Predicting Bitcoin Price using Machine Learning

Thesis · May 2018

CITATION
1

READS
2,138

3 authors, including:

Ignas Bagdonas
Aalborg University
1 PUBLICATION **1** CITATION

SEE PROFILE

AALBORG UNIVERSITY
COPENHAGEN

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Secretary: Maiken Keller

Semester: ITCOM6
Title: Predicting Bitcoin Price using Machine Learning
Semester theme: IT, Communication and New Media P6
Project period: 01/02/18 – 24/05/18
Supervisor: Per Lynggaard
Project group: 6.06

Members:

Matas Navickas

Ignas Bagdonas

Vu Ngoc Viet Nguyen

Can bitcoin price changes be predicted? Many open-source tools, libraries and cloud services for machine learning are accessible for everyone. Social media sites contain posts from millions of users about any topic. Research on crowd predictions suggest that a large and diverse group is capable of predicting more accurately compared to experts. Therefore, the next step is how to get the large amount of data required. We have analyzed, designed and built a distributed crawler capable of collecting and processing millions tweets for less than 5$. We have extracted sentiment data with VADER algorithm and used the results to train Linear Regression, Multilevel Perceptron and LSTM models. We have then compared the models to find out the best results.

Pages: 93
Finished: 24/05/2018

This page is intentionally left blank

TABLE OF CONTENTS

1 INTRODUCTION

To predict the future, has always been a goal or dream of humanity, however humans have always been terrible at it[1]. Predicting the price movements for cryptocurrency or bitcoin (more about bitcoin in section 3.1) is a relatively similar to predicting stocks or the price of USD going up or down. However, unlike a company with physical buildings and people working in it, bitcoin is purely digital and therefore very difficult to understand when and why the price would change. With no physical entity and the way bitcoin works, most of the effect on the price is based on how the world feels about it. For this reason, understanding people is one way to help predict bitcoins future. Many researchers (papers in section 3.2) have investigated sentiment analysis on different kind of social media to get a better understanding of how people feel. However as mentioned earlier, humans are terrible at predicting, even with information at their hand. For this reason, by using a machine learning algorithm, the algorithm will try to predict the future price based on the information given to it.

1.1 RESEARCH QUESTION

This project will try to make a system that can predict bitcoin with the help of machine learning, social network data, data from bitcoin and news data, which leads to the main research question:

Can Bitcoin price changes be predicted by using machine learning based on historical and sentiment data from social media and news sites?

Sub-questions:

- How to collect features and process them into useful data?
- How to effectively combine information from sentiment news data?
- How to determine impact of the news and its relevance for the price?
- How to test and validate success rate of predictions?

A good coverage of sub-questions above, has great influence on actual prototype implementation in this project.

[1] Article: https://www.theatlantic.com/science/archive/2017/11/humans-are-bad-at-predicting-futures-that-dont-benefit-them/544709/ [accessed 22/05/2018]

1.2 DELIMITATIONS

This project will not investigate law and regulations regarding consequences of data extractions and security issues etc.

This project will not investigate economical aspect, such as how to make money from this project or how to sell the prototype.

This project will not investigate user interface or have any interaction with potential users of the system.

Because of limited resources a prototype will be created rather than a full version to prove the concept of the system.

1.3 STRUCTURE OF REPORT

The chapter after introduction is the methodology (2), it goes through the process and methods used throughout the project lifetime. After that is the state of the art chapter (3), which gives a more in-depth knowledge of other researchers work and tools needed to build the prototype. When knowledge has been acquired the analysis chapter (4) right after, goes through the different methods that can be used and through arguments choose the best solution to use to create a requirement specification for the prototype in the end (4.5). After the requirement list comes the design and implementation chapter (5), which goes through how the prototype was being implemented. This are followed by testing (6), which goes through a few use cases to showcase how they were tested. The report ends with a discussion (7) and conclusion chapter (8), which goes through the problems and decisions that were made and how that impacted the project in a whole.

2 METHODOLOGY

This section explains the choice of process model used throughout the project and the method used with requirements gathered

2.1 PROCESS MODEL

The process model that fits the project the best, is the iterative prototype model. With the basic application being build and changed over time as new information and data is learned. The benefits from using this is the more accurate prediction that the application can perform with new information constantly updating itself. The final system will not be the finishing product, however a working prototype that can show the concept and after more iteration more features can be built on top of.

Before building the prototype, research had to be done, to understand what was necessary to know and do to be able to build it. This was done mostly through Aalborg University's search engine and other research papers found on google. Though most research paper concludes with an overall positive result, they were not looked at as a final answer, but however taken into consideration when design and implementation were made, to better understand and analyze the outcome. This means that their results were looked upon with critical eyes.

Although the process model is iterative, the report is written in a waterfall model, this means that the first chapter of the report is the research, then the analysis, followed by implementation and design, then testing and lastly concluded with discussion and conclusion.

2.2 REQUIREMENTS

Gathered throughout the analysis chapter (4) to create what is believed to be needed in the system. The requirements are then divided into a *MoSCoW* table, prioritizing which of the requirements to implement first for a barebone prototype. This is done by putting all the requirements into a *Must, Should, Could* or *Wont* columns. In this iteration all the *Must* requirements must be implemented, however if a second iteration of the prototype were to be made, most if not all the *Should* would be implemented. The requirements are prioritized based on time needed for implementation versus how core and important it is.

3 STATE OF THE ART

This chapter goes through six sections, to get a better understanding of how to dissect the research question into smaller pieces to solve. First is the introduction of bitcoin and then in-depth literature review of how other researchers has tackled a similar or related problem. With machine learning being a part of the research question, understanding which cloud computing services exist and what are the differences between them is in the second part of a chapter. To be able to gather sentiment data, it is necessary to understand what platforms exist and which to gather from. Lastly, technical tools and methods to gather and analyze said data.

3.1 BITCOIN CRYPTOCURRENCY

The first decentralized cryptocurrency was created in 2009 called *Bitcoin*[2]. In 2010 price for one bitcoin never reached above one dollar[3], but in May 2018 it was the one with the highest market cap of around 149 billion dollars with a little more than 17 million bitcoins in circulation[4]. This makes one bitcoin worth around 8700 dollars (Market cap / circulation). Bitcoin is a digital currency, which can be used to buy or trade things electronically, however what makes it different from normal currency is the decentralization of the system by using blockchain technology[5]. By being decentralized, it has no single entity controlling the network but instead maintained by a group of volunteers running on different computers around the world. Physical currency such as USD can have unlimited supply of currency, if the government decides to print more, which can change the value of the currency related to others. However, the increase of Bitcoin circulation is heavily guarded by an algorithm, which allows only a few Bitcoin to be created every hour by miners until it reaches a total cap of 21 million bitcoins[6].

The decentralization of the system in theory allows for anonymity, since there are no bank or centralized entity to confirm and validate transactions that are happening. In practice however, when a transaction is made, each user uses their identifier, known as addresses during the transaction. These addresses are not associated with a name, but because the transaction must be transparent because of the way decentralization of the system works, it is possible for

[2] Article: https://www.coindesk.com/information/what-is-bitcoin/ [accessed 22/05/2018]
[3] Link: https://www.buybitcoinworldwide.com/price/ [accessed 22/05/2018]
[4] Link: https://coinmarketcap.com/all/views/all [accessed 22/05/2018]
[5] Article: https://blockgeeks.com/guides/what-is-blockchain-technology [accessed 22/05/2018]
[6] Link: https://www.bitcoinmining.com [accessed 22/05/2018]

everyone to see it. A disadvantage of the system is that, in case of losing the address, forgetting it or wanting to reverse a transaction, because there is no authority involved, it is not possible to do anything about it.

3.2 LITERATURE REVIEW

In recent years, there has been a lot of research done in both finance trading (stock, forex etc.) and sentiment analysis supported by AI. This section will explore some of the more influencing works that are closely relevant for the research problem, which will be used later in the analysis section.

3.2.1 PREDICTING BITCOIN PRICE FLUCTUATION WITH TWITTER SENTIMENT ANALYSIS (2017) - E. STENQVIST, J. LÖNNÖ

In 2017 two master degree students in Computer Science (KTH Royal Institute of Technology, Stockholm) were investigating almost identical research problem [1]. The questions that they were trying to answer were:

- Is there a correlation between Twitter sentiment and BTC price fluctuation?
- Can a naive prediction model based on sentiment changes yield better than random accuracy?

Briefly, their main idea was to build accurate sentiment analyzer to predict whether the price will go up or down. Students claim that they were able to achieve 79% prediction accuracy, which sounds very promising. The project is based on some of the papers that are analyzed in this report as well, so it is important to emphasize that research done by Swedish students are strongly influencing this report. Their best practices will be followed and their weak points will be analyzed in order to improve final results.

Here is a high-level overview of what they have done to achieve this prediction power.

- Collecting BTC/USD historical exchange rates. CoinDesk API was used to collect OHLC (Open High Low Close prices) data.
- Collecting Twitter's messages in real time. Researchers have used open source python framework which works with Twitter's public API, called Tweepy. To collect relevant tweets these synonyms were used: Bitcoin, BTC, XBT, satoshi. Total of 2 271 815 Bitcoin related tweets were gathered from the Twitter API.
- Noise reduction in gathered data. All non-alphabetic symbols were removed from tweets. Also non-unique, suspicious intent or bot generated tweets were discarded. Students

have analyzed 100 thousand tweets manually to find suspicious bot accounts, hashtags, words or sentences.

- Sentiment analysis using VADER open source software (read more in 3.5.4). It evaluates every tweet with score from -1 (very negative) to 1 (very positive), which is saved as an extra column in dataset. VADER creators claim that it outperforms most of popular AI sentiment tools when analyzing social media short length messages.
- Aggregating sentiment scores. Tweets are grouped in a selected timeframe and their sentiment scores are averaged, to get average sentiment for that timeframe.
- Price prediction is tested with different timeframes[7] and shifts forward to find the best fit.

In the end, 83% accuracy was achieved (With 1 hour timeframe and shift 3 x 1 hour). All predictions are filtered by a threshold - only predictions with higher sentiment change rate than the threshold are taken into consideration. Only 12 predictions makes above the threshold in 1 hour timeframe.

In discussion of their report, authors mention some of the reasons that are in the way to achieve better accuracies: objective users, too strict spam filtering, lack of fintech domain lexicon in VADER, also predictions only say whether price will go up or down, and not by how much, this makes it difficult to measure true success of the implementation.

Researchers also noticed that there was big correlation between Bitcoin price and the number of Tweets posted, but this was not investigated further.

3.2.2 Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies by Young Bin Kim et al. (2016)

Students of Korea University in Seoul have investigated price prediction possibilities using machine learning and user sentiment analysis [2]. They have documented their process in a brief research article.

Firstly, authors acknowledge some other influential work done by other researchers [3] [4]. Students here point out that most of the researchers investigate generic social feeds such as twitter, Reddit etc., meanwhile they are planning to analyze user sentiment in community

[7] [5 min; 15 min; 30 min; 45 min; 1 hour; 2 hours; 4 hours]

forums [5] [6], that are related to digital currencies. Discussions in these forums are mostly focused about a specific topic (cryptocurrency). This property might reduce some of the noise in collected data compared to data from generic sites mentioned above.

Authors' main goal is to propose *"a method to predict fluctuations in the price and number of transactions of cryptocurrencies".* Researchers state that they can predict price fluctuations of Bitcoin, Ripple and Ethereum with approx. 74% precision, which is quite impressive.

Here is a brief overview of their system, followed with a system overview in Figure 1:

1) Crawl all comments and replies in online communities for selected period of time (or all data, since cryptocurrency communities are relatively young). Report introduces to implementation of the crawler and methods to filter out data from ads and spam.
2) Sentiment analysis - tag comments with Positive, Negative etc. labels. Authors uses VADER [7] algorithm for comments mood polarity classification. Some examples of comments and their assigned tags using VADER are shown in Figure 1: System Overview. Source [2] below.
3) Collect historical currency data from markets and exchanges APIs (CoinDesk[8], Etherscan[9] and CMC[10])
4) Create a prediction model. To create the model authors performed analysis between tagged comments and price fluctuations. They have used Granger's Causality test [8] (if X causes Y, then X can be used to predict following Y values) to evaluate predictive information in the data. Predictions were generated using AODE [9] algorithm.

