract sentiment, the positive, negative or neutral opinion that people hold regarding Bitcoin at a specific moment in time.

2.6 Vader

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool developed by Hutto and Gilbert [27] that is specifically attuned to sentiments expressed in social media. The developers stated [26] that it works well on social media style text but can generalize as well to other domains and that it requires no training data since it operates like a lexicon which is ready to use at any moment, even with streaming data. It is fully open-sourced, which gave us the chance to get a good grasp of how it works and realise that with some additions it can really prove beneficial for our particular project. It is sensitive to both the polarity (positive, negative or neutral) and the intensity of sentiments, with a focus on the social media context. The lexicon of words and their correspondent sentiment score was formed when independent trained raters gave ratings to 9000 words - on a scale from ”(–4) Extremely Negative” to ”(4) Extremely Positive”, with (0) being ”Neutral”. Some rules were formed by the developers by examining and analysing the syntax and grammar aspects of 800 tweets, and those rules in addition to the lexicon constitute the Vader analysis tool. Each sentence (in our case each tweet text) is broken down to words and each word is assigned its sentiment score. All the sentiment scores of the individual words are summed up and that is how we get the overall score of the sentence. The innovation of Vader relies on the fact that it is specifically tuned to work in a social media context, assigning scores to emoticons and emojis, treating contractions, negations, punctuation and capitalization properly and also having support for slang words (not in the cryptocurrency context, an issue which was dealt with in a manner that we will explain in a following chapter). For our project we make use of the compound score which is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive) (Figure 4.5, Figure 4.6). This is the most useful metric if you want a single uni-dimensional measure of sentiment for a given sentence. There are also the pos, neu, and neg scores, those are ratios for proportions of text that fall in each category (so these should all add up to be 1). These are the most useful metrics if you want to analyze the context and presentation of how sentiment is conveyed or embedded in rhetoric for a given sentence. Vader is used extensively in many projects that require sentiment analysis of data mined from social media, and it consistently outperforms every other analysis tool and technique for polarity classification in the social media context [26].

2.7 Python, Jupyter Notebook and Google Colab

The language used for the entirety of the code of this thesis is Python. Python was first released in 1991 and has since become one of the most popular programming languages in the world. What makes it so appealing when it comes to artificial intelligence projects and the reasons we chose it specifically for this thesis are the fact that is simple, concise and easy to use, it has a plethora of libraries for machine learning (we used some of them like Numpy and XGBoost) and a lot of support (in pages like TowardsDataScience and StackOverflow, which we used extensively to troubleshoot issues).

The platforms we used to write the code are Jupyter Notebook and Google Colab. Jupyter Notebook is an open-source web application - platform that allows users to run code for all sorts of data science tasks and present it in a clear manner. We can think of it like a digital notebook that allows you to combine code, images, markdown comments, plots and much more. Google Colab includes Jupyter Notebook in its platform but the difference is that it offers us computational power and RAM that is not tied to the limitations imposed by our local machines. In this way we were able to run a part of the code that demanded specifications which we did not possess in our personal computers.

2.8 XGBoost

XGBoost [28, 29] is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The name stands for eXtreme Gradient Boosting. It is specifically suited for time-series problems that require execution speed and good model performance. In order to really understand what it does we need to break down each word in its name. Ensemble learning is a type of machine learning that enlists many models to make predictions together. Boosting is an ensemble technique where new models (initially weak) are added sequentially to correct the errors made by the previous ones until no further improvements can be made (and the models have become powerful) [30, 31]. Gradient Boosting is the process of creating new models to predict the errors or the residuals of the previous models and then adding them together to make the prediction. The name gradient comes from the use of the gradient descent algorithm to minimize the loss when adding new models. When it comes to tabular or structured data of a small to medium size the decision tree algorithms tend to perform exceptionally well [32].

XGBoost uses decision trees, which are models that construct a graph that examines the input under various 'if' statements. Whether the 'if' condition is satisfied influences the next 'if' condition and the algorithms adds more 'if' conditions to the tree to build a stronger model and reach the prediction. What XGBoost does is consider the leaves (the part of the tree that represents the class label- the decision taken after computing all attributes) of the current decision tree and questions whether turning that leaf into a new "if" statement with separate predictions would benefit the model. By using the gradient of the loss (which includes a scoring function that measures algorithm performance) it chooses the 'if' statement and which leaf it will place it on. What separates XGBoost from other gradient boosting techniques is the use of a second-order approximation of the scoring function. This approximation allows XGBoost to calculate the optimal "if" condition and its impact on performance and store it in its memory in order to not recompute it.

We will look into the math behind XGBoost [33]. The objective loss and regularization function that we need to minimize is shown in equation 2.1.

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{2.1}$$

Where y_i is the label or target value, $\hat{y}_i^t$ is the prediction we make of the i-th instance at the t-th iteration , l is a differentiable convex loss function that measures the difference between the prediction and the target values, f_t is the added term to that loss function to minimize L (we greedily add the f_t that improves our model the most), Ω is a regularization term that smooths out the final learnt weights to avoid overfitting (it penalizes the complexity of the model by intuitevely selecting the simpler models).

If we take the second order Taylor approximation we arrive at equation 2.2

$$L^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{2.2}$$

Where $g_i = \partial \hat{y}^{(t-1)} l(y_i, \hat{y}_i^{(t-1)})$ is the first order gradient statistic of the loss function and $h_i = \partial^2 \hat{y}^{(t-1)} l(y_i, \hat{y}_i^{(t-1)})$ is the second order gradient statistic of the loss function.

By removing the constant terms we reach equation 2.3

$$L^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{2.3}$$

Which is a sum of simple quadratic functions of one variable and it can be minimized by using known techniques. Our next goal becomes to find a learner that minimizes the loss function at iteration t. To minimize a simple quadratic function we have the following equation 2.4

$$\begin{aligned} argmin_x : Gx + \frac{1}{2}Hx^2 = -\frac{G}{H}, H > 0 \\ min_x : Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H} \end{aligned} \tag{2.4}$$

The scoring function q is used to measure the quality of the tree structure (we can think of it like the impurity score in decision tree algorithms, except that it is derived from a wider range of objective functions). By applying equation 2.4 to our problem we calculate the optimal value of the loss function in equation 2.5.

$$L^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{2.5}$$

Where $I_j = i|q(x_i) = j$ is the instance set of leaf and T is the number of leaves on the tree.

The 'Exact Greedy Algorithm' starts with a single root (that contains all the training examples), iterates over all features and values per feature, and evaluates each possible split loss reduction: **gain = loss(father instances) - (loss(left branch)+loss(right branch))**
The gain for the best split must be positive (and greater than the min split gain parameter), otherwise we must stop growing the branch (pruning).

Chapter 3

Data

In this chapter we will be describing the data in detail, where it came from, how we extracted it and how we preprocessed in order to be able to use it in a meaningful way and obtain results, plus some issues that arise from handling it. In total the data we collected for this project were the extracted Twitter data and the cryptocurrency historical price data.

3.1 Data Collection

We collected the tweets using the Twitter API. First we gained access by using the unique credentials issued for this project and then we created a stream to extract tweets containing the keyword terms 'bitcoin', '#BTC', '#bitcoin', 'BTC' and '$BTC'. The idea is that almost all the tweets concerning Bitcoin will contain some, or all, of these terms. The hashtag sign ('#') is used in Twitter to connect the tweets to other tweets referring to the same topic, to categorize them and to make it easier to search among tweets, which proves very convenient in our case. The cashtag sign ('$') is used by investors who want to participate and access information on large companies listed on the stock exchanges. In our case it denotes the financial theme of the tweet [34] and the reasons for its usefulness resemble the ones regarding hashtags. The limitations imposed by the Twitter API denote that we collect the tweets in 450 iterations of 100 tweets each and wait for 15 minutes to renew the rate limit back to 450 after exceeding it. We store the tweets in a Pandas DataFrame to ease the process of preprocessing them at a later stage.

The code we wrote to collect the tweets ran for around 44 hours in total, with an unexpected occurrence happening (power outage) that led to us having a 1.5 days gap where we did not collect data. In general the process was to run it once a week and collect all the tweets issued in the previous week, save them in the text file, wait one week and then run the code again. In total the data that we gathered and used have a start date of Saturday 20 March 2021 18:17 and an end date of Monday 19 April 21:39 , with a total of 7.066.334 tweets (more information about each individual week lies in the comments in the code).

For the cryptocurrency historical price data we searched a lot for the most reliable website - tool from which to extract the information. We decided to use bitcoincharts.com [35], which provides us with the price of Bitcoin on different exchanges - marketplaces. We chose the marketplace called 'Kraken' because it's one of the most popular ones and the author's first choice when it comes to trading. We chose the United States Dollar as the currency with which to estimate the value of Bitcoin because it is the one that is mostly preferred in the investing circles when it comes to trading and investing in cryptocurrencies or stocks and it is widely used by everyone in the crypto-space. Even non-American crypto-enthusiasts prefer to value crypto in USD over Euros or any other fiat (government-issued) currency. The time windows we chose is 15 minutes and 1 minute, each one used for a different purpose which we will explain in detail at a later point. We loaded the raw data manually by copy-pasting them into a Microsoft Excel file. Figure 3.1 shows the price of Bitcoin in 15 minute intervals between March 20 and April 20.

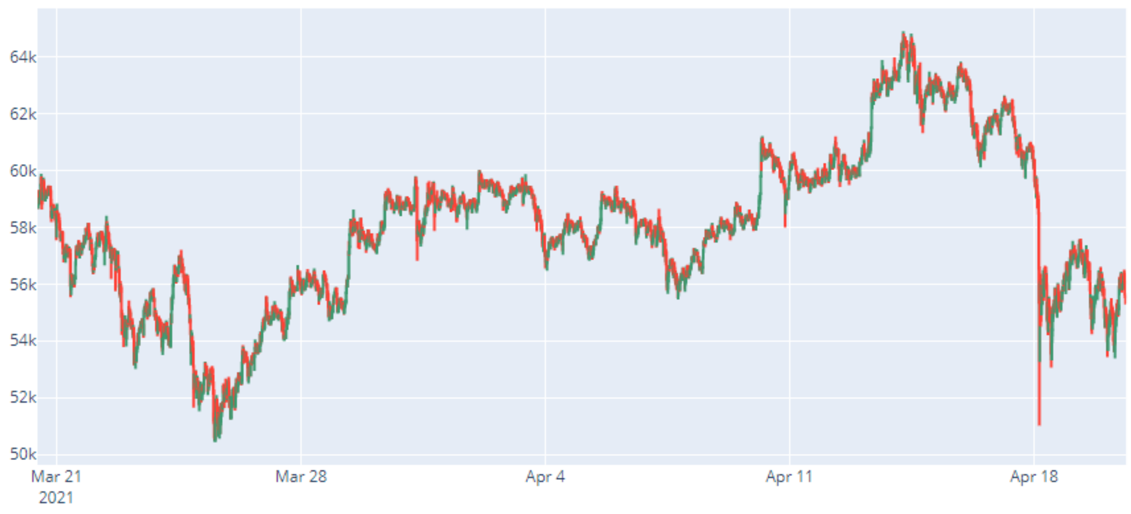


Figure 3.1: Bitcoin price (USD) March 20 - April 20 on Kraken (15 minute intervals)

3.2 Data Description

The tweets data initially contains 7 attributes (before adding more by manipulating the existing ones)

- **ID**, a fairly large integer that uniquely identifies a tweet. We could call it the key of the DataFrame.
- **Text**, a string denoting the text of the tweet. It can contain emojis, emoticons, slang , special characters and it is written using only the English language.
- **Username**, a string denoting the username of the person tweeting. It can contain emojis, emoticons, slang , special characters and it is not restricted to Latin characters.
- **UserFollowerCount**, an integer denoting the amount of followers a user has.
- **RetweetCount**, an integer denoting how many times the specific tweet was retweeted. Retweeting is the process of re-posting a tweet, sharing it with all the followers of the user.
- **Likes**, an integer denoting the number of likes a tweet received. Users like tweets that they find interesting or in accordance with what they think and like.
- **CreatedAt**, a string denoting the date and time of the creation of the specific tweet in UTC. We can call it the timestamp of the tweet.

In our project we make use of the ID field to delete the duplicate tweets and in relation to some other attributes to join datasets (the detailed processes will be explained at a later point). The ’Text’ field might be the most important attribute we have and it is widely used after a certain manipulation. The ’Username’ field is only used in one graph and to join datasets in relation to some other attributes. The ’UserFollowerCount’ field is used in a graph and in the equation that calculates the sentiment score of each tweet. The ’RetweetCount’ and ’Likes’ fields are used in the equation that calculates the sentiment score of each tweet. The ’CreatedAt’ field is used in graphs and in many parts of the project to indicate the exact moment the tweet was created.

The cryptocurrency historical price data contains 8 attributes.

- **Timestamp**, a DateTime object denoting the date and time. It covers the same dates more or less like the dates of tweet collections. Any mismatches in the dates between the two datasets get eradicated through the code (we will explain how at a later point).
- **Open**, a float denoting the opening price in USD of the cryptocurrency for the specific time window on the Kraken platform.
- **High**, a float denoting the highest price in USD of the cryptocurrency in that specific time window on the Kraken platform.
- **Low**, a float denoting the lowest price in USD of the cryptocurrency in that specific time window on the Kraken platform.
- **Close**, a float denoting the closing price in USD of the cryptocurrency for the specific time window on the Kraken platform.
- **Volume (BTC)**, a float denoting the volume of BTC traded at that specific time window (measured in Bitcoins)
- **Weighted Price**, a float denoting the Volume Weighted Average Price of BTC. This is obtained by taking the sum of all the prices, multiplied by the volume of the trade and then divided by the entire volume [36].

In our project we make use of the 'Close' column values for the prediction and the graphs and the 'Timestamp' column values to denote the date and time of the observation.