[8] Link: https://www.coindesk.com [accessed 22/05/2018]
[9] Link: https://etherscan.io/ [accessed 22/05/2018]
[10] Link: https://coinmarketcap.com/ [accessed 22/05/2018]

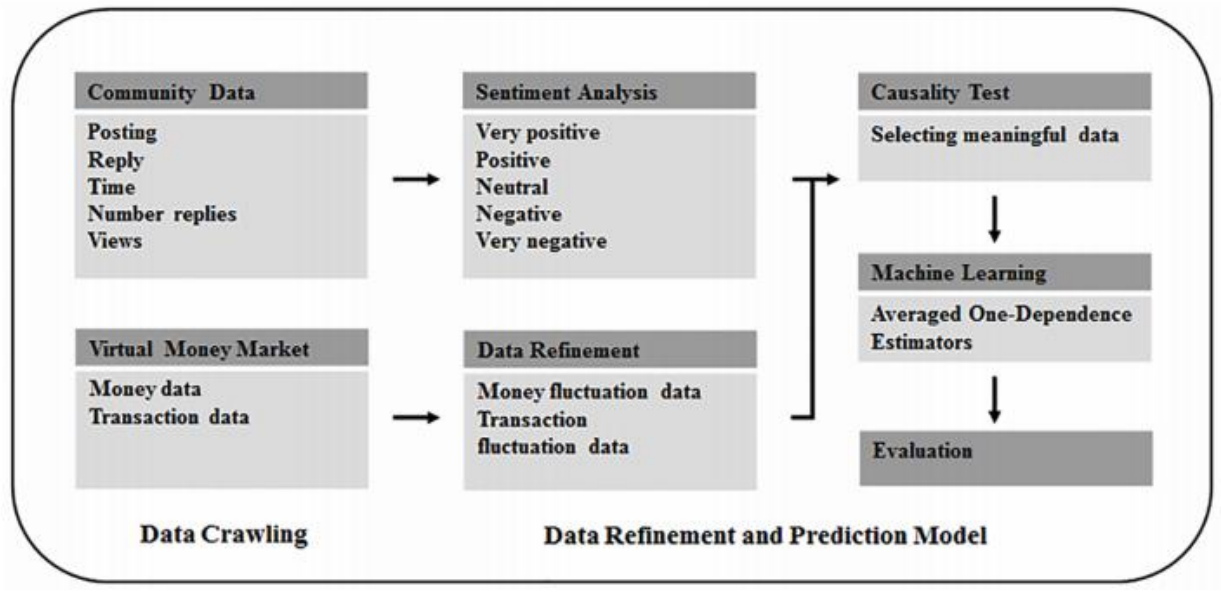

Figure 1: System Overview. ***Source [2]***

We can see from the Figure 1 above that authors not only used sentiment analysis, but as well number of replies and views as features. Figure 2 below illustrates presumed polarity of some of the tweet messages.

Opinion Criteria	Example topic sentences
Very Positive	"I am selling for $100 a Starbucks Gift card with a loaded balance of $20 worth of BTC" / "Bitcoin is the global currency of the Earth" / "How can 1 BTC eventually be worth $11 M"
Positive	"We are in Bitcoin Heaven" / "Bitcoin to eventually replace Apps like Uber" / "Russians can Pay Internet and phone bills with Bitcoin without fees"
Neutral	"Do you think Bitcoin will disappear or sopt being used?" / "What you like the best about Bitcoin?" / "Can Bitcoin make banks disappear?"
Negative	"Bitcoin: Should you stay or should you go?" / "Is there a way to earn at least $1 in BTC per hour?" / "IMF fears cryptocurrencies may circumvent capital controls"
Very Negative	"Bitcoin used to be involved in money laundering—will it become a huge problem?" / "Bitcoin cold storage—Hacked easily" / "Russia's Finance Ministry wants to ban Bitcoin"

Figure 2: Sentiment Analysis. ***Source [2]***

Authors also provide some of the formulas that they used, as well as results of causality testing and prediction validation[11] of Bitcoin, Ethereum and Ripple.

[11] 10-fold cross-validation

Finally authors were able to achieve results of 79% (with 6 day lag) and 77% (with 3 day lag) using 1 day timeframe. They also noticed that price had strong correlation with negative comments, but not so much with positive ones.

3.2.3 Stock Prediction Using Event-based Sentiment Analysis by Masoud Makrechi, Sameena Shah and Wenhui Liao (2013)

A brief research article [10] that is specifically focused on Stock Market price prediction using Sentiment analysis. Research has been made by Masoud Makrehchi from University of Ontario Institute of Technology, Canada in collaboration with Thomson Reuters Corp., New York. Researchers of this paper specifically investigates twitter feed and how to do a better job into classifying them.

After researchers introduce to some of the work in sentiment analysis field done by other engineers [11] [12] [13] [14] [15], they showcase their approach in a bit more detail.

In Related Literature section, authors elaborate on other researchers work in order to find social media factors/features that can be used for various future predictions. For example:

- Asur [13] [16] et al. tries to predict box office revenue of movies using twitter messages. In conclusion, results show that twitter sentiment could be used to predict upcoming movies success.
- Bandari [16] et al. propose method to predict popularity of some news story in social media. They use features like - source of the story, news category, subjectivity of the news, names in the story. Applying regression and classification using these features, they were able to predict ranges of popularity of a news story with 84% accuracy.
- Hong [14] et al. tries to predict popularity of recent twitter messages. They "*construct features based on message content, temporal information, metadata of messages and users, as well as the users' social graph.*"
- Gilbert [17], Zhang [18] et al. take somewhat similar approach, where they take some social media source (LiveJournal, Twitter). In LiveJournal case, author tries to estimate worry and fear from millions of posts and see how this reflects on S&P 500 stock prices. In other example, Zhang tries to estimate tweet's mood, based on *"tweet count of words expressing certain pre-decided mood (hope, fear, worry), number of followers of the mood, number of re-tweets of the moods, etc."*

- Bollen [15] predicts stock market using twitter feed as well. Sentiment analysis is approached using OpinionFinder [19] and POMS [20]. OpinionFinder assigns positive or negative label to a tweet, while POMS assigns of the mood labels (calm, alert, happy and etc.). Specific “calm” label showed high correlation with the market data, while other labels, including positive and negative indicated low correlation.

Authors state that all the approaches mentioned in their Literature Review either uses predefined mood words for unsupervised learning or manually generated training data for supervised training. However, their proposed solution generates training data automatically, so in other words - as much data as needed can be labelled without requiring any human/manual effort.

Some of the properties defined by the authors while analyzing twitter feed and social media to predict future:

- Texts from social media contains much more noise and less useful content than for example news articles.
- Social media text better represents general opinion and linguistic patterns than texts from news articles written by professionals of the field.
- Spread or scale of social media documents is huge comparing to some news articles.

Solution proposed by the authors has two main aspects: *“(i) learning sentiment from relevant labeled text using significant events, and (ii) using the learnt sentiment to predict future significant events.”*

Their system is made up of two parts: *“(i) automatically generating training data from unlabeled user tweets; (ii) building a sentiment classifier based on the training data.”*

In simple words automatic labeling (i) of unlabeled tweets is achieved by taking historical price of a stock. If price of a stock was going up, all the tweets will be labeled as positive in that given period, and vice versa if stock price was going down. Any other explicit positive or negative events for that item could be used to label the tweets.

Sentiment classifiers (ii). Two types of classifiers are implemented to be compared against each other to classify labeled tweets:

- A classifier that uses list of mood words and assigns each tweet to some of the mood class if that mood word appears in the text.

- Second classifier that selects positive features associated with positive tweets and vice versa.

Together with the positive and negative features, authors also used some features based on the metadata of the tweet, e.g. hashtags, stock ticker symbols, URLs, re-tweets, but those did not show any strong correlation with the stock price. Authors have collected around 30 million tweets.

For both feature vectors they have used Rocchio classifier [21] and were able to achieve precision over 90%.

Here is the proposed algorithm:

Algorithm 1 Stock market prediction using event-based supervised sentiment learning

Choose the significant event criteria.
Choose appropriate pre/post/contemporaneous tweets based on the chosen significant event criteria.
Assign an appropriate label for each tweet based on its associated event (for example, positive for rise or negative for fall).
Train a classifier on the labeled tweets.
Predict sentiment of new future tweets.
Aggregate tweet sentiments.
Take a long/short position based on the net aggregated sentiment.

Figure 3: Algorithm 1 Source [21]

After internal and external testing, it was quite clear that algorithm outperforms some other usual trading strategies (S&P was beaten by 20% in returns just in four months). Strength of this algorithm seems to be that they use very significant events to label their data, this helps to gather large amounts of training and testing data.

3.2.4 Sentiment Analysis of Twitter data: A Survey of Technique

A research paper written by Vishal A. Kharde (University of Pune, India) investigates sentiment analysis using various machine learning techniques. As the title suggests, research focuses on sentiment analysis of twitter data.

At first author provides very informative summary of different works and interesting approaches for classifying tweets. For example:

- Tweet classification using Naive Bayes algorithm and Support Vector Machine [22]

- Classifying tweets as objective or subjective and then assigning positive or negative labels on subjective tweets [23]. Feature space consists of retweets, hashtags, prior polarity of words and more.

Later the paper discusses preprocessing of datasets. Below is a list of actual steps that are applied on the data by the authors [24]:

- "Remove all URLs (e.g. www.xyz.com), hash tags (e.g. #topic), targets (@username)"
- "Correct the spellings; sequence of repeated characters is to be handled"
- "Replace all the emoticons with their sentiment."
- "Remove all punctuations, symbols, numbers"
- "Remove Stop Words"
- "Expand Acronyms(we can use a acronym dictionary)"
- "Remove Non-English Tweets"

The paper looks into popular features that can be extracted from the processed twitter datasets:

- Frequency or presence of words
- Presence of subjective adjectives and adverbs
- Opinion words
- Position of terms
- Negation - quite difficult feature to implement, as presence of negation must change polarity of the message (from positive to negative and vice versa)

In later parts author discusses various approaches for twitter sentiment classification, such as emoticons based or lexicon based analysis. One useful solution was using Thesaurus[12] dictionary to find similar words and group them under one category for feature extraction.

Researchers conducted an experiment trying to classify given twitter dataset. Below are the results of different machine learning approaches.

Method	Accuracy(Unigram)
Baseline	73.65
Naïve Bayes	74.56
SVM	76.68
Maximum Entropy	74.93

Figure 4: Accuracy table. Source [24]

[12] Link: http://www.thesaurus.com/ [accessed 21/05/2018]

3.2.5 The Value of Social Media for Predicting Stock Returns

Dissertation written by Michael Nofer, in TU Darmstadt, Germany, 2014 [25] investigates if user generated social media content can be used as data source to predict stock prices. The research discusses some key principles:

Wisdom of Crowds is a term from a book James Surowiecki (2004). It suggests that in many cases a prediction made by individual is less accurate than prediction made by a crowd.

The researcher compared forecast accuracy between professional analyst and the crowd of internet users and found out that crowd predicted with 0.59 point higher over entire year with T score[13] 5.43. In addition, they found that crowd performance decreased after professional analyst recommendations were introduced.

The authors looked at previous research on mood analysis and influence of mood on share returns. Given that many things can affect a person's mood the researchers found that weather such as cloudy days would affect the stocks negatively while sunny days would affect it positively. Another mood changing behavior is the daylight saving, which the following Monday gave worse return on the stocks because people were more risk averse because of being more anxious which is caused by the loss of sleep (the one hour from daylight saving). The reason, why positive mood increases the stocks can be understood with Affect Infusion Model (AIM). The basic idea is, that people view their decisions based on their mood, so having a positive mood, leads to the person to feel like taking a chance and maybe win some money, while having a negative mood would lead to a person perhaps being scared of the danger of losing money, and therefore less likely to invest.

The authors wanted to test two hypothesis, the first was "Increased social mood levels derived from twitter lead to a higher stock market return" and "increased follower-weighted social mood levels derived from twitter lead to higher stock market returns". The difference between them is based on a theory that mood is contagious, and spreads in a group, they decided to weight higher on the ones with more followers to represent that the spread is higher. The conclusion of the experiment was that there was no significant relationship between the twitter mood and the share returns, however the argument was that it might already have been included in the model. The second hypothesis was adding followers to the model as a weight, this lead to an increase of predicting the share returns, however they also noted that there were some issues that could

[13] Link: http://www.pmean.com/definitions/tstat.htm [accessed 22/05/2018]

arrive such as fake tweets, the dictionary approach missing words and slangs not being translated correctly.

3.2.6 Price Shock Detection With an Influence-Based Model of Social Attention By Keli Xiao, Qi Liu, Churanren Liu and Hui Xiong

A research paper [26] about whether attention in social media, is a factor, when a *Price Shock* happens in stocks (when unexpected changes to the price happens). To do this they developed a measurement called “periodic cumulative degree of social attention” (PCDSA) which takes into consideration of individual influence and how the information is propagated into the social network.

The social attention is modeled by looking at the social influence of each social network user and how they are related to each other. This is done by looking at how many posts a user has done and how many comments/likes another user has made during that period of time on those posts. This is to understand how much influence one user has on the other. They then put how many of the posts are related to stocks into the equation to understand the influence and spread of the stock posts in the network. Since posts are most relevant when they are new, they also use a diminishing effect for posts when they get older.

An important issue was noted that the length of the time window for data usage was important. If all posts from 1 year ago were used, it might cause more harm and create noise rather than being useful information and the same can be said if the period is short, there could be more bias regarding the data, since there is less data. Their solution was to look at how many posts were being made per day, and they decided prior to 1 week being the optimal time to gather data.

They collected stock related data from Shenzhen Stock Exchange and Shanghai Stock Exchange and social media data from Weibo.com, the biggest social network in China. The features collected from Weibo are as seen on the picture below Figure 5

Data	Feature Descriptions
Basic identifications	Weibo ID
	Account ID
	Post content
	Date and time
Influence related	Number of "likes"
	Number of reposts
	Number of comments
Reaction tracking	Reaction ID

Table 3 A Summary of Data Statistics

Data Sources	Properties	Statistics
Common	Time scale	10/08/2013–03/31/2014
	Number of days	174
Social media	Number of posts	139,855
	Number of accounts	20,410
	Number of posts per day	803.76
Stock market	Number of stocks	40
	Number of business sectors	10
	Number of trading days	119
	Data frequency	10 minutes

Figure 5: Features. Source [26]

The author's through testing and experimenting conclude that there are significant evidence that social influence impact abnormal return (stock going up), but more interestingly that they believe there is a better understanding on the price shock from the social attention than the historical data.

3.3 Cloud Computing Providers For Prediction Model Training

It is very important to be able to test and evaluate different ideas/configurations as fast as possible and to do so with complex models and big data sets a lot of computational power is required. Cloud computing[14] can provide with needed hardware as well as some high-level API services for low cost.

A table below shows some of the main features of most popular cloud providers in today's market.

[14] Comparison inspired by these blogs: https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai and https://www.predictiveanalyticstoday.com/compare/microsoft-azure-machine-learning-vs-google-cloud-prediction-api [accessed 22/05/2018]

Cloud Service Provider	Features
Amazon ML[15]	• Very high level. One of the most automated platforms in the market. Good for quick implementation, even when having little Machine Learning skills. Data preprocessing is performed automatically. • Binary classification, multiclass classification and regression predictions. Method is chosen automatically as well. • No unsupervised learning methods • Compatibility for customers that already uses AWS services • Most of the services provided for free
Amazon SageMaker[16]	• Provides with tools for quick model building and deployment • *Linear learner*, for classification and regression • Combines multiple prediction methods to provide better accuracy • *Seq2Seq* - supervised algorithm for predicting sequences, translating sentences • Latent Dirichlet method used for finding categories in documents • Compatibility with TensorFlow • Compatibility for customers that already use AWS services • Most of the services provided for free
Microsoft Azure[17]	• Provides more out-of-the-box algorithms, than other providers in the market (100+ algorithms) • Binary and multiclass classification, regression, recommendation, text analysis. • Very low level. Preprocessing, method selection, validation is done manually. This introduces steeper learning curve, but

[15] Link: https://aws.amazon.com/machine-learning/ [accessed 22/05/2018]

[16] Link: https://aws.amazon.com/sagemaker [accessed 22/05/2018]
[17] Link: https://azure.microsoft.com [accessed 22/05/2018]

	also much more flexibility. • Provides GUI • Growing community; compatibility for Windows users • Compatibility with TensorFlow • Most of the services provided for free
Google Cloud Platform[18]	• Very flexible and low level API • Access to huge datasets • Data centers are allocated world wide • Compatibility with TensorFlow • Most of the services provided for free

Most of the cloud AI providers work well with other providers. So it is encouraged to combine multiple services / providers, if this is what suits the best specific situation.

These services are not the only ones out there in the market. Some more are being developed by startups like - Rackhosting[19], Valohai[20] or other tech giants such as IBM (Watson)[21] or Oracle (Cloud)[22]. They were not taken into this comparison, because of time limitations, also some of them seem to have shallow communities, un-finished documentations or are still in beta version, which are important aspects, when choosing provider for such deadline-sensitive project.

3.4 BITCOIN RELATED NEWS AND DATA SOURCES

There is a massive amount of data across various social media platforms, networks, online forums and news websites. These data sources could potentially contain information related to Bitcoin. In this section, the largest and most common social media networks are presented and briefly discussed what data could be extracted through APIs and their limitations.

Network	Users (millions)	Network	Users (millions)	Network	Users (millions)	Network	Users (millions)

[18] Link: https://cloud.google.com/products [accessed 22/05/2018]
[19] Link: https://www.rackhosting.dk [accessed 22/05/2018]
[20] Link: https://valohai.com/features [accessed 22/05/2018]
[21] Link: https://www.ibm.com/cloud/watson-studio/features [accessed 22/05/2018]
[22] Link: https://cloud.oracle.com/en_US/ai-platform/features [accessed 22/05/2018]

Facebook	2,234	Instagram	813	Reddit	330	Viber	260
YouTube	1,500	Tumblr	794	Twitter	330	Snapchat	255
WhatsApp	1,500	QQ	783	Baidu	300	LINE	203
FB Messenger	1,300	QZone	563	Skype	300	Pinterest	200
WeChat	980	Weibo	392	LinkedIn	260	Telegram	200

Figure 6: Global social networks ranked by number of users[23]

Facebook[24] - At its core a social media, which focus on connecting friends and families with each other, this however also means that anonymity is very unlikely. This allows for any type of people to use their media, which are reflected in the Figure 6 above, of being the social media with the highest number of active users. Companies and restaurants among others are also using it to promote themselves. With no direct specialization, Facebook has the highest diversity of people of all the platforms, which allows for many different opinions of any given topic. Facebook has many similar features as the ones explained below in the other platform, however it can be said that they are like jack of all trades, being able to do everything and not specialized in any of them.

[23]Source of figure: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/#0 [accessed 22/05/2018]
[24] Link: https://www.gcflearnfree.org/facebook101/what-is-facebook/1/ [accessed 22/05/2018]

Reddit[25] - a platform that lets anonymous users submit posts which other users can up vote, down vote or make comments to, however these posts are made underneath sub communities, which are more specialized into a specific topic. This allows people who are interested in the same topic, to be in the same sub community and be able to communicate with each other. In 2016 almost 65% of the users were between 18 and 29, making it more biased towards young adults than teenagers and older people.

Twitter[26] - a platform that allows users to post short posts about anything as a blog. Other users can follow and get any new updates that might arrive from the user or retweet, putting the post on their own twitter for other to read who had followed them. With the way retweeting and following works, it has become a normal place for politics and news to get spread out quickly. With 23% of adults between 18-29 who uses internet, are using twitter, this is the highest demographic of the twitter users.

[25] Link: https://www.brandwatch.com/blog/what-is-reddit-guide/ [accessed 22/05/2018]
[26] Link: https://blog.hootsuite.com/twitter-statistics [accessed 22/05/2018]

Telegram Messenger – is an instant message like WhatsApp where users can join groups and they can talk to each other, however unlike WhatsApp, there is a relatively big community whom specializes in bitcoin or crypto currency. They allow users to join their group and hear what advice they have for them on the upcoming days. Users are able to take advantage of the information and understanding that the creators of the group has, that the other users might be lacking.

Other social media – **WhatsApp** and **FB messenger** among others are private messages, which means they cannot be extracted without security issues. Weibo is like twitter but for Chinese people, written mostly in Chinese. YouTube is a video sharing media for users to post comments, like or dislike the content of the video.

Web sites – To a degree any websites that has information related to bitcoin can be written here, but most important type of sites would be major media outlets and different government sites of different countries for any change on regulations.

Internet forums – Bitcointalk.org is created by one of the founder of bitcoin, and is one of the biggest forums on bitcoin, however many smaller ones exist as well, whom dedicates themselves to news and information on bitcoin. These sites are mostly guaranteed to have users who are interested and understand bitcoin, which mean most posts will be made by educated users on the topic and be related to bitcoin.

Price collection – To obtain prices on bitcoin, sites like Bitcoin exchange exists where it is possible to buy or sell bitcoins. Since every sites have different prices on the bitcoin, it is not possible to guarantee that the biggest site is the one with the cheapest price. According to article[27] the top 3 most recommended is Coinbase, for being the biggest, Gemini Exchange for low fees and Changelly since they have lesser known crypto currencies.

These social platforms and other data sources will be looked upon and analyzed to see which would fit best to gather data to use for predicting bitcoin prices. Although each platform has their own demographic, many of them are overlapping, this could be a problem if multiple platforms were used, giving duplicate data rather than more data. Facebook has the potential of giving the most data, however since the main demographic probably do not know or use Bitcoin, it might be mostly data which does not influence Bitcoin price. Twitter and Reddit on the other hand might have less data to obtain, however more specialized knowledge of the users might lead to some changes in the Bitcoin price. Telegram being the extreme of them all, with the least users, but with the highest expertise.

[27] Article: https://www.techradar.com/news/the-best-bitcoin-exchange [accessed 22/05/2018]

3.5 Data Mining

Data can be acquired from data sources in many ways, therefore it is necessary to investigate what kind of tools and APIs already exist in the market and how they work. This section will present several scrapers and their working principles, APIs and methods of data processing.

3.5.1 Tweepy Crawler

An easy-to-use Python library for accessing the Twitter API.