3.3 Data Privacy Concerns

The main concern when it comes to working with data that is extracted from the internet (Twitter) and used by a third-party as a means to achieve a goal (in our case the prediction) is the protection of the privacy of the users tweeting. It is actually stated in the terms of agreement that users accept when they create a Twitter account that by compiling and sending the tweets through Twitter, the author makes the information publicly available to anyone who seeks it. Tweets are immediately viewable and searchable by anyone around the world [37]. The only non-public way to communicate on Twitter is through private messages and protected tweets. What the users sometimes do not have in mind is that their data could be used in a project like this one. Keeping that in mind we pay close attention to not publicize this kind of data online and to try to hide the usernames and any other personal information as best as we can.

3.4 Data Cleaning

In order to have a full image of the dataset and to be able to treat it as a whole during the preprocessing procedures, the first thing that is needed is to merge the different parts into one. As we mentioned, while collecting the tweets we had to separate them into different files in order for the data to be collected in a correct and precise way. A total of 10 files are merged together, after dealing with some problems like the encoding not allowing one file's contents to be displayed correctly. After experimenting we deduce that the emojis being displayed correctly plays a difference in the sentiment score of the text, so we pay extra attention to display them in the proper way.

The file that contains the merged components now contains duplicate values. So our next move is to remove those duplicates. The 7 million rows of the file make this procedure rather demanding in terms of computing power. We tried running it on vanilla Jupyter Notebook, which means the Jupyter Notebook that is just running on the local PC without any outside help. The crashing of the RAM caused the program to stop running so we decided to switch to Google Colab for this task. Google Colab offers us tools to help combat the problem encountered, mainly the back-end Python 3 Google Compute engine with almost 13 GB of RAM and 108 GB of disk space available. We removed the duplicates by using the distinctive identifier of the tuples, the attribute 'ID', which as we mentioned is unique to each tuple except for the duplicate ones. In the end we are left with around 7 million rows, out of the 9 million ones that existed after the merging and we are ready to visualise the data and move on to the preprocessing procedures.

3.5 Graphs

In order to show in a graph the number of tweets collected per day, we convert the ’CreatedAt’ column from string to DateTime object and we are left with Figure 3.2. The mean daily volume of tweets is around 220.000 , with a standard deviation of around 75.000.

We again use Google Colab to ease our computation efforts. We calculate that we have around 1.1 million unique users in the dataset tweeting around 2.7 million unique tweets. That shows that around 4 out of 7 tweets in the dataset are retweets.

Figure 3.3 shows that around 600.000 users tweeted just once and most of the users tweeted less than 5 times about Bitcoin over the course of 1 month.

For our next graphs we wanted to make Wordclouds of the text found in the tweets of the full and of the non-botted dataset. Wordclouds are graphically represented collections of words. The words that form the cluster differ in size based on the frequency that the word appeared in the text. We wanted to make a Wordcloud using all the text from the full dataset, but we ran into a memory problem (RAM crashes, even the one from Google Colab). So we chose only part of the dataset (1.8 million tweets) to run the process. The Wordcloud for the full dataset is shown in Figure 3.4. We notice that words like ’giveaway’, ’https’ (denoting links), ’retweet’, ’follow’ and ’Bitcoin’ are prevalent in the text, confirming the fact that there are lots of botted texts in the dataset producing duplicate texts. The WordCloud for the non-botted dataset is shown in Figure 3.5 and words like ’ecological disaster’, ’difficult’, ’RT’ and ’Bitcoin’ are prevalent.

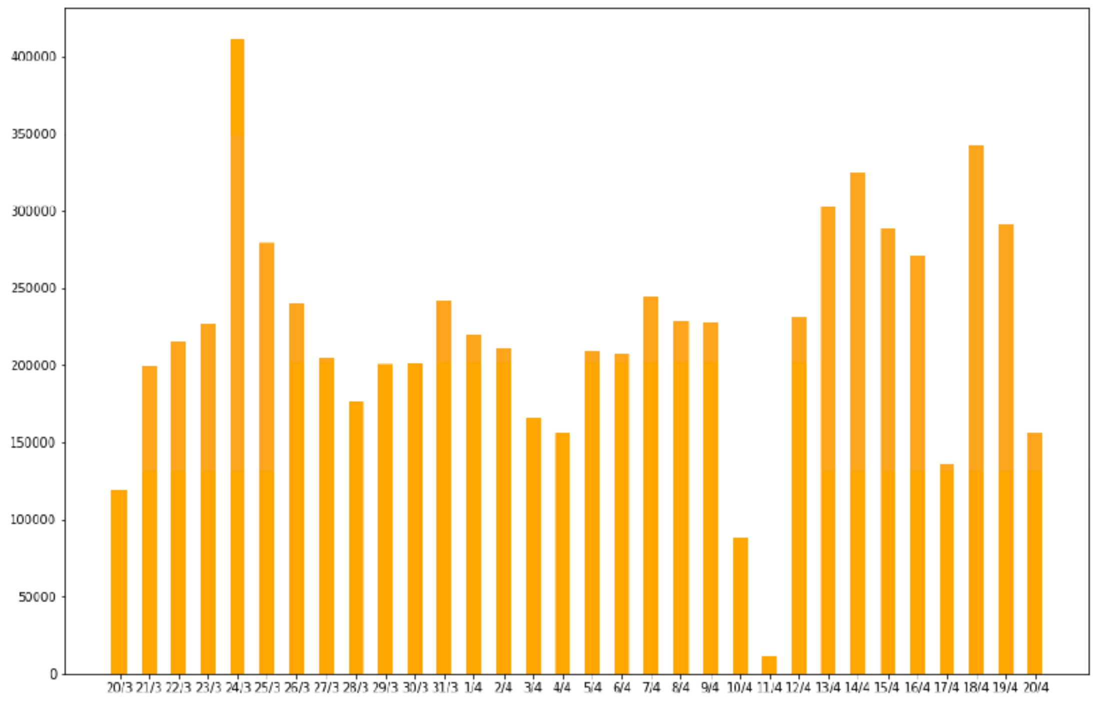


Figure 3.2: Daily Bitcoin related tweet volume

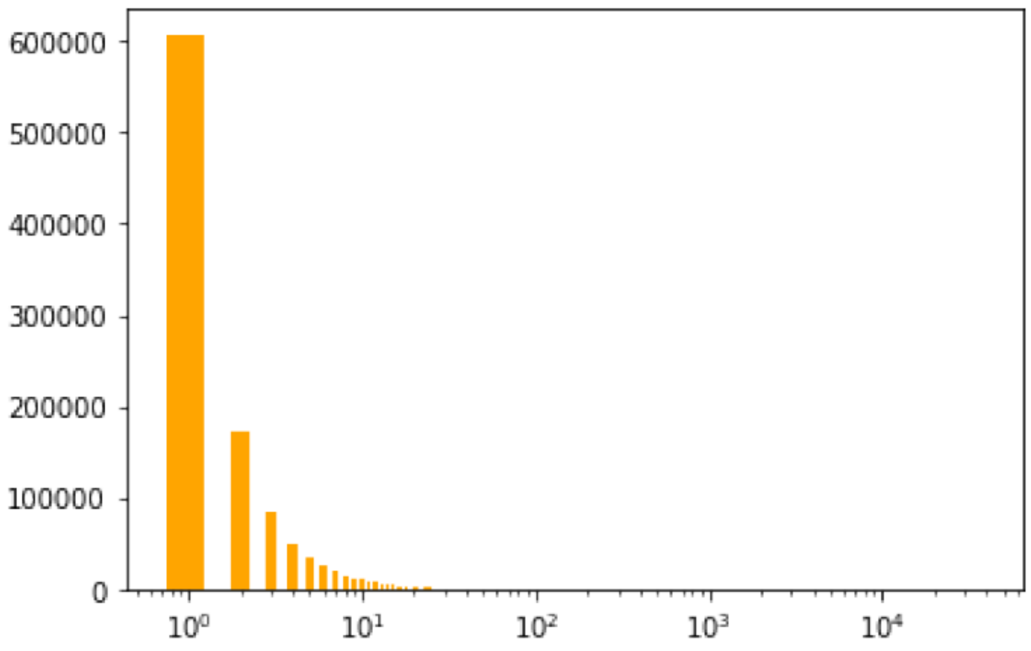


Figure 3.3: Amount of Bitcoin related tweets per user

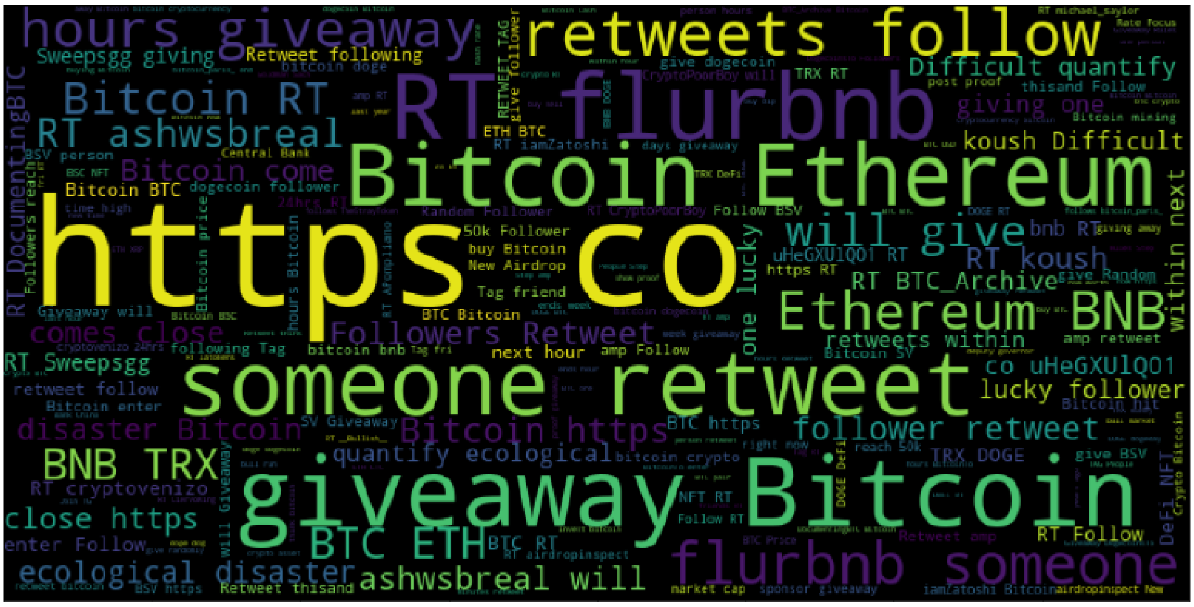


Figure 3.4: WordCloud of the full dataset

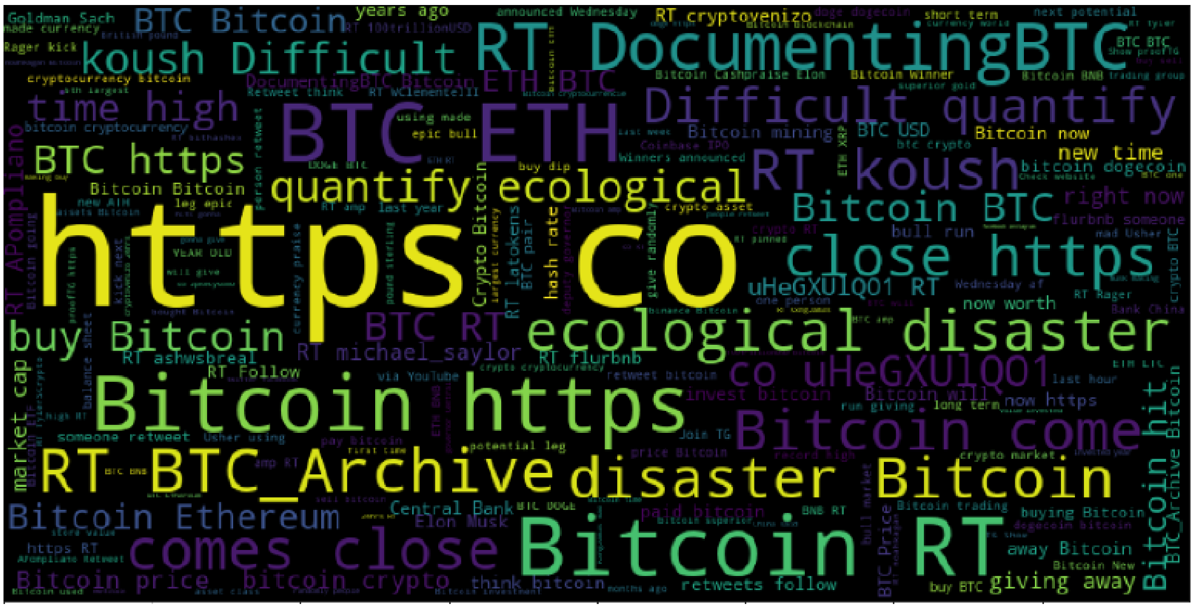


Figure 3.5: WordCloud of the non-botted dataset

Chapter 4

Methods

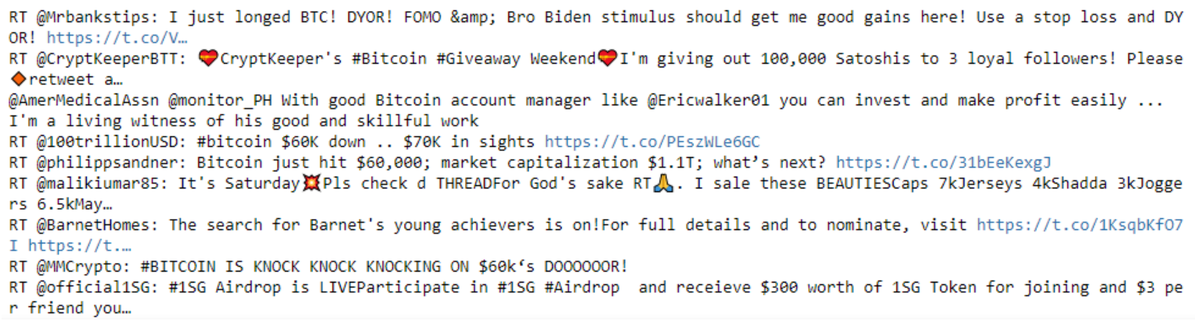
In this chapter we will be describing the methods used in order to produce the results. To sum it up the methods discussed are Bot Detection, Data Preprocessing, Sentiment Analysis, Correlation Analysis and Prediction.