Tweepy[28] is a very popular open source python library for Twitter API, at the time of this report the library had 145 contributors and over 5000 stars (likes) on their GitHub page. It requires Twitter account and Twitter application, both of which can be easily setup for free of charge. With the help of this library and Twitter API[29], real-time tweets containing necessary keywords as shown in Figure 7 can be returned.

```
stream.filter(track=['btc','bitcoin','xbt', 'cryptocurrency'],
languages=['en'])
```

Figure 7: Code example: filtering keywords from stream. Source: our own

Pros:

- Real-time data
- Supports multiple search keywords
- Free and Open Source
- Well maintained library and active developers
- Easy to Use

Cons:

- Twitter API limits search to last 7 days only.
- Requires Twitter account

[28] Link: https://github.com/tweepy/tweepy [accessed 22/05/2018]
[29] Link: https://developer.twitter.com/en/docs/tweets/filter-realtime/overview [accessed 22/05/2018]

3.5.2 TWINT - TWITTER INTELLIGENCE TOOL

*Advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles **without** using Twitter's API.*

Twint is a lightweight open source crawler, which does not use Twitter API and crawls the search through HTTP. At the time of this report, the library had 6 contributors and 229 stars (likes) on their GitHub page[30]. This library can be used to gather historical data as shown in Figure 8.

```
python twint.py -s bitcoin --count --database 2014-01.db --since 2014-
01-01 --until 2014-02-01
```

Figure 8: Code example: starting Twint crawler for specified period. Own source

Pros:

- Allows to bypass Twitter API limits and gather historical data
- Supports tweet collection for specific periods (max 1 month)
- Free and Open Source
- Does not require Twitter account
- Actively maintained by contributors (weekly updates)

Cons:

- Rapid releases risk breaking the crawlers if updated
- Library contains bugs
- Requires tweaking to allow larger scale crawling

3.5.3 CCXT PYTHON LIBRARY FOR HISTORICAL PRICES

CryptoCurrency eXchange Trading Library[31] (CCXT) provides API to connect and trade with 115 different cryptocurrency exchange markets. CCXT allows for many APIs, however the most important are the public APIs with information from all exchange markets such as:

- Market data
- Price feeds
- Trade history

[30] Link: https://github.com/haccer/twint [accessed 22/05/2018]
[31] Link: https://github.com/ccxt/ccxt [accessed 22/05/2018]

Since they are public, an account or API key of a specific exchange are not necessary to get the data.

To get the market data, or the price of a cryptocurrency a method used is called *fetch_OHLCV*. The method gives back the exchange market's buy and sell price in an array from highest to lowest.

3.5.4 VADER

Valence Aware Dictionary and sEntiment Reasoner (VADER)[32] is a tool used for analyzing sentiments in text-based social media data. Based on dictionary/lexicon and rule-based system it can handle the following:

- Normal negations (Not good)
- Contractions as negative (wasn't good)
- Intensity of punctuation usage (Good!!!)
- Degree modifier (very good, kind of good)
- Many different slang words (sux which means sucks)
- Emoticons (☺ happy smiley)
- Utf-8 encoded emojis
- Acronyms (lol - laughing out loud)

VADER works by taking each word in a sentence and score them with a number between -4 to +4, depending on the degree of negative or positive they are. The scoring works with -4 being the most negative and +4 being the most positive, while 0 is the neutral section. As an example, a word like horrible is mapped to -2.5 while okay is mapped to 0.9 unless it is negated which will then result in a -0.9 (not good) etc. VADER gives 4 results when used: pos, neu, neg, and compound. The first 3 are the ratio of the proportion of the sentence in each of the categories, meaning how many words of the sentence is either positive, neutral, or negative. If summed up, the numbers would give a number close to 1, as a representation of the whole sentence. The compound is each of the words score summed together and then being standardized so it becomes a number between -1 and +1. This makes -1 the most negative sentence while +1 would be the most positive.

[32] Link: https://github.com/cjhutto/vaderSentiment [accessed 22/05/2018]

Example:

```
VADER is smart, handsome, and funny.----------------------------

{'pos': 0.746, 'compound': 0.8316, 'neu': 0.254, 'neg': 0.0}

VADER is smart, handsome, and funny!----------------------------

{'pos': 0.752, 'compound': 0.8439, 'neu': 0.248, 'neg': 0.0}

VADER is very smart, handsome, and funny.-----------------------

{'pos': 0.701, 'compound': 0.8545, 'neu': 0.299, 'neg': 0.0}
```

3.5.5 TF-IDF

A simple way to use a feature is to look at words that are being repeated multiple times in a set of data, where for example the price is dropping, and then note these as negative words. However, because of the simplicity of the system, it also comes with a flaw. Unless manually done, there is no easy way to weight these words. A common and simple weighting scheme used in search engines called TF-IDF [27] (Term frequency-inverse document frequency) can do this. First TF (Term Frequency) measures the frequency of a term (word) that occur in a message, however since messages are never equally long, which means the term can occur many more times in a long message rather than a short message, it is vital that the calculation takes this into consideration and takes the specific term and divides it with the total number of terms in the message.

(TF) frequency of the term = Term / total terms in message

This however only looks at the frequency of the term, to put a weight and understand how important the term is, the IDF (inverse document frequency) is introduced into the equation. As mentioned earlier, words that would usually occur a lot of times in a message such as “a” would unlikely have any impact on the message even though its frequency is high. Inverse document frequency allows for high and low frequency terms to be weighted low while medium frequency terms are being weighted high. This is done by taking the inverse of all the messages and divide it with the amount of message with the specific term in it.

(IDF) the importance of the word = log(total messages / amount of message with term)

Now knowing how often the word appears and how important it is, by multiplying the TF with IDF the weight is given, which can be used as a feature, by either putting it in negative or positive to resemble if the price will be going up or down.

weight = TF * IDF

3.6 Cross validation methods

If all the data is used as training set, it is not possible for the model to evaluate the accuracy of new data, which has not been seen by the model. To solve this, Cross validation methods are used, where a data set is split into training and test subsets, to verify the error rate of the data and from that, the accuracy of the prediction. It also helps find which model and algorithm fits best with the dataset. The methods mentioned below is a few of the commonly known and used cross validation techniques

3.6.1 Holdout Method

Though called a cross validation method, Holdout Method[33] does not actually cross validate, since no iteration of the validation is being done, compare to the others mentioned below. First it takes random selected data points and puts them into two different data subsets, a training and test subset. Although not defined, the data subsets is usually split with the training being around 60-80% and the testing (holdout) with the rest. The training subset is used to train the model without any influence from the testing subset, to get the parameters for the model. These parameters are then used on the testing subset, to see how accurately the parameter and model is. The problem with this method is that the data points are put in either training subset or testing subset randomly, which allows for data point bias. This means that if the model were to be trained with the same dataset, however with different training subset (other data points in the training subset), it would more than likely give a different result. The simplicity of the system makes it fast, however not all of the dataset is being used. Since the test subset data points is only being used to confirm how well the model is performing, it is essentially lost information that could have been used for better training.

3.6.2 K-fold cross validation

Unlike Holdout not using valuable data, K-fold cross validation[34] (K-fold CV) uses all the data for training and testing in an iterative way. K-fold CV takes the dataset and splits it into k subsets

[33] Article: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html [accessed 22/05/2018]
[34] Article: https://magoosh.com/data-science/k-fold-cross-validation [accessed 22/05/2018]

and each of the subsets are used individually as test subset while being trained on the rest of the k subsets.

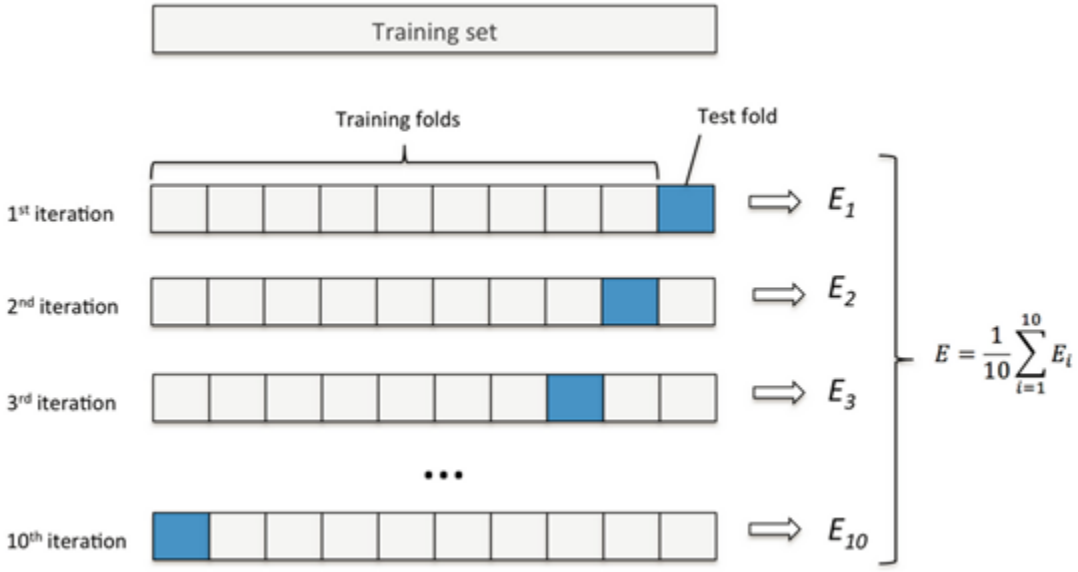

Figure 9: 10-Fold Cross Validation[35]

As seen in the Figure 9 above, in a 10-fold Cross Validation, 10 iterations of training are being made. First the first 9-fold is used for training and the 10th are used for testing. Then the model gets memory wiped out, and does a new iteration of the same process, however this time with the 9th fold being the test and the 10th fold being included in the training. When all the iterations have been done, the average of all the results are taken, and this is the accuracy of the model. Unlike holdout, the result is always the same, no matter how many times it is being done, if the data set is the same. K-fold Cross Validation also allows for detection of overfitting and noise. This means that by simply removing or adding features and then re-run the validation, allows for a way to see if the model became more accurate or less, just by averaging all the results and see if the number are higher or lower than before.

3.6.3 LEAVE ONE OUT CROSS VALIDATION

This method is like k-fold however, instead of splitting the dataset in k subsets, it is instead split into subsets of one data point. This means that with a large dataset, the amount of iterations can be overwhelmingly large, which requires large amount of computational power. Though Leave on out CV is less bias than k-fold it comes with the cost of higher variance, since all the

[35] Figure source: http://karlrosaen.com/ml/learning-log/2016-06-20 [accessed 22/05/2018]

data points except for one, is being used in every iteration. This means that because each iteration of training subsets has a high degree of overlapping (using the same points), making the error rate have a high correlation with each other.

4 ANALYSIS

With a better understanding of what tools, method and technology that exist from the chapter 3, four main problems were identified represented as sections in this chapter that had to be discussed to solve the research question and the sub questions (section 1.1). First is where to collect data, as mentioned in the state of the art section 3.4, and what kind of data needs to be collected. Secondly when the data is collected, how to preprocess the data so it can be used for the model and algorithm, this is important since some machines needs data standardization or normalization to be able to understand and interpret data. Third and also an important part of the system is to find features for the model from the data collected. Lastly build a model that uses these features to train and predict future outcome as shown in Figure 10. These are all summed up in a list of requirements in the end of the chapter, that makes the baseline of prototype that must be implemented.