4.1 Bot Detection

Bots are automated accounts that post content or interact with other users with no direct human interference. As far as Twitter bots are concerned, their main purpose is to tweet and retweet content on a large scale with a specific goal in mind. That goal could be to sway the public opinion towards a side, to advertise a product or service, to place the spotlight on an individual and many other shady and non-shady tactics and businesses.

Bots have had an impact in a number of social and political events. As for the United States presidential elections of 2016, there have been claims that at least 30% of the Twitter followers of both presidential candidates were bots [38]. During TV debates for these aforementioned elections, 3.8 million tweets, equal to one fifth of all tweets on the topic, were posted by 400.000 automated accounts. As for the Brexit debate, bots were active on Twitter and Facebook supporting both sides, but swaying the public opinion more in favour of the Brexit. Researchers at Carnegie Mellon noted that out of 200 million tweets concerning the Coronavirus, 45% of them were made by botted accounts [39].

As we can see, the prevalence of botted accounts on Twitter is a known fact. During our extraction of tweets from the stream we noticed an abundance of tweets that were coming from accounts with 0 followers and with the text of these tweets looking suspiciously automatically generated (usually containing links, a lot of emojis and hashtags and not a lot of sense coming out of the message). Figure 4.1 showcases some examples. The main behavior of cryptocurrency related bots is stating to give away free cryptocurrency, posting links to other bot accounts, impersonating someone famous or calling another user or post a scam [40].



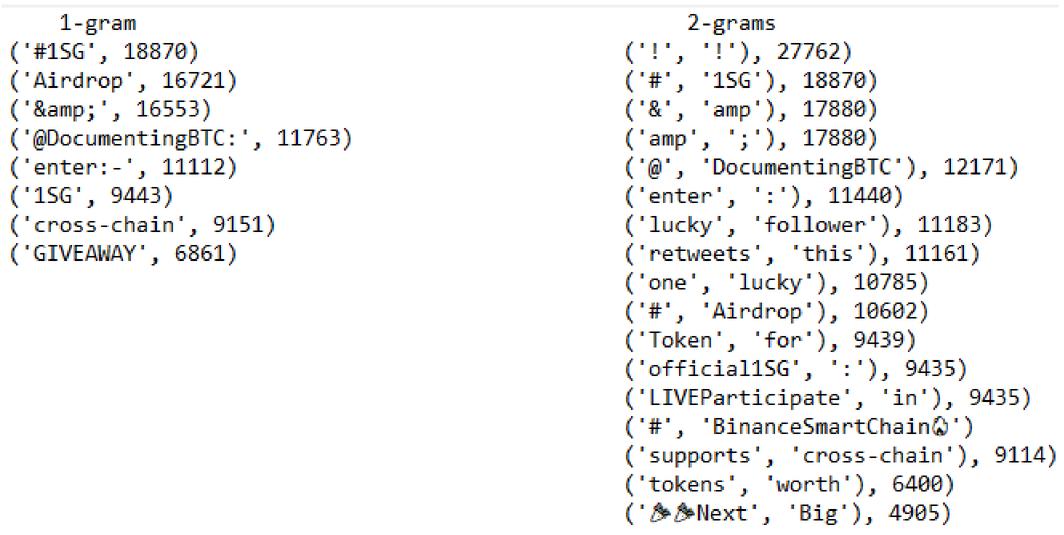
```
RT @Mrbankstips: I just longed BTC! DYOR! FOMO &amp; Bro Biden stimulus should get me good gains here! Use a stop loss and DY
OR! https://t.co/V…
RT @CryptKeeperBTT: ❤️CryptKeeper's #Bitcoin #Giveaway Weekend❤️I'm giving out 100,000 Satoshis to 3 loyal followers! Please
🔶retweet a…
@AmerMedicalAssn @monitor_PH With good Bitcoin account manager like @Ericwalker01 you can invest and make profit easily ...
I'm a living witness of his good and skillful work
RT @100trillionUSD: #bitcoin $60K down .. $70K in sights https://t.co/PEszWLe6GC
RT @philippsandner: Bitcoin just hit $60,000; market capitalization $1.1T; what's next? https://t.co/31bEeKexgJ
RT @malikiumar85: It's Saturday💥Pls check d THREADFor God's sake RT🙏. I sale these BEAUTIESCaps 7kJerseys 4kShadda 3kJogge
rs 6.5kMay…
RT @BarnetHomes: The search for Barnet's young achievers is on!For full details and to nominate, visit https://t.co/1KsqbKfO7
I https://t.…
RT @MMCrypto: #BITCOIN IS KNOCK KNOCK KNOCKING ON $60k's DOOOOOOR!
RT @official1SG: #1SG Airdrop is LIVEParticipate in #1SG #Airdrop  and receieve $300 worth of 1SG Token for joining and $3 pe
r friend you…
```

Figure 4.1: Text from tweets of accounts that resemble bot behavior

The problem with botted accounts is that their messages often carry very little sentiment and contribute nothing but noise to the overall dataset. It is uncertain whether those message influence the overall prices of the cryptocurrency. So we decided to remove any row that contained a bot-like message , run the preprocessing, sentiment analysis and prediction on the non-botted and the full dataset and compare results to see if we notice any change in accuracy. The method for detecting bots is a filter that was inspired by the works of Evita Stenqvist et al [10] and Olivier Kraaijeveld et al [11].

First of all we choose a subset of 500.000 tweets to construct the first part of the filter. We again use Google Colab to run the code for the same reason mentioned above. Out of those 500.000 tweets, we keep the text that is duplicated among them. If many different users tweet out the exact same message, it might be a case of retweeting, but it may also be a case of spam messages coming from bots. Almost half of the 500.000 tweets are duplicated ones. The duplicated text is then run through a function to find out the most common words that are present in the text. After all the stopwords were removed (words like 'the', 'I', 'is' that hold no sentiment and are of no value to our bot removal program) we are left with the most common single words that are found in the text. By performing a bigram check we find out the most common bigrams (think of it like 2 words together, e.g. 'lucky'-'follower' appearing around 11.000 times) and the most common trigrams (three words together, e.g. 'what'-'a'-'coincidence' appearing almost 15.000 times) that appear in the duplicated set.

By manual scrutiny and evaluation we decide on the suspicious n-grams that we will manually investigate (Figure 4.2). It turns out that a lot of tweets containing these words and bigrams can be considered of automated origin.



```
    1-gram
('#1SG', 18870)
('Airdrop', 16721)
('&amp;', 16553)
('@DocumentingBTC:', 11763)
('enter:-', 11112)
('1SG', 9443)
('cross-chain', 9151)
('GIVEAWAY', 6861)
```

```
      2-grams
('!', '!'), 27762)
('#', '1SG'), 18870)
('&', 'amp'), 17880)
('amp', ';'), 17880)
('@', 'DocumentingBTC'), 12171)
('enter', ':'), 11440)
('lucky', 'follower'), 11183)
('retweets', 'this'), 11161)
('one', 'lucky'), 10785)
('#', 'Airdrop'), 10602)
('Token', 'for'), 9439)
('official1SG', ':'), 9435)
('LIVEParticipate', 'in'), 9435)
('#', 'BinanceSmartChain🚀')
('supports', 'cross-chain'), 9114)
('tokens', 'worth'), 6400)
('🎉🎉Next', 'Big'), 4905)
```

Figure 4.2: Suspicious N-grams

Next up we load the full dataset of 7 million rows in order to remove the tweets posted by bots. Based on the list we compiled , we remove tweets that contain words that are included in the list. Another method used in the filter is to calculate the mean and standard deviation of hashtags in the tweet text of the full dataset. By adding the mean plus two times the standard deviation of the hashtag counts of each tweet text, we arrive at the number 8. If the tweet text contains more than 8 hashtags, it is considered to fulfil a condition that in combination with another condition it will classify the text as botted. This heuristic was based on this work [11] and on the valid assumption that botted messages usually contain a large number of hashtags to become more visible and appear on a lot of searches. The same is done for cashtags, with the 'magical' number being 6.

For the final part of the filter, we create some conditions to be satisfied in order to classify the text as botted. The first condition is whether the text contains the words 'giveaway' or 'giving away'. In the second there are the words 'pump' and 'register' appearing together or 'join' appearing alone. The third and fourth conditions are the hashtag and cashtag counts that were mentioned above.

So in the end, to consider a tweet as botted it has to satisfy any 2 of those 4 conditions or contain the words included in the list. This constitutes the entirety of our filter, and after using it a total of around 5.85 million tweets are left in the non-botted dataset, out of the 7 million that is the full dataset, 82% of the original.

4.2 Dataset Preprocessing

Now that we have our full dataset containing bots and our non-botted dataset, we proceed to preprocess both of them in order to perform our sentiment analysis and our prediction.

Our first thought was to remove every word that does not exist in the NLTK's (Natural Language Toolkit) words corpus [41] from the dataset. But after some thinking we realised that words-acronyms like 'FOMO' (Fear Of Missing Out), 'HODL' (Hold - meaning do not sell) do not exist in the NLTK's words corpus so they would be removed. Removing them would be a mistake because they hold sentiment value in the context of our project, so we decide against it. The list of cryptocurrency related terminology that we compiled will be shown in the sentiment analysis section. For now we just confirmed that those words are included in our extracted text.

After a lot of research, we decided that there is no need to casefold the texts (from capital to small letters), to remove any punctuation or to remove stopwords. Vader considers capital letters and punctuation important for the assignment of sentiment values, and removes stopwords on its own. We also do not apply Lemmatisation (the practice of grouping together the inflected forms of a word so they can be analysed as a single word, e.g. running, runs, ran to run). The explanation behind that decision is that VADER has different ratings depending on the form of the word and therefore the input should not be stemmed or lemmatised [42]. Overall the incredible thing about Vader is that it does not need a great deal of preprocessing to work, tokenization and lemmatisation are unnecessary and on top of that Vader can understand emojis and extended punctuation among other things. Despite all that, we decide to perform a great deal of preprocessing steps to ensure the validity and high accuracy of our results.

The steps of preprocessing are as follows:

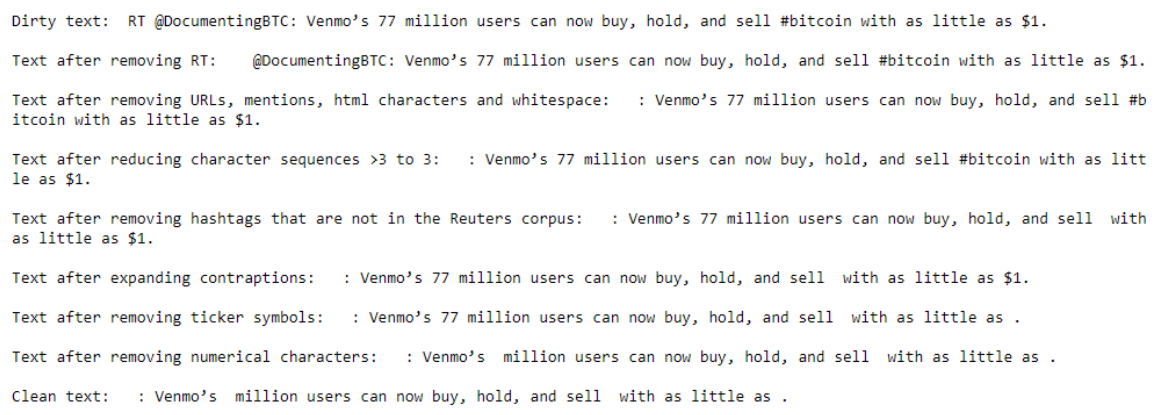
- We use a contractions list derived from here [43] to **expand contractions** (e.g. "could've" to "could have"). We do that because Vader needs to assign sentiment values and the contracted words do not change the overall value of the sentence.

- We **reduce consecutive characters** to 3 when they are more than 3 (e.g. 'Helloooooo' to 'Hellooo').

- We **remove the 'RT'** at the start of texts (it denotes that the tweet is actually a retweet, so these two letters add no sentiment value to the overall text).
- We **remove http links** as they add no sentiment value and also might be leading to fishy websites.
- We **remove www links** for the same reason as above.
- We **remove excess whitespace**, blanks in the text that make it longer without contributing anything.
- We **remove @ mentions**, which are mentions of other users as part of a retweet or just to mention or call someone out. We choose to remove them because they do not contribute to our goal and also pose a privacy problem as mentioned in the 'Data Privacy Concerns' paragraph.
- We **remove HTML characters**, the & symbol with letters following it since it holds no sentiment value.
- We **remove tweets with less than 4 tokens**. Tweets that contain 3 or less words are not considered meaningful to keep.
- We **remove hashtags** by using a special method. If the hash-tagged word exists in the NLTK words corpus then we remove the hashtag sign but keep the word. If it does not exist then we remove the hash-tagged word completely. This happens because we might have for example a tweet text containing the tokens '#awesome', '#BTC', '#crypto'. The last 2 tokens are considered irrelevant for our sentiment analysis so they can be removed completely. Meanwhile, the first token is really important and we decide to keep it and make use of it by removing the hashtag sign in front of it.
- We **remove ticker symbols**. The $ sign is the financial equivalent of the hashtag and we remove it for the same reasons as above.
- We **remove tokens containing numerical characters** because they also add nothing to our computations.

The preprocessing code ran for 10 days straight, it reached around 60% of completion and we stopped it to save the progress and continue from where we left off. Overall it took around 400 hours to finish, dealing with all the 7 million rows. We show an example of the steps taken for the preprocessing in Figure 4.3.

For the non-botted dataset we thought about following the same procedure, but given how time consuming it was we decided against it and tried to think of a more clever way. We already have the preprocessed text from the full dataset, and the non-botted dataset is just a subset of the original. So if we merged the two datasets we would have the non-botted dataset with the preprocessed text, in a much faster way. Merging based on the column 'ID' would be ideal, since it is considered the key attribute of both datasets. However, for some unknown reason it is not displayed correctly in the full dataset, so we had to find another way. We merge the datasets based on the combination of the attribute 'CreatedAt' and the attribute 'Username'. This combination proves unique and our job is done in a correct and efficient way.



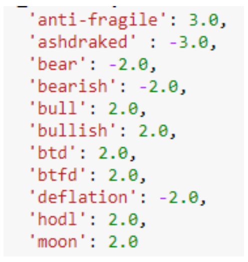
```
Dirty text:  RT @DocumentingBTC: Venmo’s 77 million users can now buy, hold, and sell #bitcoin with as little as $1.

Text after removing RT:    @DocumentingBTC: Venmo’s 77 million users can now buy, hold, and sell #bitcoin with as little as $1.

Text after removing URLs, mentions, html characters and whitespace:   : Venmo’s 77 million users can now buy, hold, and sell #b
itcoin with as little as $1.

Text after reducing character sequences >3 to 3:   : Venmo’s 77 million users can now buy, hold, and sell #bitcoin with as litt
le as $1.

Text after removing hashtags that are not in the Reuters corpus:   : Venmo’s 77 million users can now buy, hold, and sell  with
as little as $1.

Text after expanding contraptions:   : Venmo’s 77 million users can now buy, hold, and sell  with as little as $1.

Text after removing ticker symbols:   : Venmo’s 77 million users can now buy, hold, and sell  with as little as .

Text after removing numerical characters:   : Venmo’s  million users can now buy, hold, and sell  with as little as .

Clean text:   : Venmo’s  million users can now buy, hold, and sell  with as little as .
```

Figure 4.3: Preprocessing steps

4.3 Sentiment Analysis

In order to perform sentiment analysis, we load the full preprocessed dataset. As mentioned before, we compiled a list of cryptocurrency specific terminology (otherwise called jargon or slang) from multiple sources [44, 45]. Out of all those words we chose the eleven words that can be assigned with positive or negative sentiment, and we heuristically gave them values (Figure 4.4).

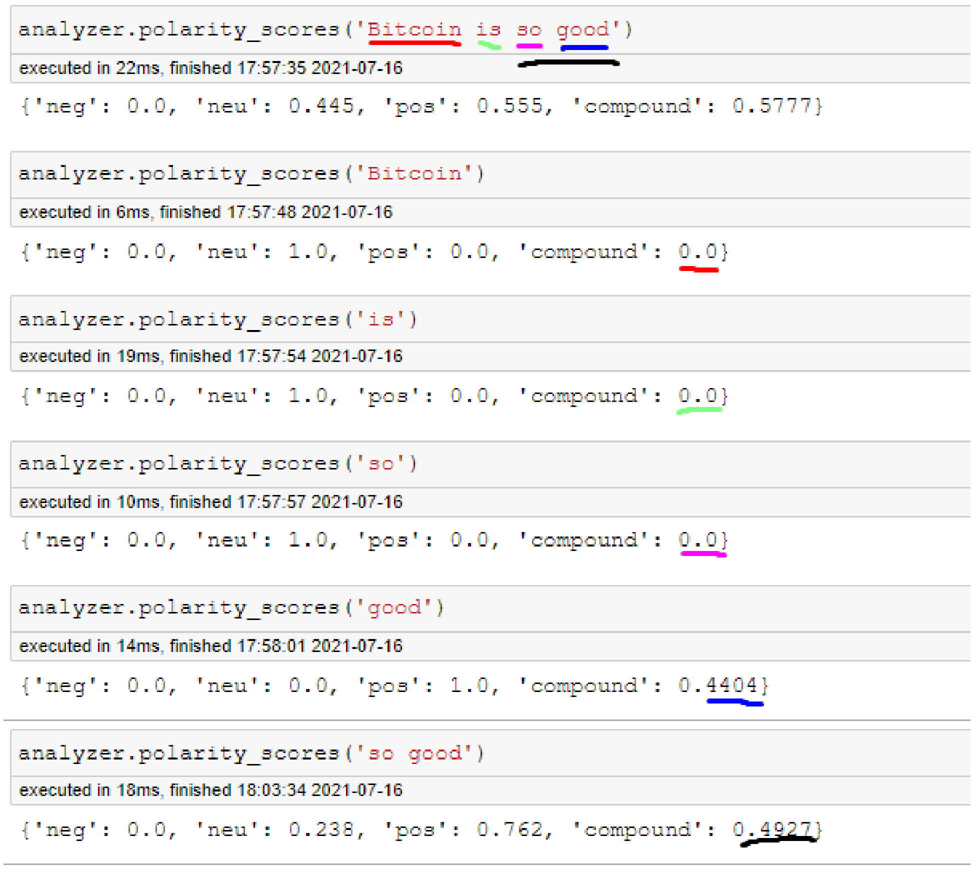
The assignment of values is arbitrarily done based on the feeling that each work provokes. For example, when Bitcoin is described as 'bearish', this is a pessimistic prediction saying that prices will drop, so we assign a Vader score of -2 to the word. When it is described as 'bullish', that is an optimistic approach that predicts that prices will rise, so we assign +2 to the word. We add these words and their corresponding values to the VADER lexicon.



```
'anti-fragile': 3.0,
'ashdraked' : -3.0,
'bear': -2.0,
'bearish': -2.0,
'bull': 2.0,
'bullish': 2.0,
'btd': 2.0,
'btfd': 2.0,
'deflation': -2.0,
'hodl': 2.0,
'moon': 2.0
```

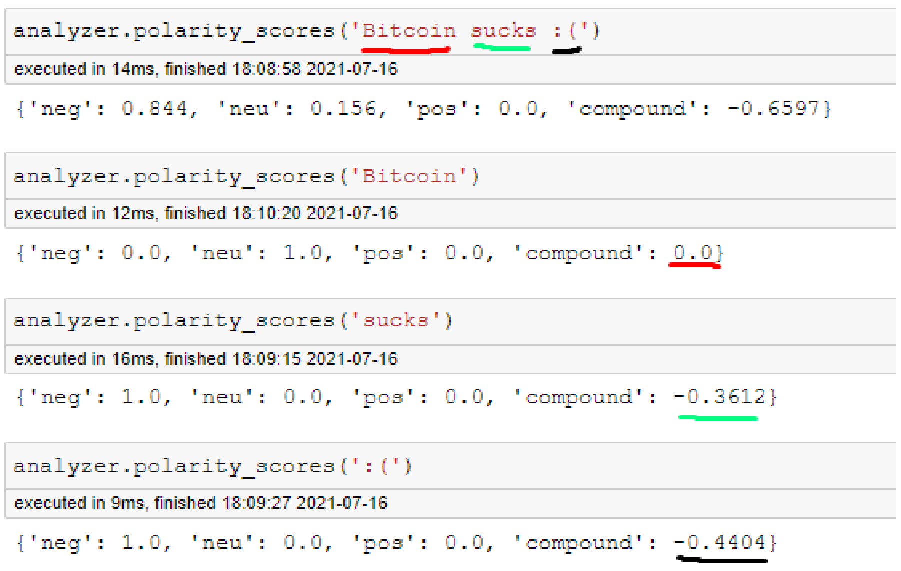
Figure 4.4: Cryptocurrency slang Vader scores

Through VADER's functions we calculate the compound score of each tweet text. Every sentence is split into words and every individual word is assigned a sentiment score (the words that already exist in VADER's lexicon, plus the ones we added ourselves). The total compound score of each tweet text is calculated by adding the individual scores of each word, with some weights attached to them. We show an example of positive sentiment compound score measurement in Figure 4.5 and of negative sentiment compound score in Figure 4.6.



```
analyzer.polarity_scores('Bitcoin is so good')
executed in 22ms, finished 17:57:35 2021-07-16
{'neg': 0.0, 'neu': 0.445, 'pos': 0.555, 'compound': 0.5777}

analyzer.polarity_scores('Bitcoin')
executed in 6ms, finished 17:57:48 2021-07-16
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

analyzer.polarity_scores('is')
executed in 19ms, finished 17:57:54 2021-07-16
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

analyzer.polarity_scores('so')
executed in 10ms, finished 17:57:57 2021-07-16
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

analyzer.polarity_scores('good')
executed in 14ms, finished 17:58:01 2021-07-16
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.4404}

analyzer.polarity_scores('so good')
executed in 18ms, finished 18:03:34 2021-07-16
{'neg': 0.0, 'neu': 0.238, 'pos': 0.762, 'compound': 0.4927}
```

Figure 4.5: Positive sentiment compound score



```
analyzer.polarity_scores('Bitcoin sucks :(')
executed in 14ms, finished 18:08:58 2021-07-16
{'neg': 0.844, 'neu': 0.156, 'pos': 0.0, 'compound': -0.6597}

analyzer.polarity_scores('Bitcoin')
executed in 12ms, finished 18:10:20 2021-07-16
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

analyzer.polarity_scores('sucks')
executed in 16ms, finished 18:09:15 2021-07-16
{'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.3612}

analyzer.polarity_scores(':(')
executed in 9ms, finished 18:09:27 2021-07-16
{'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.4404}
```

Figure 4.6: Negative Sentiment compound score

The metric used in our model is not the compound score. Instead we calculate the sentiment score, based on the compound score we measured, getting inspired by Mohapatra et al[7] who thought of a very clever equation. The idea is that some users are more influential than others, either because they possess a lot of followers or because their specific tweet received a lot of likes and retweets. In that case, their tweet should be assigned a bigger value than the tweets that received no attention at all. That is because a tweet concerning Bitcoin that receives attention through followers, likes and retweets, will be more likely to affect the price. The final score is calculated through equation 4.1:

$$SentimentScore = CompoundScore * UserFollowerCount * (Likes + 1) * (RetweetCount + 1) \tag{4.1}$$

The +1 added on the likes and retweets count serves as a way to not have the total sentiment score go to 0 when the likes or the retweets are 0 (a common occurrence in our dataset). We avoid adding +1 to the follower count as another way to get rid of botted accounts. A real human user is rarely going to have no followers, whereas for bots it is very common. The same process is followed for the non-botted dataset and we are left with the sentiment scores of each tuple in our datasets.

4.4 Sentiment Analysis Graphs

On Figure 4.7 we notice what percentage of tweets are strongly, mildly or weakly positive, negative or neutral. Overall around 50% of tweets are on the positive side, 38% on the neutral side and 12% on the negative side.

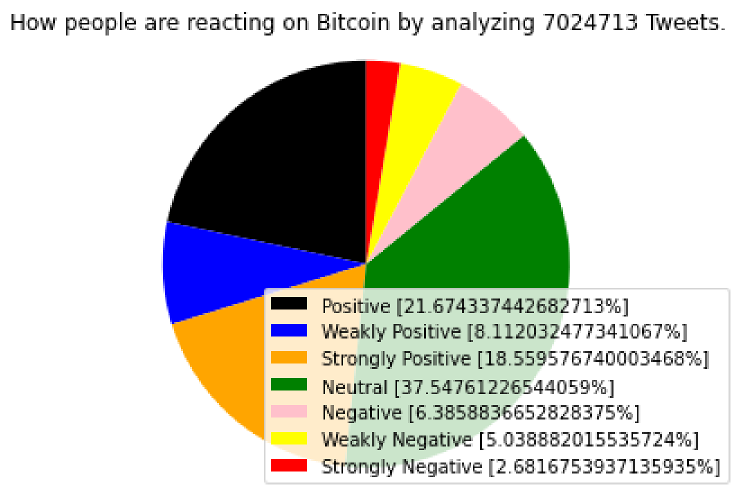


Figure 4.7: Negative Sentiment compound score

For the next two graphs we chose our favourite, WordClouds. We choose all the tweet text from tuples that hold a sentiment score of above 0 and we construct the WordCloud shown in Figure 4.8. We notice that words like ’lucky’, ’giving’, ’reward’ and ’invest’ are shown. Correspondingly, by choosing the tweets with a negative sentiment score we arrive at Figure 4.9. We notice that words like ’disaster’, ’ecological disaster’, ”don’t”, ’think’ are shown.

In order to create graphs with the hourly summed-up sentiment score we round every tuple’s ’CreatedAt’ column to the nearest hour. So for example, if the time that the tweet was created is 15:23:12, the result that will be inserted in the new column with the rounded-up times will be 15:00. This happens so that we can add up all the sentiment scores from the same time-frame and create a list with per hour summed-up sentiment scores. We load the hourly price of Bitcoin for the same time window in order to put them on the same graph with the summed-up scores. The problem is aligning the time-windows at exactly the right time and dealing with some missing values, but with some code it is alleviated quickly and we arrive at Figure 4.10. We notice some spikes in Sentiment score corresponding (with a bit of lag) to the changes in price. This intrigues us and makes us want to explore the correlation of price and sentiment in detail, which we will do in the next section. Lastly, Figure 4.11 shows the hourly change in compound score.