Machine Learning generally does well with large or very large datasets, therefore it is necessary to investigate how to collect large amounts for data from variety of data sources. Key areas for data collection, as introduced in 3.4:

- News articles
- Social media platforms
- Internet forums
- Private networks
- Currency exchange platforms

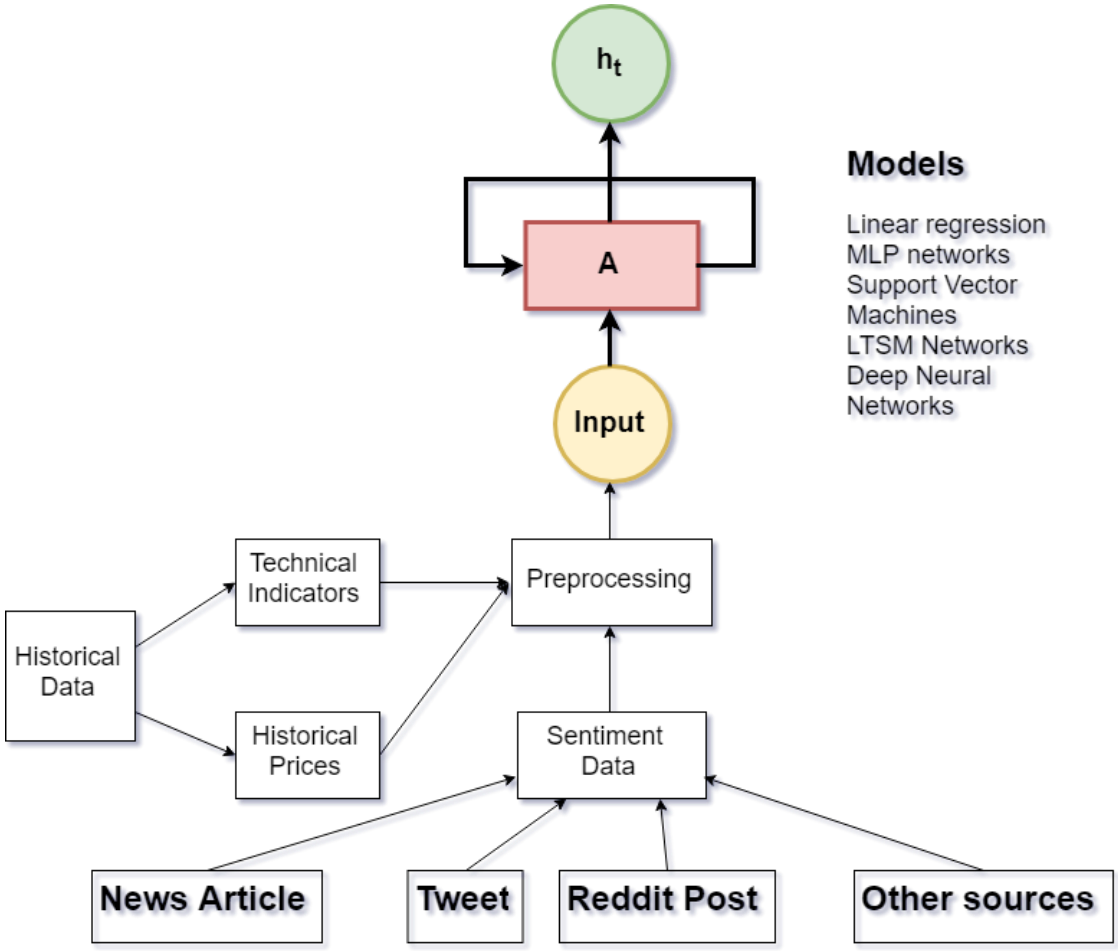

Figure 10: System Overview. Source: our own

4.1 DATA WHICH COULD AFFECT BITCOIN PRICES

One of the most difficult tasks is to understand, during the process of this research, what information is just noise (makes more harm to the prediction than if it left out) and what information has a lot of impact on the bitcoin price. With the research being focused around sentiment data, the first part is about psychology of humanity, why sentiment score might be valuable information rather than noise and where to gather such data from. This information can be gathered from different social network platforms, but which depends on the people that the sentiment score is aiming for. Sentiment data can also be gathered from global news, that not necessary has to affect the whole world, but has a big enough impact for people to notice. Lastly is the technical data, such as price of the bitcoin or how many were sold etc.

In the world of economics, they believe that the psychology affects the investors' decision also known as Animal Spirit[36]. This means that negative and positive emotions of the investors can have influence on the price. The research paper [25] confirms this with happy investors is more likely to take more risk making the bitcoin increase, because of them being more likely to buy it. Related but not completely the same is the media coverage of the bitcoin, the investors can be more comfortable with their bitcoin and that will result in the price increasing, however the opposite is also true with bad publicity, they might sell their bitcoins in fear of it dropping, resulting in it dropping. Positive and negative media news and personal emotion is related however not the same, as they are data collected different ways (read more in 3.4). Another way to look at it, is that a person has their emotions however it can be affected or changed based on good or bad news. However with the internet today, the news of people's feelings can be spread out extremely quickly through social networks (read more in 3.4) such as Twitter, Reddit, Facebook etc. This means that not only does news on a national or global scale have a big effect on people, but friends or celebrity can have big impact on their emotions as well.

Like most other products, any change in the governments law is directly influencing changes to the product's price. Nevertheless, since cryptocurrency is still relatively new, and the governments around the world are still figuring out how to handle it, new rules are being made to take into consideration of it. One of these recently changes as seen on the Figure 11, is the "Bitcoin legal tender in Japan", which is a law that allows bitcoin to be legally used in Japan. A high increase in the price is seen right after the legislation is accepted. The other way is also true, when China banned ICOs, which means that all cryptocurrency got strictly banned from the country, the price dropped. Looking at this and strictly saying that the legislative changes are the only reason for the increase and decrease of the price, would be foolish, however they were most likely one of the core reasons, since people were afraid (negative emotion) of what it would mean for the bitcoin.

[36] Article: https://www.economist.com/media/pdf/animal-spirits-akerloff-e.pdf [accessed 22/05/2018]

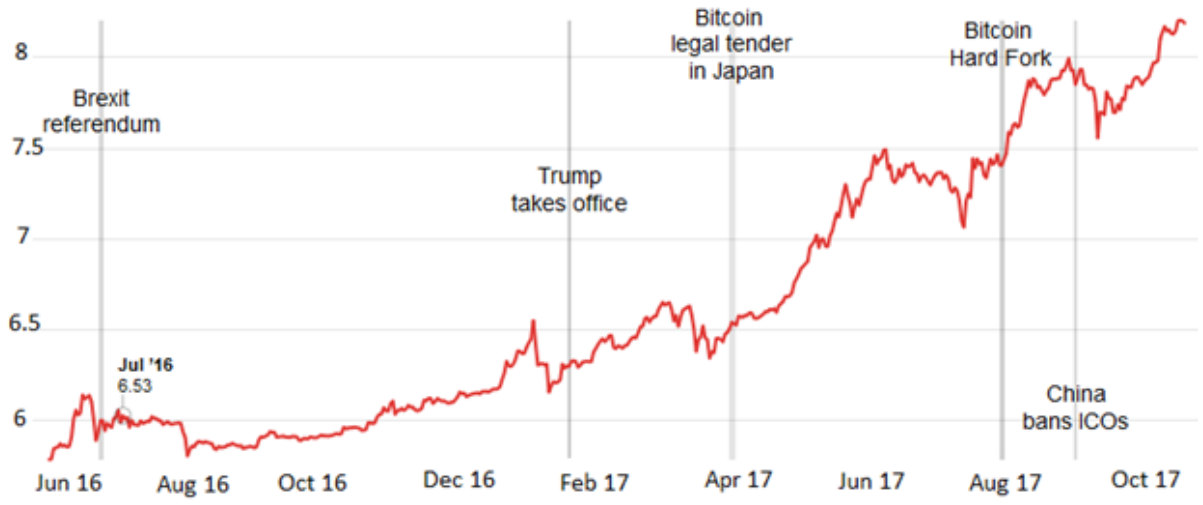

Figure 11: Events driving the price of bitcoin[37]

With the way cryptocurrency works, many political situations can indirectly affect the price as well. One of the main points of bitcoin or any cryptocurrency for that matter is that it is not centralized in one place, like money is in a bank. This allows for easy transfer across countries in case something happens in their country. Common sense would say that with the situation in Greece in 2015 people trying to leave, the price of bitcoin would increase, however unconventionally that did not happen[38]. However, in the event of Brexit there was an increase of bitcoin price of around 65%[39]. Though Brexit might not be the sole reason for the increase it is something worth noticing.

4.1.1 DATA SOURCES WHICH COULD AFFECT BITCOIN PRICES

As mentioned earlier the state of mind of the people can affect the price, and to get a better understanding on how it is, can be found in many ways. Questionnaire, interview or polling among others can reveal the mood of the crowd in general, however they take time and is not a good source to get a lot of data from, nor are they reliable, there is no way to know if they are answering it honestly. The research paper [24] took another approach, by looking into social media that has a lot of information going in and out. Many social media exists(section 3.4), and they all have each their own strength and weaknesses (read more in 3.4). In a world with infinite resources and time, collecting data from all of the social media, to later see which of the

[37] Figure source: : http://theconversation.com/four-factors-driving-the-price-of-bitcoin-87244 [accessed 22/05/2018]
[38] Article: http://money.cnn.com/2015/06/29/technology/greece-bitcoin/index.html [accessed 22/05/2018]
[39] Link: https://www.coingecko.com/en/price_charts/bitcoin/gbp [accessed 22/05/2018]

features does the best, would likely be the best idea, however in this case several was looked upon and reviewed (section 3.4), to find one platform to extract data from, because of limited resources. Facebook is the most used platform, however it comes with a price of being used by the general population of the world, and they are not the same people who buys bitcoin[40]. This makes a lot of the data extracted not very useful, since sentiment analysis on people who do not buy bitcoin would unlikely yield anything. Instant message social media such as WhatsApp and WeChat comes with a barrier of usually being small groups writing together and a need for invites to be able to enter the group, which means that most of data are encrypted and not accessible for extraction. Lastly are the more specialized social media such as Reddit and Telegram Messenger. They have areas where they are more specialized in cryptocurrency, so data gathered are more likely to be from people who understand or uses some cryptocurrency. This however also means, that the data gathered can be more biased towards the specialized people and data gathered will be primary only focused around cryptocurrency, so news or information that might have an indirect effect on the bitcoin, might not be gathered from this source such as a new president being picked in the US[41]. A middle road between specialized and general usage become the choice of the social media, called Twitter. It gives data from every areas, with specific key search, called hashtag it is possible to filter tweets (messages) for extraction, it is also becoming one of the most popular platform to announce news and political agenda[42]. This also allows gathering of news and government legislation changes on the same platform.

Government legislation changes or any big news would likely be mirrored in the social network, by having people write about it, their information would likely be based on their own opinion and therefore be biased, however that would give their sentiment on the news which is also useful. To circumvent the problem, getting the data elsewhere on top of twitter would be the best solution. This is needed in situation where government news has big impact on bitcoin were there are a chance of a big spike in the price, such as a ban of bitcoin in some country. However, since there is no global government data site, this would have to be taken from a 3rd party site which risk in getting biased information or extracted individually from individual countries government webpages.

---

[40] Article: http://time.com/money/5116904/who-is-buying-bitcoin [accessed 22/05/2018]
[41] Link: https://www.coindesk.com/bitcoin-price-donald-trump-president [accessed 22/05/2018]
[42] Article: https://www.theguardian.com/technology/2016/jul/31/trash-talk-how-twitter-is-shaping-the-new-politics

[accessed 22/05/2018]

Another set of data is needed to make a more precise prediction and that is the historical price data of bitcoin. This information can be used for two things, the first is that without knowing the current price, it is impossible to know how much a 2% increase is. The 2nd thing is the technical indicators such as the price cap and standard deviation

4.1.2 DATA COLLECTION OF TWEETS AND BITCOIN PRICES

Data collection from websites and news articles is done by distributed web crawlers(read more in 3.5). The search engines have been using these tools to crawl the web for hyperlinks and data. Many tools are available as open source software. In case of website Reddit, their API can be used to extract data related to bitcoin.

Social media platforms are usually not crawled by search engines. To collect data from these networks in many cases it is necessary to use their APIs, which means there probably are some rate limitations for fetching too much data from the past. To collect public data from Facebook it is necessary develop a custom application. To collect data from Twitter two different tools have been identified: *Tweepy(3.5.1)* and *Twint(3.5.2)*. These tools will be used to collect historical and real time tweets from Twitter.

Internet forums are usually better indexed by web crawlers compared to social media platforms, however to collect more recent data it is necessary to create an account and customize the web crawler.

Private networks like Telegram requires specialized crawlers to collect data. Telegram chats are commonly used by cryptocurrency traders to discuss changes and predictions in the market, however not every group is open for public users.

To train the model it is necessary to collect historical data for prices and any other data which could have impacted the price of bitcoins. In addition, to make a useful prediction it is necessary to collect data real-time or as close as possible to real-time.

Currency exchange platforms is gathered into one API by using CCXT library (section 3.5.3). It allows for gathering of market data, price feeds and trade history among other things for better prediction. Since bitcoin is not being sold for the same price in all of the exchange markets, CCXT can see which of them currently has the lowest price for bitcoin.

It is not always clear when to collect the data and should it be updated. In case of social media networks it is possible to get early estimate about possible reach of tweet or message based on

how many people follows the sender. However, some messages are shared or retweeted by other users and therefore exponentially increases the reach of the original message.