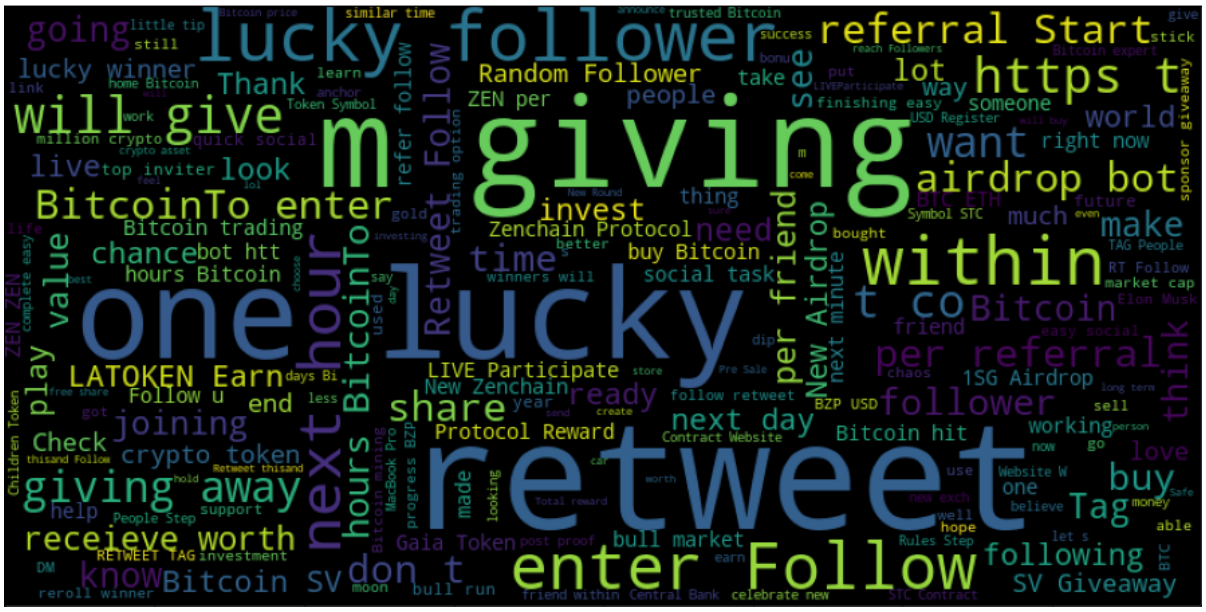


Figure 4.8: Positive sentiment WordCloud

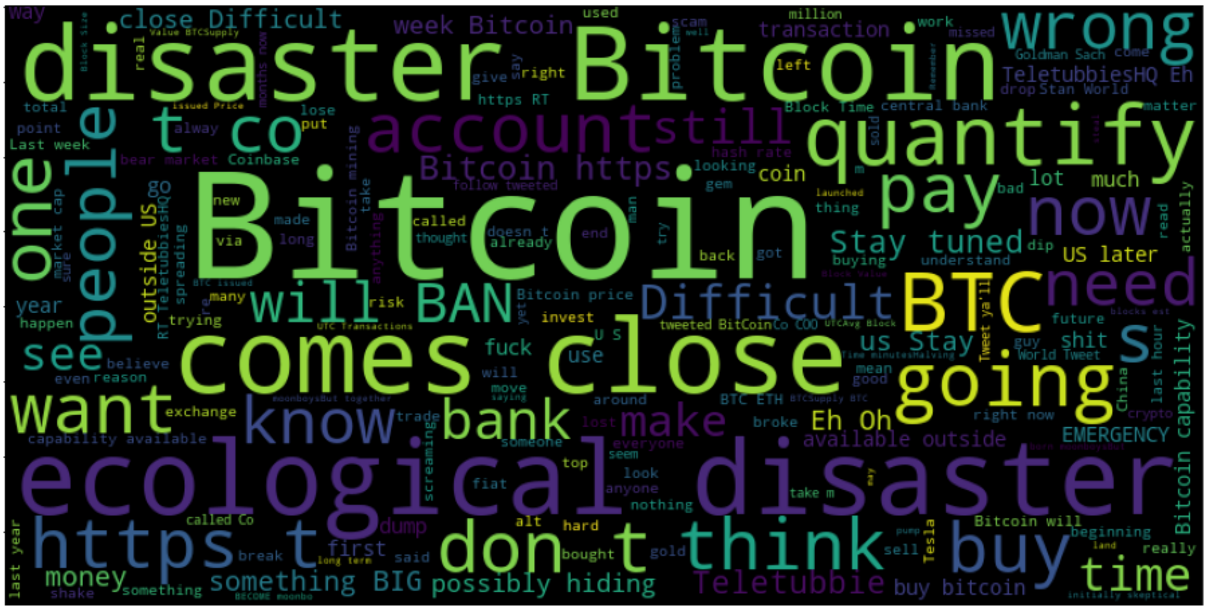


Figure 4.9: Negative sentiment WordCloud

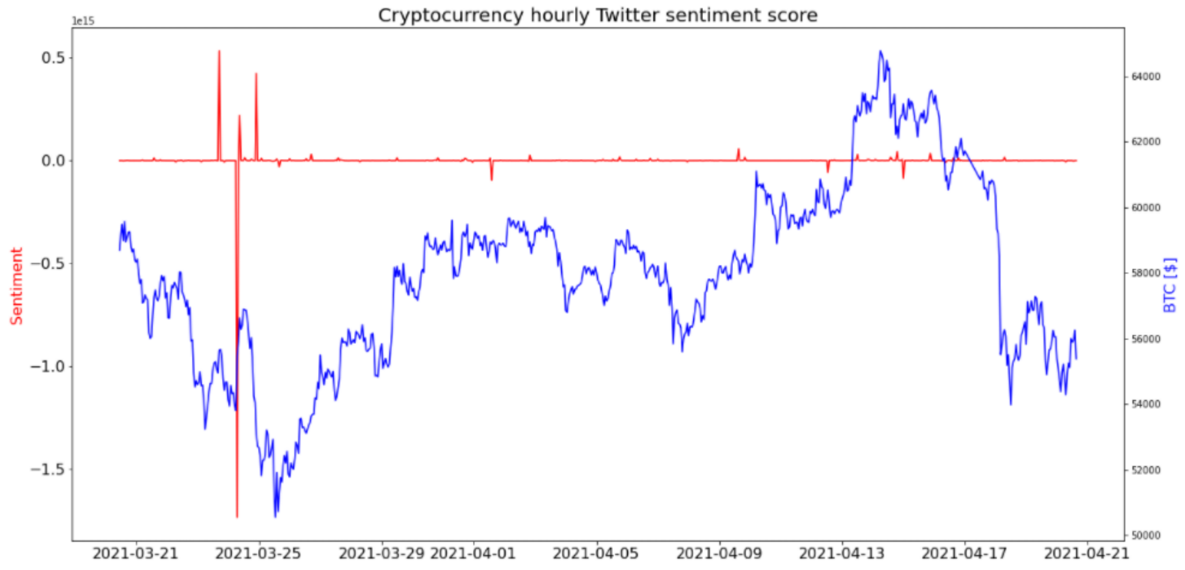


Figure 4.10: Hourly Sentiment score and Price

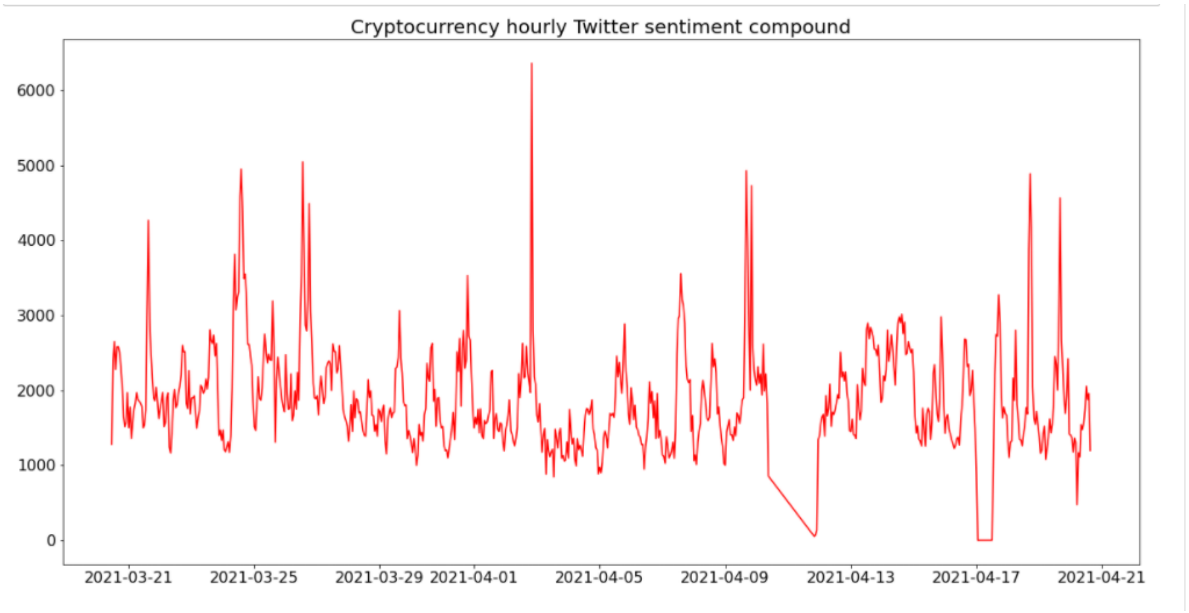


Figure 4.11: Hourly Sentiment Compound

4.5 Correlation Analysis

Our main purpose in this particular endeavor is to capture the correlation between the hourly price of Bitcoin and the hourly sum of sentiment scores. So in this direction, we load the full dataset of tweets and allign it with the price dataset.

The **Pearson** cross-correlation, denoted by ρ for a population (equation 4.2) and r for a sample (equation 4.3), is a statistical measure that shows the strength of the linear relationship between two variables [46]. It indicates how far the points of the two variables lie from the line of best fit. It takes values from -1 to +1, with 0 indicating no association between the variables, a value greater than 0 indicating positive association, from weak to strong depending on how close we are to +1 and a value less than 0 indicating negative association, again with the intensity depending on how close we are to -1. We notice in Figure 4.12α□ that the Pearson correlation for our data peaks at around 0.05, a fairly weak positive correlation. The lag axis depicts the delay in time-windows (one hour each).

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \tag{4.2}$$

Where:

X,Y = the two variables

cov = the covariance of those variables (the measure of their joint variability)

$\sigma_x \sigma_y$ = the product of the standard deviations of the variables

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 (y_i - \overline{y})^2}} \tag{4.3}$$

Where:

n = the sample size

x_i, y_i = the individual sample points of the two variables

$\overline{x}, \overline{y}$ = the sample means of those variables

The **Kendall** cross-correlation or Kendall's τ is used to test the similarities in the orderings of data when it ranked by quantities [47]. It uses pairs of observations and determines the strength of association based on the pattern of concordance (consistency , meaning that x_2 - x_1 and y_2 - y_1 have the same sign - ordered in the same way) and discordance (inconsistency - opposite sign) between the pairs. It basically calculates the dependence between ranked variables and it can be calculated for continuous as well as ordinal data. It can be described by equation 4.4. We notice in Figure 4.16β□ that the Kendall correlation for our data peaks at around 0.09.

$$\tau = \frac{c-d}{c+d} = \frac{S}{\begin{pmatrix} n \\ 2 \end{pmatrix}} = \frac{2S}{n(n-1)} \tag{4.4}$$

Where:

c = the number of concordant pairs

d = the number of discordant pairs

The **Spearman** cross-correlation (also denoted as ρ) can be interpreted as the rank-based version of the Pearson correlation and it can be used for variables that are not normal-distributed and have a non-linear relationship, which it measures using the equation 4.5. In Figure 4.12γ□ we notice that the correlation peaks at 0.12 for a lag of around 28 hours.

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.5}$$

Where:

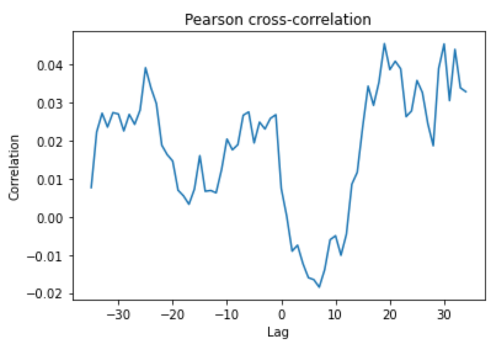
d = the pairwise distances of the ranks of the variables$x_i and y_i$

n = the number of samples

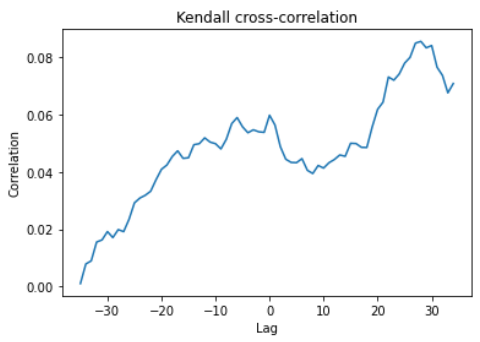
We try normalising the data (by dividing the each component of the two variables by their absolute maximum value) to see if there is any change in the correlation coefficients and, as the theory already suggested, there is indeed no change (Figure 4.13, Figure 4.14).

We take the derivatives of the score and the price to notice if there is any difference in the correlation coefficients, and indeed we notice a slight increase in the numbers (Figure 4.15,Figure 4.16α,Figure **??**,Figure 4.16γ).

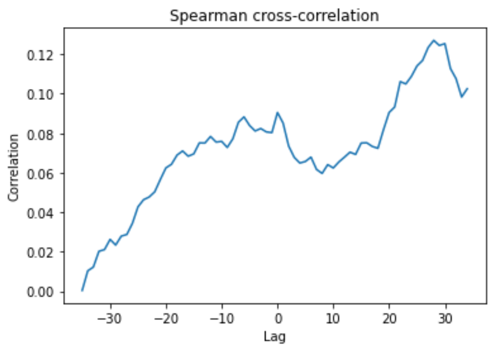
When the same procedure is carried out for the non-botted dataset we notice no difference in the results that come up (graphs are included in the code).



(α) Pearson Cross-Correlation



(β) Kendall Cross-Correlation



(γ) Spearman Cross-Correlation

Figure 4.12: Pearson, Kendall and Spearman Cross-Correlation Graphs

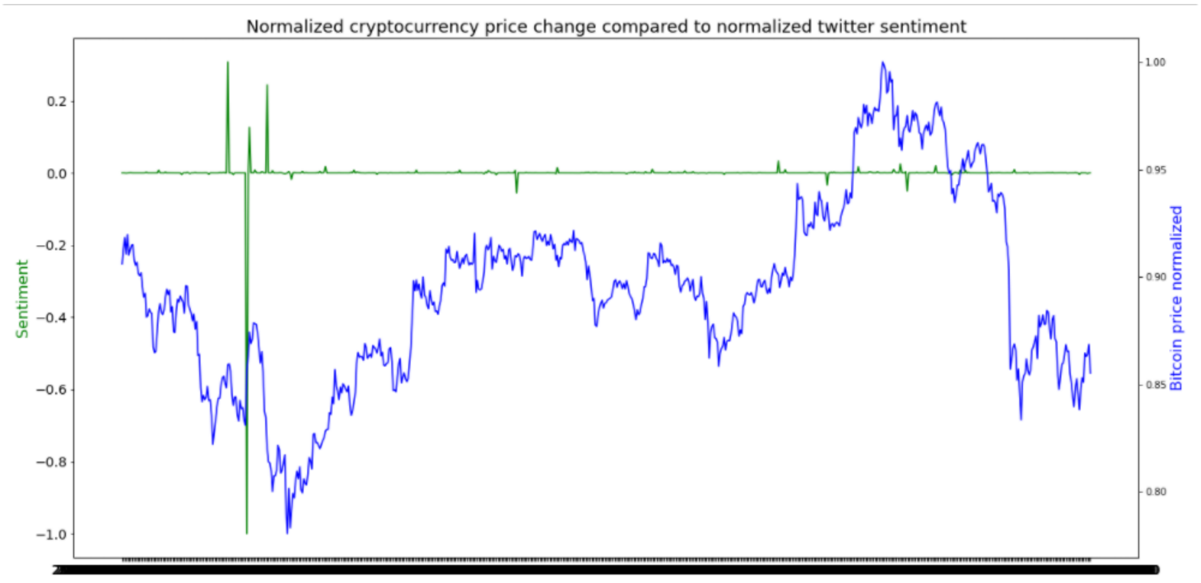
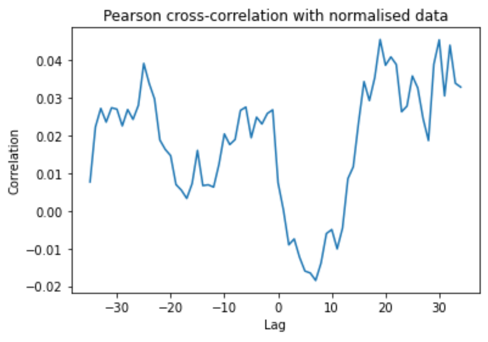
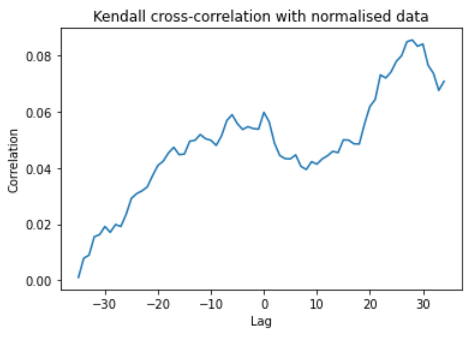


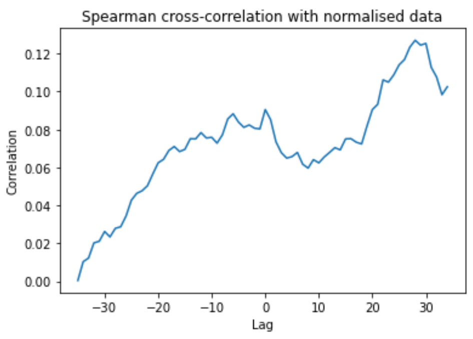
Figure 4.13: Normalised Price compared to normalised Sentiment Score



(α□) Normalised Pearson Cross-Correlation



(β□) Normalised Kendall Cross-Correlation



(γ□) Normalised Spearman Cross-Correlation

Figure 4.14: Normalised Pearson, Kendall and Spearman Cross-Correlation Graphs

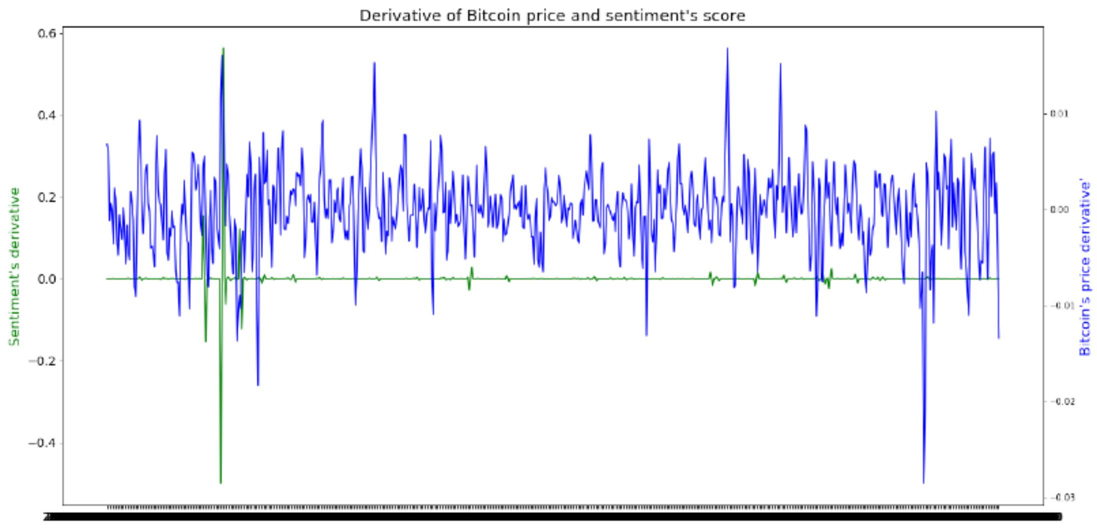
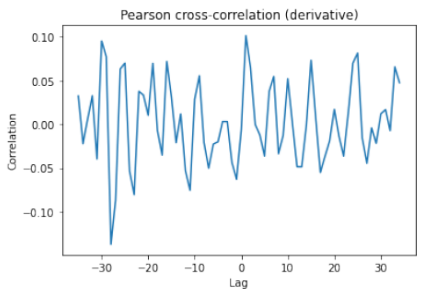
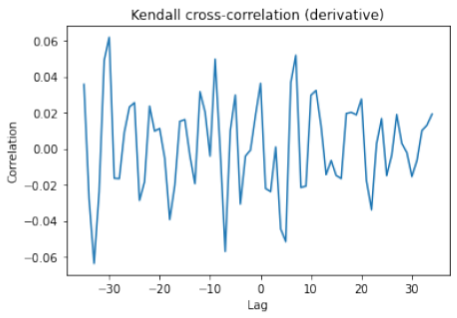


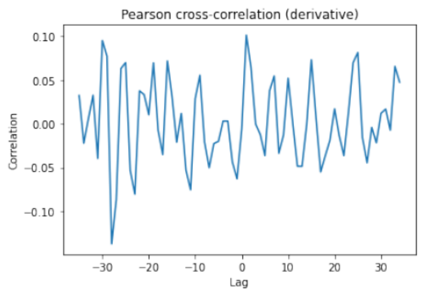
Figure 4.15: Derivative of Bitcoin's price and derivative of Sentiment score



(α□) Derivative values Pearson Cross-Correlation



(β□) Derivative values Kendall Cross-Correlation



(γ□) Derivative values Spearman Cross-Correlation

Figure 4.16: Derivative values Pearson, Kendall and Spearman Cross-Correlation Graphs

4.6 Prediction

4.6.1 Twitter Stream

In order to carry out our online predictions we first need to set up a stream on Twitter to capture incoming tweets and calculate their sentiment scores on the spot. So we set up the VADER sentiment analyzer function and by using the same formula as in equation 4.1, with one little change - we add +1 to the UserFollowerCount so that we do not ignore tweets from users with zero followers, we calculate the score of each tweet. We write the score and the timestamp in a file, from which we will read when the time to make the prediction comes. By using the API keys that we obtained for our project from our Twitter Developer account, we manage to stream and extract tweets in real time and calculate their scores. The stream produces data for prediction using both the full dataset and the non-botted one. It could go on forever but we interrupted it manually after around 16 hours, so that we can use the data gathered to check for the accuracy of the model.

4.6.2 Prediction

First we split the already preprocessed full dataset into four parts, so that we can run for every tuple in an efficient and less time-consuming manner the function that rounds up the time (in 1 minute intervals, so 15:12:21 becomes 15:12:00). Then we merge the dataset back together, now with a new attribute, the rounded time. This all happens so that we can make our prediction model have a time-frame of one minute intervals. We load the price per minute (which we extracted from here[35]) by doing a lot of copy pasting. We can see in Figure 4.17 how the price changes in time for our specific time window. What we will need for our prediction is the closing price, so that we can predict the price as soon as the one minute has passed. Also , for every individual minute, we need to sum up the sentiment scores that arrive from the tweets that were extracted through the Twitter stream. The problem that we encountered was that the price per minute list has rows for every minute, while in the full dataset of tweets gathered for some minutes we do not have sentiment scores (because either we were not gathering data in that exact minute due to system failure or there were just no tweets at that minute concerning Bitcoin). We solve that problem by removing the rows that do not exist in both lists. Also, in the price lists there are some NaN values, so we need to remove them and also remove the corresponding elements in the sentiment score list.

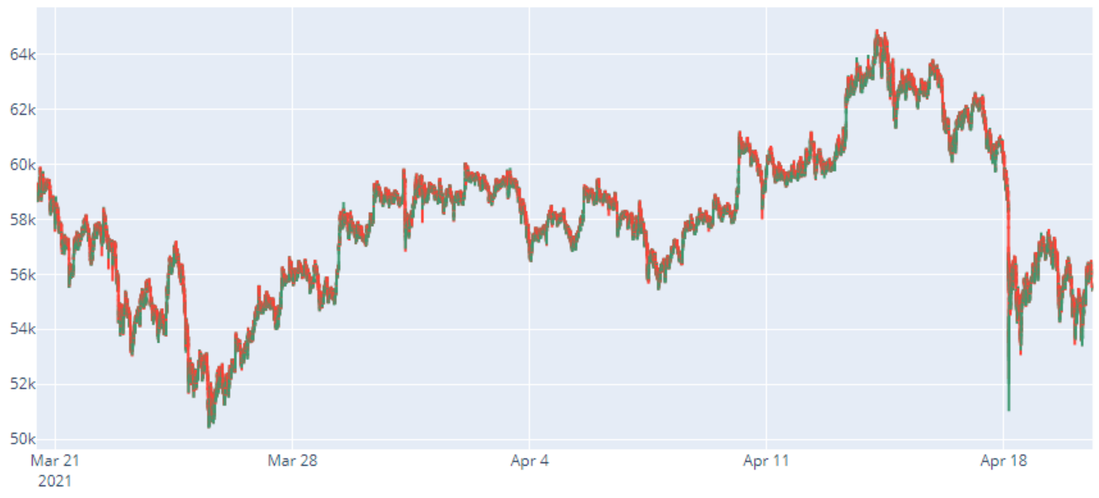


Figure 4.17: Price of Bitcoin (USD) per minute on Kraken - 20 March until 20 April

At this point, our formed dataset consists of the attributes that refer to the timestamp, the price and the sentiment score at each minute between the 20th of March and the 20th of April (with some minute-frames missing). In order to prepare the dataset for the prediction we create a new attribute, the 'Previous Close Price'. This refers to the closing price of Bitcoin in the previous time-frame, so in our case one minute before the moment in time we want to make the prediction for. This will help to train the model, since one the basis of our time-series prediction is the previous price so that we can predict the next one. The other attribute we create is called 'Moving Average of Close Price' and it is the moving average of the close price for a window of 100 preceding values (equation 4.6). The thought process behind this decision is that by having a number that tracks down the progress made in the change of values through time (namely in the last 100 minutes) we can use it as a valuable feature in our model. As we move from price value to price value, this number changes slightly and by feeding it to our model we produce better results.

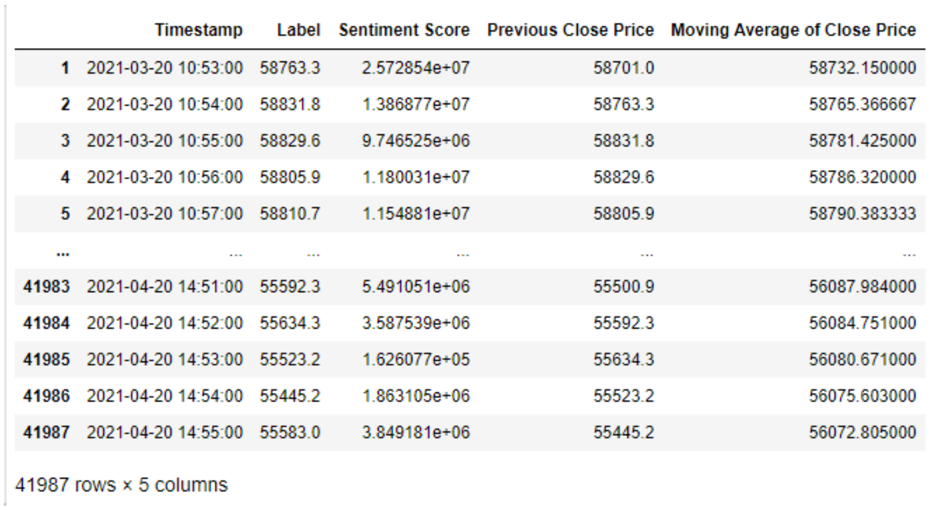
$$\overline{p}_{\mathrm{SM}} = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i} \tag{4.6}$$

Where:

$$\overline{p}_{\text{SM}} = \text{the simple moving average of the P values}$$

$$\text{n} = 100$$

Another thought was to add one more attribute, the moving average values of the sentiment score, but after testing it out we reached the conclusion to not include it because it caused the results to become more erratic. The current closing price is our target variable for the training process, our so-called Label. Our training dataset has reached the point depicted in Figure 4.18.



	Timestamp	Label	Sentiment Score	Previous Close Price	Moving Average of Close Price
1	2021-03-20 10:53:00	58763.3	2.572854e+07	58701.0	58732.150000
2	2021-03-20 10:54:00	58831.8	1.386877e+07	58763.3	58765.366667
3	2021-03-20 10:55:00	58829.6	9.746525e+06	58831.8	58781.425000
4	2021-03-20 10:56:00	58805.9	1.180031e+07	58829.6	58786.320000
5	2021-03-20 10:57:00	58810.7	1.154881e+07	58805.9	58790.383333
...	...	...	...	...	...
41983	2021-04-20 14:51:00	55592.3	5.491051e+06	55500.9	56087.984000
41984	2021-04-20 14:52:00	55634.3	3.587539e+06	55592.3	56084.751000
41985	2021-04-20 14:53:00	55523.2	1.626077e+05	55634.3	56080.671000
41986	2021-04-20 14:54:00	55445.2	1.863105e+06	55523.2	56075.603000
41987	2021-04-20 14:55:00	55583.0	3.849181e+06	55445.2	56072.805000

41987 rows × 5 columns

Figure 4.18: Training Dataset

Our preprocessing procedure includes deleting the variable that keeps track of the date and time of the tuple. Since this is a time-series problem we only care if the values are consecutive and not about the exact timestamp of our row. We drop missing values, make the label column our y value (value to predict or dependent variable) and then drop the label column and we make all the other variables correspond to our x values (values used as predictors or independent variiables). For our prediction we use the algorithm XGBoost, explained in detail in the section 2.8.