In addition, some data provided by the API could be within incorrect time frame. Example: *Tweet has been posted in 2014 and crawled only in 2018 and shows that user has 2000 followers*. There is no guarantee and is unlikely that user had same amount of followers several years ago. Using this data for training our model might produce incorrect results. If based on test results follower count is shown to be important feature, one of the solution could be to predict the growth of user's followers (read more in [25]). Social media analytic platform Socialblade[43] estimates the growth of social media channel with linear function.

Many news sources provides some feedback about how well the article was received by the public, in terms of social media similar statistics can be visible on the post. Likes, dislikes, shares, views, comments and other features could be used as a way to identify the reach of the post. The main issue is that this data is not available at the time of posting and is accumulated over time.

Based on information above, it is best to collect data done within a specific time frame. Historical data has to be collected for required periods and features like follower count should be adjusted or removed from dataset.

The goal is to be able to predict if the bitcoin price change would increase or decrease and by how much, however this goal can be accomplished in many ways. How often does it need to be able to predict? Whether it predicts every minute, 10 minutes or hourly will all decide the time frame the data set will be collected in. If predictions are 10 minutes, then all social media in a 10 minute timeframe will be used to predict, however if it is 1 hour, then 1 hour worth of data will be collected to predict etc. In all, how often to predict would be based on the amount of data gathered in a day, the more data gathered the smaller timeframe, however according to they concluded [25] that the shorter the frame, the more influence sentiment value had which is something to take into consideration as well. As shown in the Figure 12 below, in barely 40 minutes the price changes more than 100USD. However news and social network messages takes time to get around. A longer time frame would help on that, since messages and news has to be written and when it reaches people, they also have to open and read it.

[43] Link: https://socialblade.com [accessed 22/05/2018]

For this reason it is proposed that the period predictor should be around 2 hours or maybe even 1 day, however the final decision for this will be based on how well the testing phase goes (more on this in 4.4).

Figure 12: Open and Close prices[44]

As concluded in this section and backed by other researchers (section 3.2), all social media networks have their own strong points, however, it seems that Twitter offers great influence on cryptocurrency community as well as relatively easy approach to extract data, as discussed in problem formulation (1.1). Twitter will be the choice of social network for the first prototype. Following sections explain more in-depth how can twitter data be used for such price prediction problem.

4.2 PREPROCESSING COLLECTED DATA

There is no doubt that one of the most important steps in machine learning is feature extraction of given dataset, however this cannot be done before data is prepared. This section will

[44] Figure source: https://www.cryptocompare.com/coins/btc/overview/USD [accessed 22/05/2018]

describe steps and tools for cleaning up the data as well as explanation of what exactly is clean data in this specific research.

4.2.1 CLOUD API SERVICES

This section provides brief comparison of high-level cloud API services for sentiment extraction, which could be used in this project for twitter feed analysis. Since natural language processing (problems like language translation, sentiment analysis, spelling correction and etc.) are not main focus of this research paper, using existing cloud services will save expensive development time.

Here is a feature comparison Figure 13:

	Amazon	Microsoft	Google
Key phrase Extraction	✓	✓	✓
Sentiment Analysis	✓	✓	✓
Entity Recognition	✓	✓	✓
Spell check		✓	
Word breaking		✓	
Language Detection	✓	✓	✓
Language Translation	✓ (6 languages)	✓ (62 languages)	✓ (> 100)
Topic Modeling (Collection of documents)	✓		✓
Multiple language support (no need to translate before analyzing)	✓	✓	✓

Figure 13: Feature comparison ***Source: our own***

Most of the services are available for free for at least 12 months, with some quota limitations. Since there can be a lot of tweet data, these limitations should be taken into consideration.

As can be seen from the Figure 13 above, Microsoft has advantage in amount of features. They also provide quite low-level APIs, which allows developer to create customized solutions. Google benefits from its huge datasets, and that is seen in the amount of supported languages.

However, before data can be optimized with external API tools, it must be prepared and cleaned up, which will be covered in following section.

4.2.2 Preparing Twitter dataset

To start with, usual steps to preprocess any dataset before feature extraction and learning can begin are defined. These steps were discovered in papers, analyzed in section 3.2.

1. Analyze structure of dataset (columns, fields and etc.)
2. Name columns
3. Drop unnecessary columns
4. Remove missing or corrupted values

And here are processing steps that are more specific for social media text-related datasets:

5. Decode HTML encoding (& " etc...)
6. Convert to lower case
7. Remove mentions (regex)
8. Remove URLs (regex)
9. Remove # symbol, but leave the text (regex)
10. Remove numbers

Let's look at the format of current dataset, to see which processing steps could be applied and which are not so relevant. Dataset contains 15 columns, but only few will be used in feature extraction step:

Keep	Drop
# date	# id of a tweet
# time (UTC)	# timezone
# tweet (text)	# userId
# followers_count (number of author's followers)	# user (username)
# listed_count	# replies (number of replies)

(number of public lists, in which author is a member)	
	# likes (number of likes)
	# retweets (number of retweets)
	# hashtags
	# friends_count (number of accounts followed by the author)
	# favourites_count (number of tweets author has liked)
	# statuses_count (number of tweets author has posted)

Figure 14: Columns transformation. Source: own source

In the list above (Figure 14), column that is marked red, will be removed at the step 3. Replies, likes and retweets are removed because tweets were collected live, and the values of those 3 fields are zeros. However more columns might be included in later iterations for extracting more interesting features.

Tweet text will be used for sentiment analysis, while followers count and listed count will be used to evaluate how much the tweet is potentially spread across Twitter platform.

Steps 5, 6, 7 and 8 are quite self-explanatory and should improve training of the model. Regarding last two processing steps: Text that follows hashtag might show some predictive power, while symbol "#" is not necessary. Numbers do not reflect any context on their own, and will be removed from tweets.

After preprocessing steps were established on the Twitter data, python library - Wordcloud[45] was put to test to visualize frequency of terms in dataset. This tool is nice to get a visual overview, although basic term frequency might not be so sufficient as a feature in algorithm training. More suitable TF-IDF approach is discussed in 3.5.5.

Word cloud was evaluated on two different size datasets – 10000 tweets (Figure 15) and 100000 tweets (Figure 16) randomly selected messages. Tweets that contain hashtag *#Bitcoin* were collected from April 15th to 30th 2018. Below are the results:

[45] Link: https://github.com/amueller/word_cloud [accessed 22/05/2018]

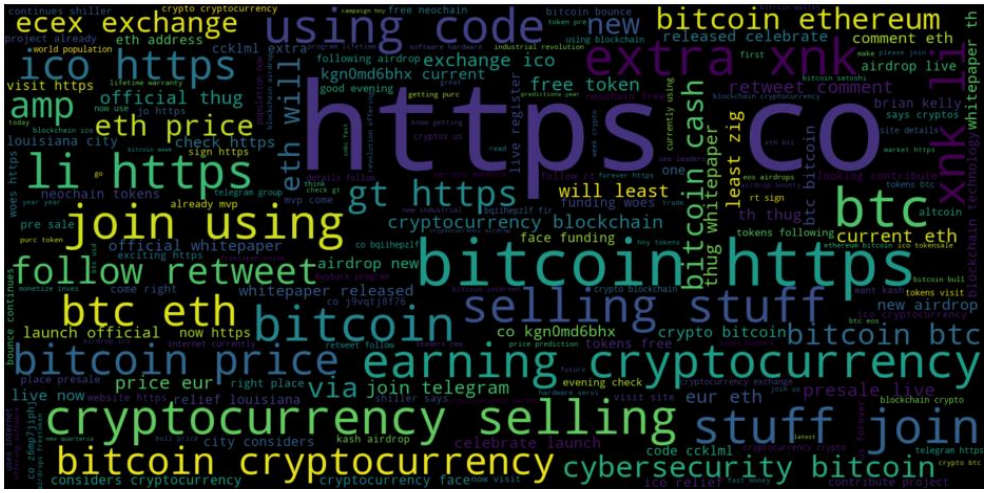

Figure 15: 10000 Tweets Source: our own

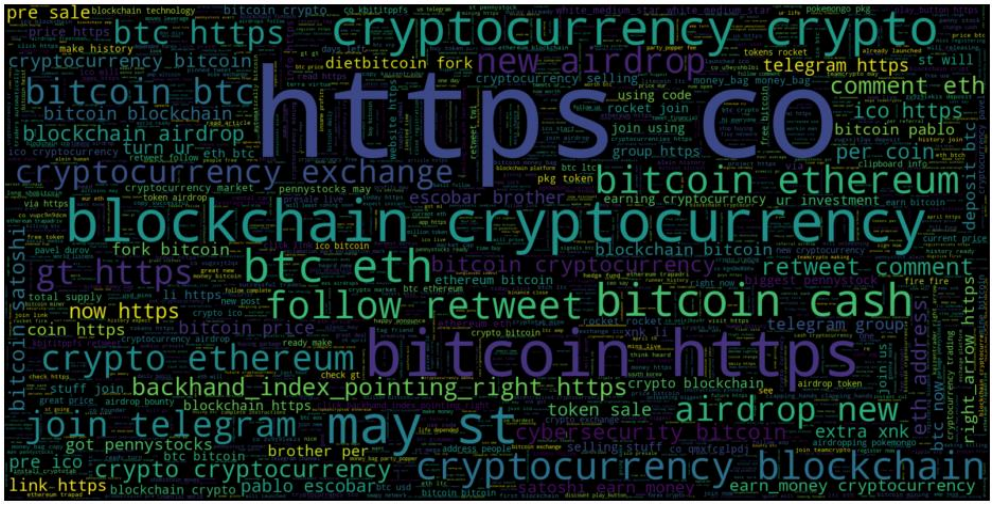

Figure 16: 100000 Tweets Source: our own

This gives a quick insight into most frequent 2000 keywords of the dataset. It also helps to detect some stop words that could have been missed by the library, but should be removed before feature extraction step (e.g. https, now and etc.).

4.2.3 Preparing Historical Price dataset

As discussed in other similar works [1], another dataset that must be preprocessed before feature extraction is Bitcoin's historical OHLCV price data (Open High Low Close Volume, described in 3.4 and 3.5.3). Here are the raw header columns:

# Timestamp	# Open	# High	# Low	# Close	# Volume

Price data for prototype training is collected over period *01/04/18 - 05/05/18* and contains prices for every minute. 49728 rows in total.

There are two main uses of this data:

- Use it to extract target values, against which the model is trained. This is the value that model will be predicting, after it is trained. E.g. Closing price at the end of selected timeframe, or increase / decrease percentage over the timeframe.
- Use the data to extract more interesting features, which could supplement sentiment analysis data with some technical insights, e.g. volume or market cap change over timeframe (read more in 4.3).

Since data comes in an organized structure (OHLCV) it is easy to clean it up. Here are the preparation steps:

1. # Timestamp column entries are in *timestamp* format, however it is much easier to work visually with *datetime* format (*year/month/day hour:minute:seconds*) and it is still suitable for python scripts. Transformation must be done and column renamed to # Datetime.
2. Sort data by date
3. Filter out rows by date. Only prices in the period when twitter data was collected are relevant (15/04 - 30/05).

More specific transformations for this use case will be made in feature extraction process.

4.2.4 TOOLS FOR PROCESSING RAW TWEET AND PRICE DATA

This section describes how some of the tools found in sections 3.2 and 3.3 can be used for data processing. Implementation details can be found in chapter 5.

From the review in section 3.3, seems like Microsoft Azure infrastructure is a good fit for this project, given its free tier for beginners. It not only provides with possibilities to train models on cloud GPUs, but also provides platforms like *Portal* or *Workbench* to easily import data and work with it visually.

In particular - *Workbench*. It is a great tool to process datasets with smart row and column operations. Both datasets (tweets and prices) will be managed through *Workbench* with help of some custom python scripts. All 4 of the first processing steps, described above, can be done on this platform.

Below are the examples[46] of unprocessed datasets loaded in *Workbench:*

Columns: 6
Rows: 49728

	Column1	Column2	Column3	Column4	Column5	Column6
49	1524221460000	8508	8520.98	8505	8518	58.402375
50	1524221520000	8515.01	8522	8506	8512.63	83.361435
51	1524221580000	8508.96	8514.58	8506.74	8510	30.350127
52	1524221640000	8513.99	8514.96	8506	8508	44.413834
53	1524221700000	8508	8510	8485	8489.99	103.318667
54	1524221760000	8485	8495.96	8479	8486.95	69.534713

Figure 17: Unprocessed OHLCV data. Source: our own

Columns: 16
Rows: 10000

Next

	id	userid	date	time	ti...	user	tweet	replies	likes	retweets	hashtags	followers...
1	9855681...	963703738...	2018-04-15	17:18:55		illner_a	solid https://t.co/Gd6h8mjNIP	0	0	0		41
2	9855681...	3160738079	2018-04-15	17:18:56		slydesexy75	RT @PlanetZiggurat: Ecex Exchange IC...	0	0	0		176
3	9855681...	1942711694	2018-04-15	17:18:56	UTC	kushan294	RT @miBoodleITO: #miBoodle utilises t...	0	0	0	#miBoodle,...	5484
4	9855681...	2792071502	2018-04-15	17:18:57		primero25	:boom::boom::boom::boom::boom:	0	0	0	#airdrop,#...	12
5	9855687...	952081864...	2018-04-15	17:21:27		PuleApril1	RT @Crypto_goat_: $NEO Could Becom...	0	0	0	#Ethereum...	250

Figure 18: Unprocessed Twitter data sample. Source: our own[47]

To work with dataframes, strings and numbers in python scripts - some of the most popular libraries will be used:

- *pandas*[48], data analysis library, that can transform dataframes, add new columns and etc.
- Python's *re* module, for regular expression operations.
- *matplotlib* is used for plotting graphs

After applying all of the processing steps on the datasets, *Workbench* data looks like this:

[46] All original size images can be found in appendix
[47] Appendix A. full size
[48] Link: https://pandas.pydata.org [accessed 22/05/2018]

Columns: 4
Rows: 10000

	Datetime	tweet	followers_count	listed_count
1	2018-04-15 1...	solid	41	1
2	2018-04-15 1...	ecex exchange ico	176	0
3	2018-04-15 1...	miboodle utilises the erc token amp our token...	5484	16
4	2018-04-15 1...	:boom::boom::boom::boom::boom join ethere...	12	3
5	2018-04-15 1...	neo could become the ultimate ethereum cryp...	250	0

Figure 19: Processed Twitter data sample. Source: our own

Columns: 6
Rows: 10000

	Date	Open	High	Low	Close	Volume
1	2018-04-01 00:00:00	6922	6969	6922	6943.62	50.773394
2	2018-04-01 00:01:00	6956.75	6969	6943.63	6965	20.457221
3	2018-04-01 00:02:00	6968.79	6996	6965	6996	34.722016
4	2018-04-01 00:03:00	6996	7013.84	6985.47	6997.15	75.296711
5	2018-04-01 00:04:00	6986.61	7005	6983.12	6990	40.633702

Figure 20: Processed OHLCV data. Source: our own

At first processing steps were run on small data samples, for fast testing. After whole datasets got through the steps it is ready for feature extraction process. Experimentation was inspired by these blog posts[49][50]. Implementation details can be found in chapter 5.

Now take a closer look into the raw price and tweets datasets. Below is a Bitcoin price chart (Figure 21) plotted from current dataset (each tick in x-axis is one hour). Price increase from 17/04 to 25/04 is ~ 22%, which is good indicator of high volatility in such short period.

[49] Blog: https://towardsdatascience.com/create-a-model-to-predict-house-prices-using-python-d34fe8fad88f [accessed 22/05/2018]
[50] Article: https://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python-with-keras [accessed 22/05/2018]

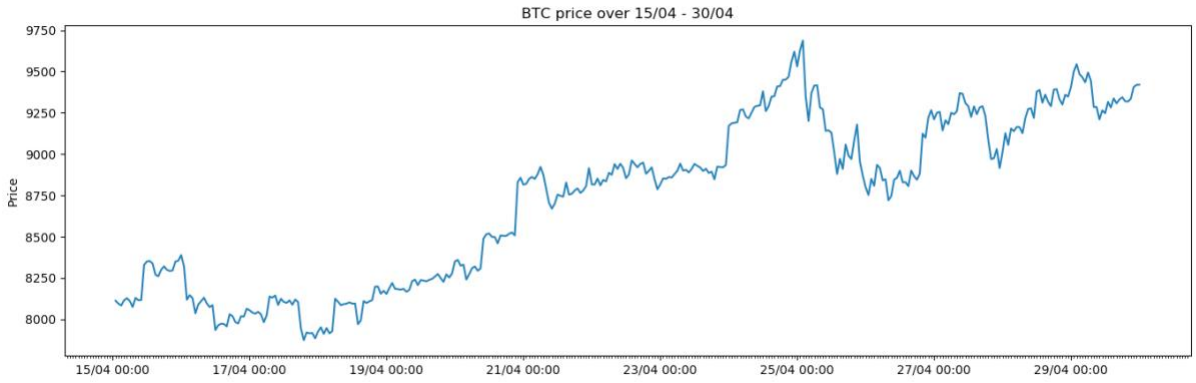

Figure 21: Bitcoin price chart. Source: our own[51]

Below is Tweet volume chart (Figure 22) in the same period. One tick in x-axis also mean one hour and y-axis represents number of tweets per hour. What is noticeable at first, that there is very strong repeating pattern over a single day, most likely related to people sleep time, habits and free time. But more importantly, it also seems that there is some correlation between price and tweet volume over two weeks period. This could be one of the extra features for the learning algorithm.

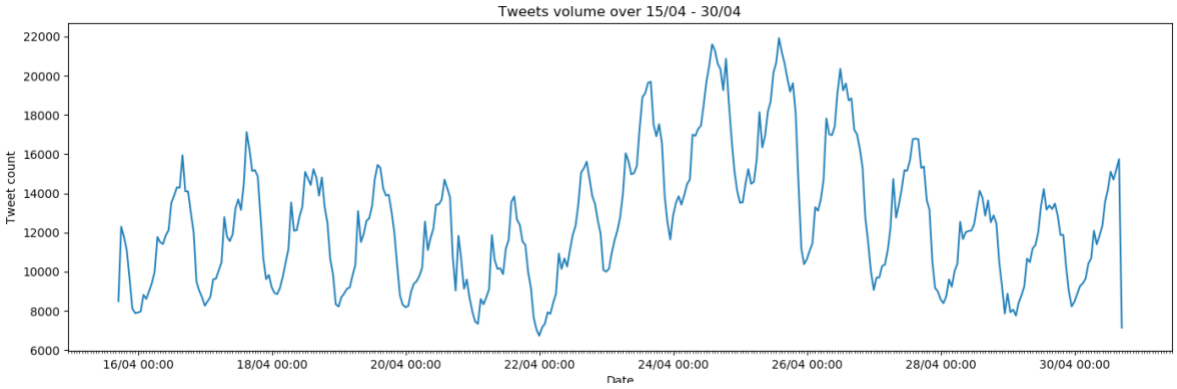

Figure 22: Tweet volume chart. Source: our own[52]

[51] Appendix B. full size
[52] Appendix C. full size

4.3 EXTRACT FEATURES

This chapter will test and compare different NLP services, which had the best number on sentiment score in relation to Bitcoin. There will also be a section about how and what features are being extracted from the preprocessed data.

4.3.1 SENTIMENT ANALYSIS APPROACH

This section describes results of a brief testing that was made with popular NLP services: Microsoft Azure, Google Cloud, Amazon and VADER algorithm.

Testing was carried out in these steps:

- Choose three tweets with hashtag #Bitcoin, that seem to be subjective about the topic and complex to semantically understand their mood.
- Pass these tweets to each of the NLP solutions.
- Evaluate sentiment scores. Since format is different from each of the providers, they were normalized manually for comparison.

Below are the sentences used for testing. These samples were selected from first tweets that appeared when searching with *#Bitcoin,* they have either positive or negative mood towards bitcoin technology, so assumption is made that they could be influential for the price too:

1. "*I know someone well-connected to the small universe of traders who trade large blocks of $BTC #**bitcoin** off the exchanges. He told me $25 billion to buy on the books today. This is institutional money. You see it as 3X the average daily volume. I see it as 20% of the floating supply.*" Mood assumption – Weak positive.
2. "*The problem with fiat money is that it rewards the minority that can handle money, but fools the generation that has worked and saved money. #**Bitcoin***" Mood assumption – Positive.
3. "*DeCoin is utilizing a Proof of Stake (POS) consensus algorithm. Energy saving is one of the many advantages compared to the POW algorithm. Read more here: http://bit.ly/2vR5c2f".* Mood assumption – Weak negative.

Each tweet was ran through selected services, recording they're scores. Values were manually normalized in range between 0 and 1 for easier comparison.

Microsoft Azure Text Analytics API[53]

[53] Link: https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

Run	Positive	Negative	Neutral	Combined
#1		0.48		
#2		0.85		
#3			1	

Google Cloud Natural Language[54]

Run	Positive	Negative	Neutral	Combined
#1	0.3			
#2		0.4		
#3		0.7		

Amazon Comprehend[55]

Run	Positive	Negative	Neutral	Mixed
#1	0.08	0.17	0.73	0.02
#2	0.15	0.31	0.22	0.32
#3	0.24	0.01	0.74	0.05

VADER[56]

[54] Link: https://cloud.google.com/natural-language/

[55] Link: https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html

[56] Link: https://github.com/cjhutto/vaderSentiment

Run	Positive	Negative	Neutral	Combined From 1 (Very positive) to - 1 (Very negative)
#1	0.0	0.04	0.96	- 0.23
#2	0.17	0.18	0.65	- 0.1
#3	0.16	0.0	0.84	0.56

It can be seen that results are very controversial, and more rigorous testing with more tweets is needed to fairly evaluate the performance of the services.

Price of these services is another important aspect, since each tweet from the dataset must be passed to sentiment analysis. And most of the services are charging per request or per number of characters and running through millions of tweets might end up costing a fortune.

Rough estimates of sentiment analysis costs are provided below.

# of tweets	Azure	Amazon	Google	VADER
3 million	**2500$** (Provides 200$ worth of services for first month)	**300$** (or 50 thousand tweets per month in a free tier)	**1500$**	**FREE**

Since sentiment scores did not show any major differences between the services and most cost efficient solution being VADER, this approach will be used for the prototype. Although paid services should be consider for production version, as they provides important features such as topic modelling for all of the documents in the dataset.

Below Figure 23 plots VADER compound scores of the twitter dataset (15/04 - 30/04). Green spot represents compound mean of positive tweets in 60 minutes. Red spot - negative tweets. Since negative tweets has compound in range from 0 to -1, absolute values were taken for

better comparison.

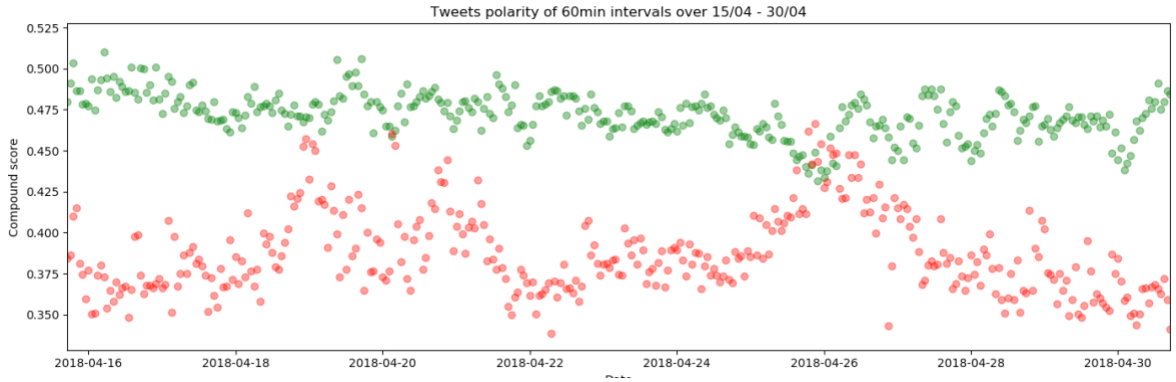

Figure 23: Tweets polarity of 60 min intervals over 15/4-30/04 Source: our own[57]

While looking at the BTC price chart in Figure 21, even though price was climbing almost whole 2 weeks period, seems there is no visible correlation between the price and positive tweets shown in Figure 23. However, it is very clear that positive tweets have higher positive compound score, than negative compound of negative tweets. Also, number of positive and negative tweets is quite expressive:

Positive (compound > 0) - **2219668 tweets**

Negative (compound < 0) - **520568 tweets**

Neutral (compound == 0) - **1838862 tweets**

On other hand neither positive or negative tweets have a very strong compound score (> 0.6), which shows a lot of neutral mood in the period.