First the GridSearch algorithm is used to find the ideal parameters of our XGBoost model. Grid search is basically an optimization algorithm that allows you select the best parameters for your optimization problem from a list of parameter options that you provide, hence automating the 'trial-and-error' method [48]. By exploring some options we reach to the conclusion that the ideal parameters are:

- a **max depth** of 3. This controls the size of the decision trees, the number of layers of the depth that the trees will have. Shallow trees with not a lot of depth will have a hard time capturing the details of the problem, whereas deeper trees will have the opposite effect, they will capture too many details leading to the over-fitting of the problem.
- a **learning rate** of 0.01 (also found as eta in documentation). This controls how much we slow down the learning in the gradient boosting model by applying a weighting factor for the corrections by new trees when added to the model.
- the **number of estimators** is set to 100. This controls the number of decision trees in the model. We impose a cap on the number of trees because we might reach some point where adding more trees leads to no improvement in performance. If we want to find out why we have to think how the model is constructed. It is sequential and each new tree attempts to correct the errors made by the sequence of the previous tree.
- the **colsample by tree** is set to 0.7. This controls the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. In other words it defines what percentage of features (columns) will be used for building each tree. So for each tree, 70% of features are selected randomly to construct it.

After figuring out the parameters of our model, we compose our functions that will help with the prediction. The bootstraping function includes preprocessing the data in the way that we mentioned before, fitting the parameters chosen after the GridSearch algorithm and fitting our predictors and our values to predict to the model. The prediction function includes the inputs which are the last minute sentiment score collected from the real time Twitter stream, the previous minute close price of Bitcoin and the moving average value of the close price of Bitcoin for the last 100 time windows (one minute each). The retraining function adds the current Bitcoin price as a feature along with the rest of the features to predict the price in the next time-window.

In order to use these functions to make our prediction, we first connect to the file created at the Twitter Stream Notebook, receiving the data for the past minute and adding them up to calculate the sum of the previous minute sentiment scores. We calculate the moving average of the prices in the latest hour and use it as one of the inputs to our prediction function, along with the sentiment score in the last minute and the previous closing price. We print out the prediction for the closing price in the current time-frame and the actual current price of Bitcoin (through CryptoCompare's API [49]). The current price is used both to test our prediction and to improve it. The improvement comes by using it as a feature to retrain the model, along with the other features that we used in the prediction. We save the data on the text file and plot it accordingly (we will showcase the plots and the overall results in the next chapter).

Chapter 5

Results

In this chapter we will be showcasing the results of our prediction (by using both the full dataset and the non-botted one for training) as well us some plots and some error metrics to highlight the accuracy of the model.

5.1 Prediction Plots

By using the full dataset to train the model we still obtained some pretty satisfying results. Figure 5.1 shows the plot of predicted and actual values after 70 minutes have passed. We notice that the price difference is somewhat small. It comes as no surprise that by removing botted accounts, the prediction becomes even more accurate. Figure 5.2 shows the same plot as before, but this time for the non-botted dataset. We notice that the points are actually closer than in the full dataset prediction.

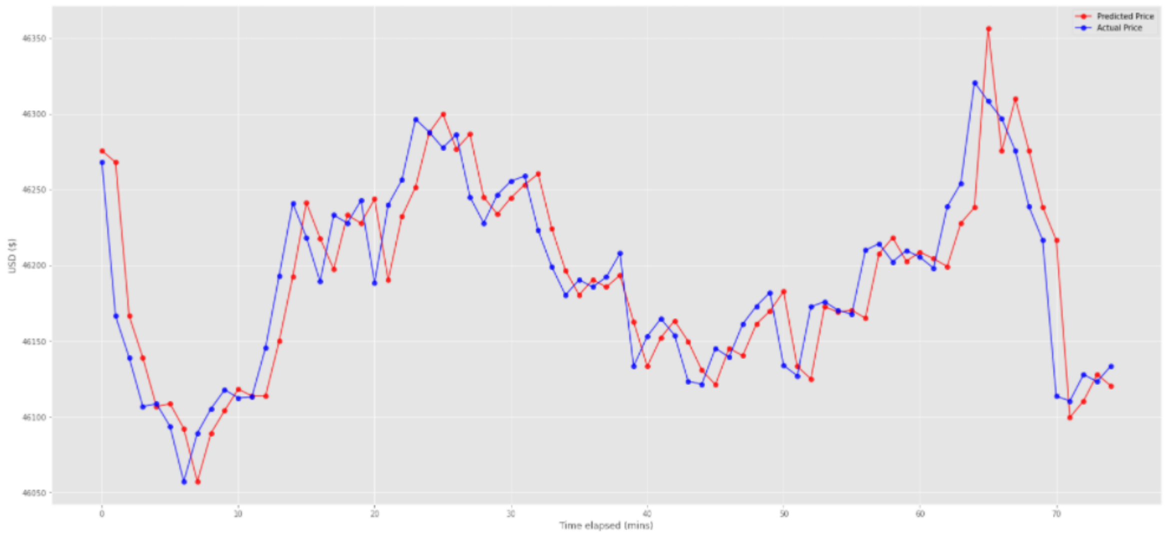


Figure 5.1: Actual and Predicted Prices Full Dataset - 70 minutes

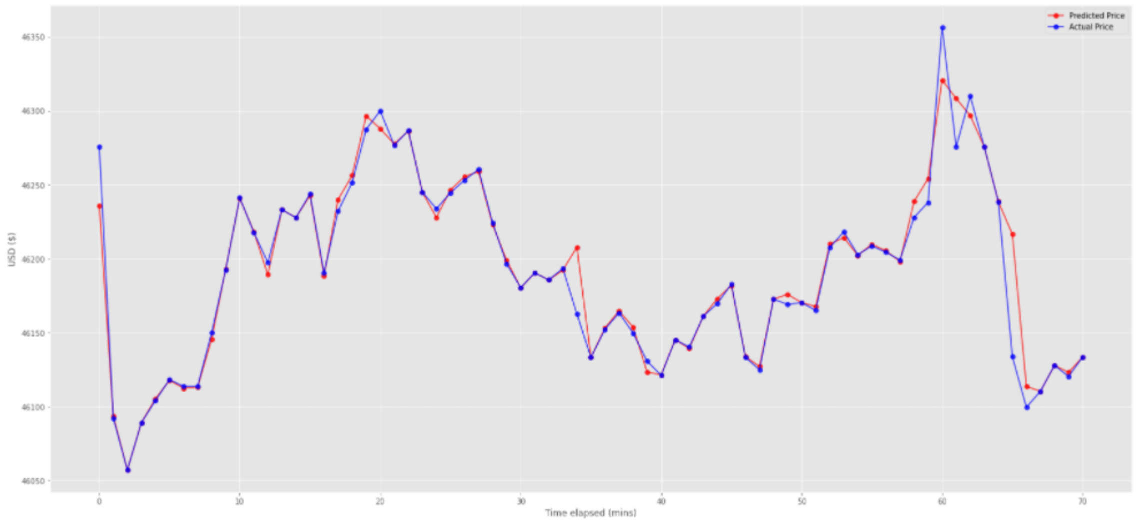


Figure 5.2: Actual and Predicted Prices Non-botted Dataset - 70 minutes

For the full running of our code (around 16 hours) we have the results shown in Figure 5.3 and Figure 5.4 for each dataset respectively.

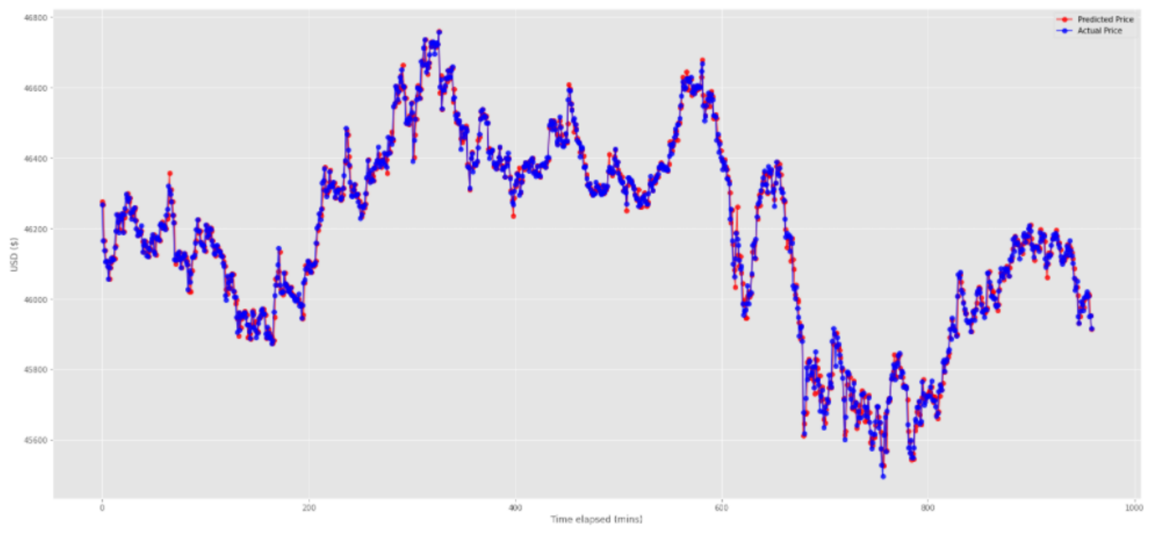


Figure 5.3: Actual and Predicted Prices Full Dataset

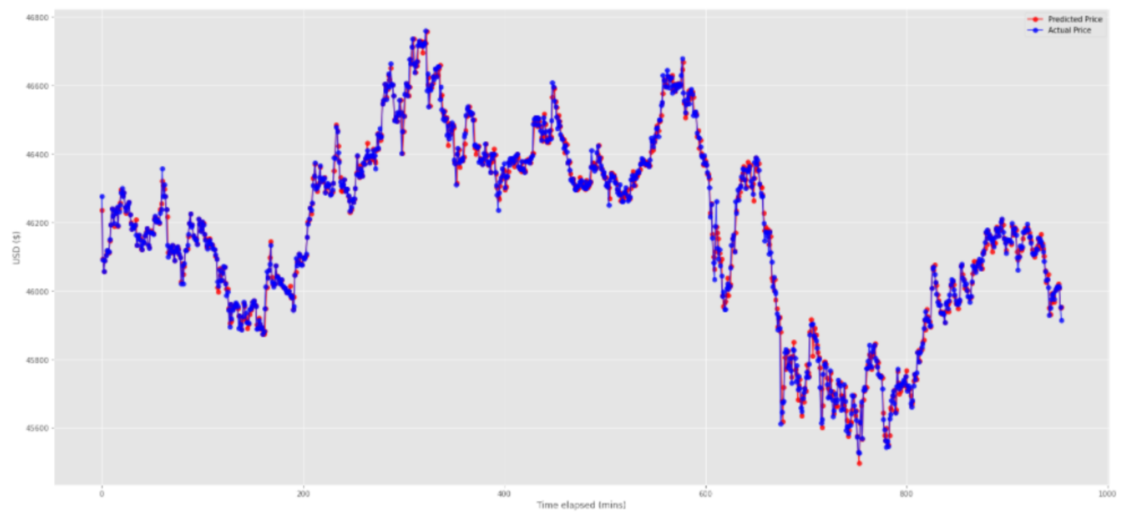


Figure 5.4: Actual and Predicted Prices Non-botted Dataset

5.2 Feature Importance

We use the feature importance function to get a general idea about which features the model is relying on most to make the prediction. This is a metric that simply sums up how many times each feature is split on. We can see in Figure 5.5 that the most important feature for a smaller run of the full dataset is the last minute score and the least important one is the moving average of the price.

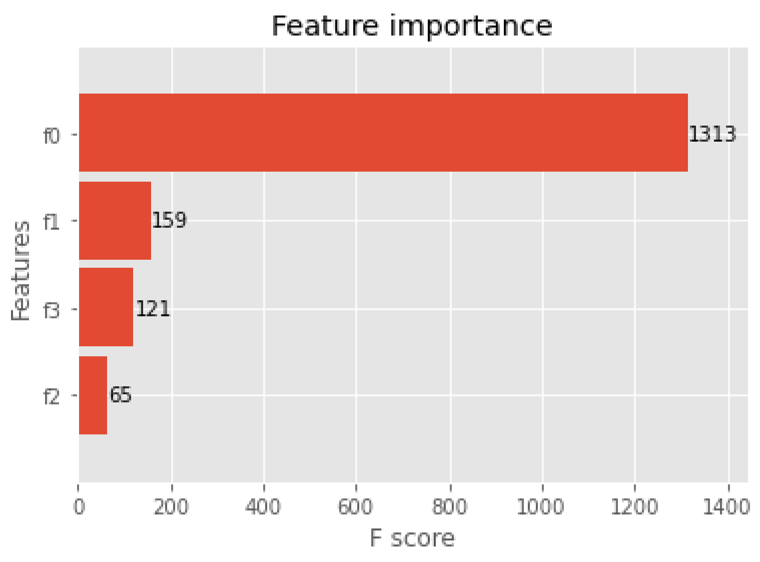


Figure 5.5: Feature Importance

5.3 Errors

Aside from the plots that help the visual types realise the success of our prediction, the most important teller of the performance of the model are the error metrics.

For starters we check the **Mean Forecast Error**, which is the mean value of the list that contains as elements the difference between the real and the predicted values. For the full dataset it is 0.174, which means that the real value is by average 0.17 USD higher than the predicted value . For the non-botted dataset it is equal to -0.285, which means that the predicted value is by average 0.28 USD higher than the real value. Overall this is not the most important metric, since it actually includes the sign of each element in the final calculation.

The **Mean Absolute Error** is the same as the Mean Forecast Error, but with the absolute values of each element considered in the calculation. For the full dataset it is equal to 17.92 and for the non-botted dataset it is 14.52

The **Mean Absolute Percentage Error** is the average times 100 of the list that contains elements that are the outcomes of the calculation $\frac{(RealValue-PredictedValue)}{RealValue}$. For the full dataset it is equal to 0.0388% and for the non-botted dataset it is equal to 0.0314%.

The **Root Mean Squared Error** is equal to the sum of the list that contains elements that are the outcomes of the calculation $\sqrt{(RealValue - PredictedValue)^2}$. For the full dataset it is equal to 26.95 and for the non-botted dataset to 24.66. It is in our opinion the most important metric for our particular project and that is why we also visualised it in the course of time for our two datasets (Figure 5.6, Figure 5.7).

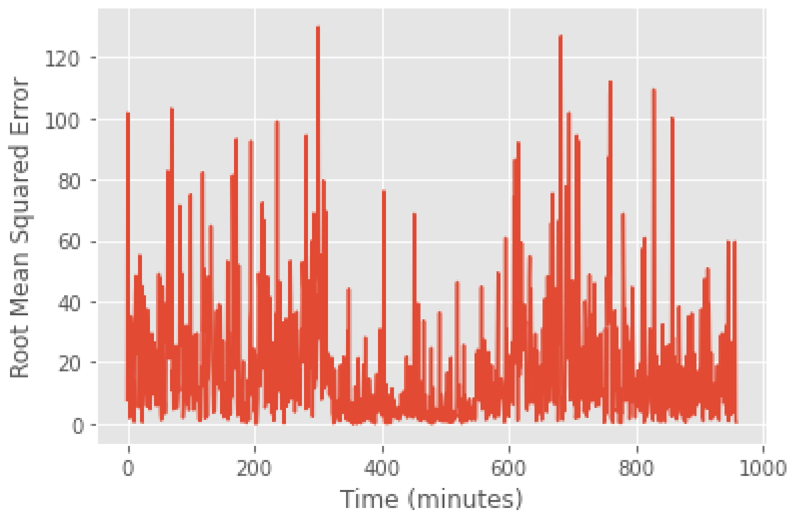


Figure 5.6: Root Mean Squared Error - Full Dataset

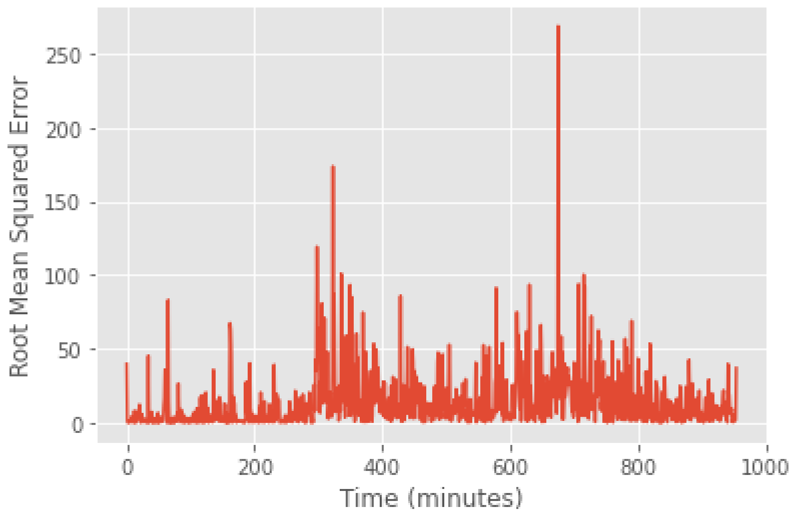


Figure 5.7: Root Mean Squared Error - Non-botted Dataset

Chapter 6

Conclusion

We will summarize the study prepared in the context of this thesis, while also providing some future work that could be done to further improve this project.

6.1 Summary

The main purpose of this thesis was to predict the price of Bitcoin in a certain time window by performing sentiment analysis on data extracted from Twitter. In order to move towards that direction we gathered a substantial amount of tweets on that subject in an efficient way. We detected the botted accounts among them by using the specially constructed filters, reducing the size of the dataset by 20% and thus removing a lot of noise. The preprocessing actually took a lot of time and that is considered a downside because it hindered the progress of the whole project. Maybe with some different optimisation we could have saved some time during the preprocessing stage. We performed sentiment analysis, including the specific terminology related to cryptocurrency in the process, which is considered quite innovative. The correlation analysis showed some weak positive correlation between the price and the tweets, which was later disproven by the results of the prediction (there is actually a substantial amount of predictive power in that pair). In order to predict the price we started a stream which extracted tweets and calculated their sentiment score on the spot. By summing up the minutely sentiment score and including the price of Bitcoin on that same minute and some other features specifically constructed for this purpose we create the input for our model. Our model learns from past failures by getting retrained and by the sole nature of XGBoost algorithms. We predicted the price of Bitcoin after one minute by using XGBoost with surprisingly low RMSE scores. The fact that the RSME score for the non-botted dataset is even lower than the one for the full dataset shows that removing bots was a step towards the right direction. The graphs depicting the results of the sentiment analysis, the correlation analysis and the prediction proved to be useful for the visual types in order to fully grasp the outcomes. We can come to the conclusion that is possible to predict the price accurately in a precise, cost and time efficient way.

6.2 Future Work

Overall the research done and the results produced seem very promising, but with some extensions and some improvements we can achieve even better outcomes. We dealt exclusively with Bitcoin but there are lots of other projects that could draw our attention (for example Ethereum, Cardano). These cryptocurrencies, also called altcoins, are rising day by day, they are having an abundance of mentions on social media and pose investment opportunities. So it would seem only fair to also include them (or maybe the top 10 cryptocurrencies in terms of market capitalization) in the price prediction project and gauge the results. There was a thought to include some extensions to the sentiment analysis model, like a sarcasm detection tool (which was dismissed because we did not notice a lot of sarcasm existing in the text that was mined) and the inclusion of a financial corpus (which was not included in the end). We noticed that some botted text slipped away from our filter so a future venture could be to improve the filter and catch 100% of the botted tweets that exist in the dataset. By using different algorithms and testing them with varying parameters we could improve the accuracy of our model. Also there are other factors that drive the prices that could be included in the model, like tweet volume, coin traded volume, instabilities due to external circumstances (for instance, El Salvador giving away 30$ worth of Bitcoin to citizens could cause the price to rise or fall [50]). The biggest challenge in my opinion would be to accurately predict the price by using a large time window, for example one month into the future. That would prove to be very profitable for the user as well as something worth publishing.

Bibliography

[1] Market capitalization of bitcoin from april 2013 to august 15, 2021(in billion u.s. dollars). `https://www.statista.com/statistics/377382/bitcoin-market-capitalization/`.

[2] G. T. Panger. *Emotion in Social Media. UC Berkeley.* PhD thesis, 2017.

[3] Brendan O'Connor, Ramnath Balasubramanyan, Bryan Routledge, and Noah Smith. From tweets to polls: Linking text sentiment to public opinion time series. volume 11, 01 2010.

[4] PAUL TETLOCK. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62:1139–1168, 02 2007.

[5] Hongkee Sul, Alan Dennis, and Lingyao Yuan. Trading on twitter: The financial information content of emotion in social media. pages 806–815, 01 2014.

[6] Pieter J. de Jong, S. Elfayoumy, and Oliver Schnusenberg. From returns to tweets and back: An investigation of the stocks in the dow jones industrial average. *Journal of Behavioral Finance*, 18:54 – 64, 2017.

[7] Shubhankar Mohapatra, Nauman Ahmed, and Paulo S. C. Alencar. Kryptooracle: A real-time cryptocurrency price prediction platform using twitter sentiments. *CoRR*, abs/2003.04967, 2020.

[8] Jethin Abraham, Daniel Higdon, J. Nelson, and J. Ibarra. Cryptocurrency price prediction using tweet volumes and sentiment analysis. 2018.

[9] Connor Lamon, Eric Nielsen, and E. Redondo. Cryptocurrency price prediction using news and social media sentiment. 2017.

[10] Evita Stenqvist and Jacob Lönnö. Predicting bitcoin price fluctuation with twitter sentiment analysis. 2017.

[11] Olivier Kraaijeveld and Johannes De Smedt. The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, 65:101188, 03 2020.

[12] Franco Valencia, Alfonso Gómez-Espinosa, and Benjamin Valdes. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *Entropy*, 21:1–12, 06 2019.

[13] David J. Fiander. 11 - social media for academic libraries. In Diane Rasmussen Neal, editor, *Social Media for Academics*, Chandos Publishing Social Media Series, pages 193–210. Chandos Publishing, 2012.

[14] Statista twitter active users. `statista.com/statistics/303681/twitter-users-worldwide/`.

[15] Twitter statistics oberlo. `https://www.oberlo.com/blog/twitter-statistics`.

[16] Twitter developer platform. `https://developer.twitter.com/en`.

[17] A short history of cryptocurrencies. `https://daviescoin.io/blog/a-short-history-of-cryptocurrencies.html`.

[18] Cryptocurrency crime and anti-money laundering report,february 2021. `https://ciphertrace.com/2020-year-end-cryptocurrency-crime-and-anti-money-laundering-report/`.

[19] Cara Lapointe and Lara Fishbane. The blockchain ethical design framework. *Innovations: Technology, Governance, Globalization*, 12:50–71, 01 2019.

[20] Bitcoin consumes 'more electricity than argentina'. `https://www.bbc.com/news/technology-56012952`.

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[22] C. Bendiksen, S. Gibbons, and Eugene Lim. The bitcoin mining network-trends , composition , marginal creation cost , electricity consumption & sources. 2018.

[23] What is natural language processing? introduction to nlp. https://algorithmia.com/blog/introduction-natural-language-processing-nlp.

[24] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[25] Sentiment analysis challenges and how to overcome them. https://www.repustate.com/blog/sentiment-analysis-challenges-with-solutions/.

[26] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 01 2015.

[27] Vader github repo. https://github.com/cjhutto/vaderSentiment.

[28] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[29] Github repository of xgboost. https://github.com/dmlc/xgboost.

[30] A gentle introduction to xgboost for applied machine learning. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[31] Xgboost simply explained (with an example in python). https://www.springboard.com/library/machine-learning-engineering/xgboost-explainer/.

[32] Xgboost algorithm: Long may she reign!. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.

[33] Xgboost mathematics explained. https://towardsdatascience.com/xgboost-mathematics-explained-58262530904a.

[34] Martin Hentschel and Omar Alonso. Follow the money: A study of cashtags on twitter. *First Monday*, 19(8), Aug. 2014.

[35] Bitcoincharts.com bitcoin price on kraken. https://bitcoincharts.com/charts/krakenUSD#rg30zig1-minzczsg2021-04-21zeg2021-04-22ztgSzm1g10zm2g25zv.

[36] Bitcoin volume weighted average price (vwap) explained. https://bitcoinchain.com/bitcoin_price.

[37] Twitter privacy policy. https://twitter.com/en/privacy.

[38] Social bots – detection and impact on social and political events. https://www.boxcryptor.com/en/blog/post/social-bots-detection-examples-of-political-impact/.

[39] Researchers: Nearly half of accounts tweeting about coronavirus are likely bots. https://www.npr.org/sections/coronavirus-live-updates/2020/05/20/859814085/researchers-nearly-half-of-accounts-tweeting-about-coronavirus-are-likely-bots?t=1629974460600.

[40] 6 outrageous moments in crypto twitter scam history. https://www.coindesk.com/6-outrageous-moments-crypto-twitter-scam-history.

[41] Accessing text corpora and lexical resources - chapter 4.1. http://www.nltk.org/book/ch02.html.

[42] When (not) to lemmatize or remove stop words in text preprocessing. https://opendatagroup.github.io/data%20science/2019/03/21/preprocessing-text.html.

[43] Expanding english language contractions in python. https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python.

[44] Coinmarketcap.com crypto glossary. https://coinmarketcap.com/alexandria/glossary.

[45] Blockspot.io crypto dictionary. https://blockspot.io/crypto-dictionary/.

[46] Pearson product-moment correlation. https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php.

[47] Kendall rank correlation explained. https://towardsdatascience.com/kendall-rank-correlation-explained-dee01d99c535.

[48] Grid search optimization algorithm in python. https://stackabuse.com/grid-search-optimization-algorithm-in-python/.

[49] Current price of bitcoin in usd (cryptocompare) . https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC&tsyms=USD&api_key=f311d68ba556b86506598d21aba8bcb0833a7c0431c859463e75ff8a934d0f6c.

[50] Bitcoin: El salvador divided over legal tender law. https://www.bbc.com/news/technology-58438525.

Appendix

Code Repository

In this appendix we explain where to find the code for this project and how to run it.

1 How to

You will find the code in GitHub using this link `https://github.com/FivosTzavellos/Thesis_ECE_UTH`. All the codes can run in Jupyter Notebook. In order to run them you need to download Anaconda from here `https://www.anaconda.com/products/individual-d` and follow the instructions to install and open Jupyter Notebook either from the cmd or from Anaconda. Alternatively you can open Google Colab `https://colab.research.google.com/` and upload the files there to run them online without the need to download anything.

In order to run the Notebooks titled ”1. Data Collection” and ”9. Twitter Stream” you need your own Twitter Developer Account, a new Project with the API and secret API key. In order to run parts of the other code you need the dataset as it was extracted by the author (`https://uthnoc-my.sharepoint.com/:x:/g/personal/ftzavellos_o365_uth_gr/EZy919GZ9ntKqaT4KxMgevcBEmQKZ9wDvcUwIUO1I-Pl0A?e=VuNgI2`) or you need to create your own dataset.

The cryptoslang text file contains definitions of cryptocurrency specific terms and the excel file with the same name contains terms and the sentiment score we assigned to them. The Suspicious n-grams text file contains 2 lists of words and bigrams that we deemed suspicious of bot-like behaviour (more explanation in section 4.1).