If positive and negative tweets compound mean, min, max values are calculated for a selected timeframe without first separating the tweets, results are much lesser expressive, and especially influenced by larger amount of positive tweets.

4.3.2 Feature extraction from processed data

This section elaborates on using sentiment analysis to extract features for price predictions, as well as what difficulties might arise. Later on some of the most important features are visualized using plots and interpreted. This helps to see if there are any visible correlations between feature and price and should any results be expected.

[57] Appendix D. full size

It has already been established that positive and negative emotions affect people, which in the end affect their choices. Sentiment analysis allows for a feature extraction which can be used to understand how the person who wrote the message feels about the topic or just in general. As many other researchers also has concluded (section 3.2) cryptocurrency has a tendency of following the same path as the mental state of people. However, there are few problems that can occur by using sentiment analysis. First and foremost is the tool used for analyzing, has to be able to not only understand what is negative and positive messages, but also to what degree of positive/negative it is. Reason for this is because if a message that might say “I had a good day”, which is obviously positive are compared to a message saying “I had an amazing day” knowing which one has more positivity is important. To add on this, the following feature, which alone could be a feature, combined with sentiment would create better results [25]. The argument to back it up is, if a person has millions of followers, they not only get their message read by a lot of people, but if the message is negative or positive, that emotion or sentiment is being pushed to the ones reading it. This means that a weighting system should be implemented based on the number of followers.

It is difficult to say if a post count as a feature can improve predicting performance. The idea is that the more posts a user has made, the more active and more likely people are to read it. However, that information can already be established from the follower feature. This can be tested and removed if result was not improved by the feature.

Likes by itself can be a feature, however if combined with the sentiment, it is possible to weight the feature with the sentiment, since if heavy negative or positive message are liked, the one who liked it probably feels the same way, and is in the same state of mind.

Re-tweets is similar to followers, in the sense that it can be used to measure how far the spread of the message is. However the difference is if a user has 100 followers, it is highly likely that those 100 read the message, but if one of the followers re-tweeted the message, his followers more than likely have read it, increasing the amount of people reading it possibly exponentially faster than followers. Another researcher [26] used followers in a different way, they tried to calculate the spread of the information rather than the spread of sentiment, however the idea and way to use the feature is relatable if not the same.

Current price or price change does not only have to be the result for training the model but could also be features that show a correlation of the growth of the bitcoin over time.

Following part of the section presents preliminary list of features that will be extracted from the datasets and used for training and testing. Features are derived from combination of tweet followers count and VADER sentiment scores. Couple features are extracted from BTC price dataset too. Some of the important features that were assumed to have relationship with price are plotted against price changes, to see if there are any noticeable correlations. Important to notice that these features are timeframe aggregates of single tweet features. Every column of features are aggregated applying some simple operation e.g. taking a mean or sum of all feature values in that period.

List of features

- TFIDF word dictionary
- Followers (count)
- Likes (count)
- Re-tweet (count)
- Followers (count)
- VADER compound (mean & sum)
- VADER positive/negative/neutral (mean & sum)
- Volume[58] (sum)
- Standard deviation (std of Closing prices in the period)
- Market cap change[59]
- Followers compound[60] (sum)
- Followers positive/negative/neutral[61] (sum)

Many of these features are extracted using different operations on the same qualities, which could just introduce more noise than help training. Different configurations of features will be tested to find best fit.

Figure 24 and Figure 25 below demonstrates relationships between 60 minutes price change and two most influential features (based on researches in 3.2):

- *compound_mean*
- *followers_compound*

[58] Volume that Bitcoin was traded during period.
[59] Change in total value of bitcoins in the market (in selected timeframe)
[60] Followers compound for each tweet: VADER compound * followers count
[61] Followers positive/negative/neutral for each tweet: VADER positive score * followers count

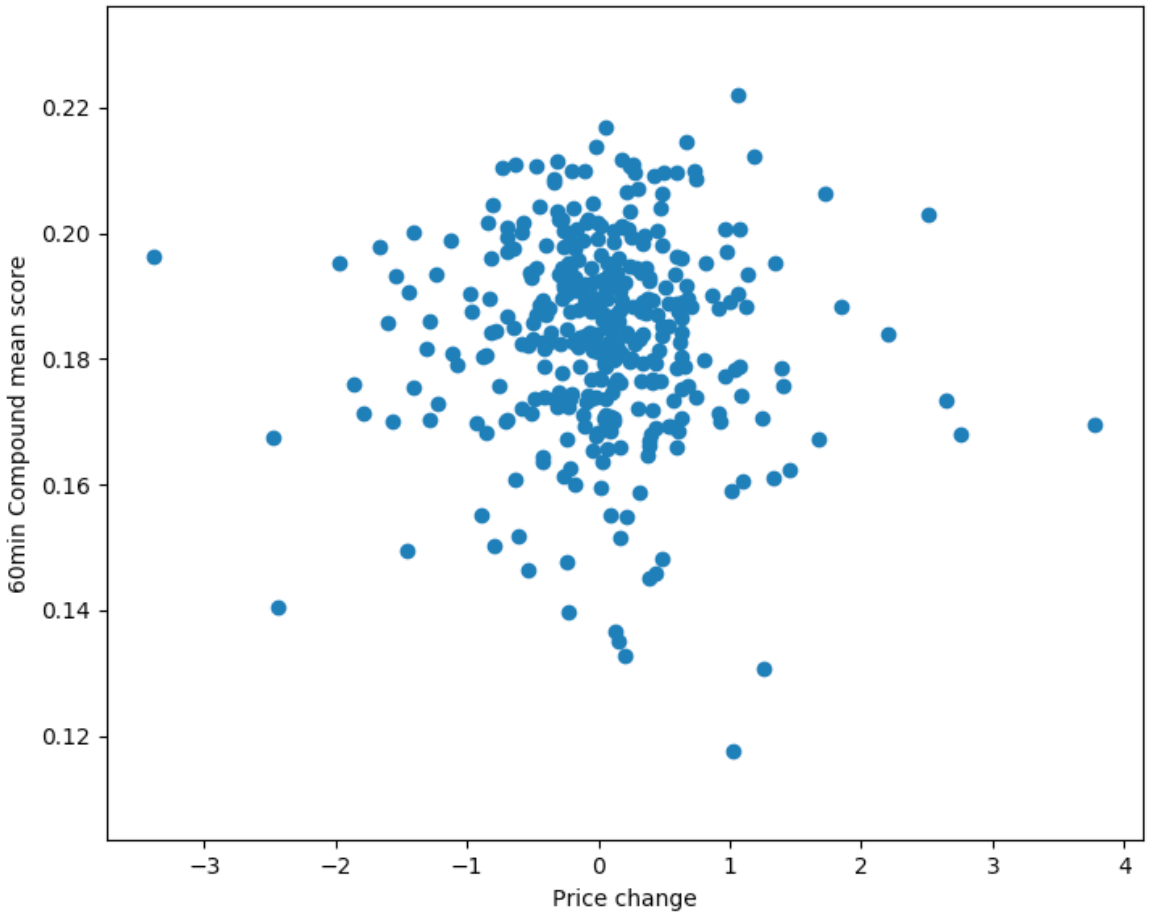

Figure 24: Compound mean[60min] and Price change scatterplot. Source: our own

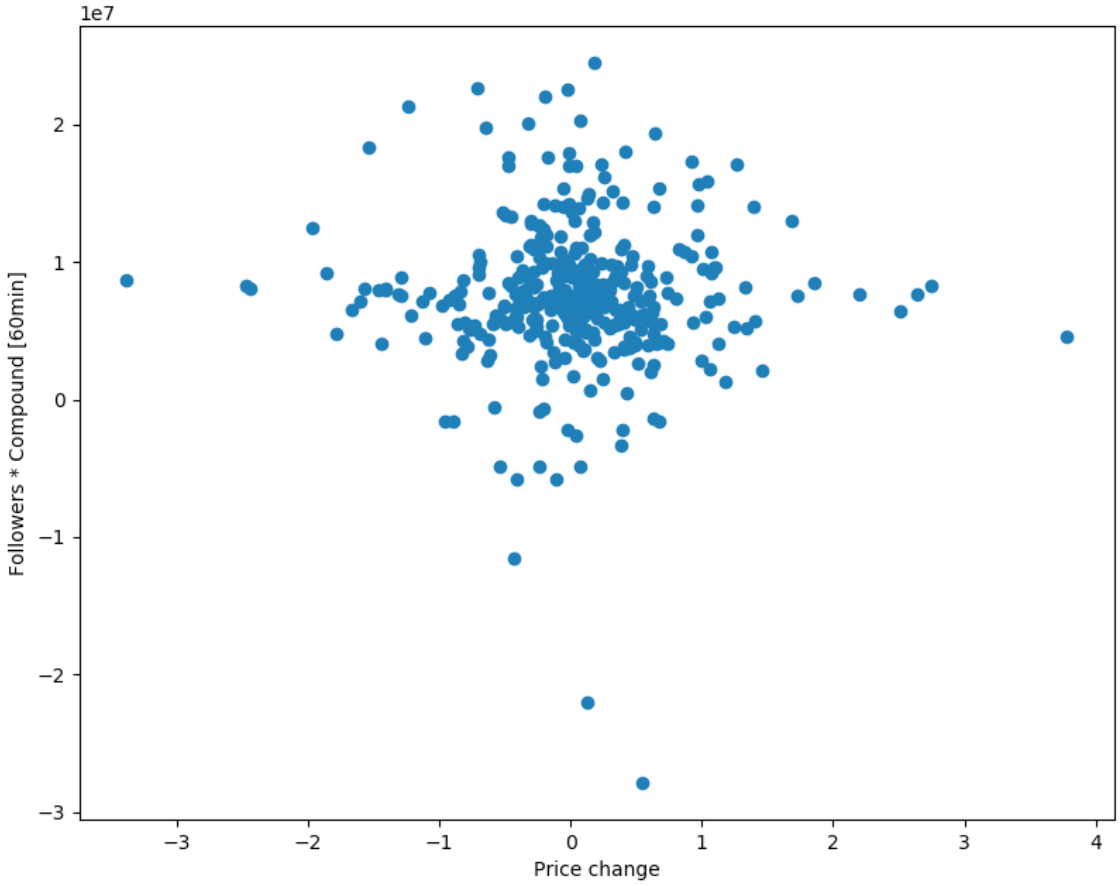

*Figure 25: Followers * Compound [60min] and Price change scatterplot. Source: our own*

As can be seen in both plots above, selected features do not reflect any correlation. This raises critical concerns about predictive power of selected features. However it also shows that most of the data is very neutral (located around 0), which could indicate problems with VADER sentiment scores in this context. At last, since no linear regression was detected, more complex machine learning algorithms might offer better results.

4.4 MACHINE LEARNING MODEL FOR PRICE CHANGE PREDICTION

This section elaborates on the steps that were taken - from building the model to making predictions.

4.4.1 SPLITTING THE DATA INTO INPUT FEATURE VECTOR AND OUTPUT TARGETS

Processed dataset has to be split into Input (X) and Output (Y) feature vectors. Input data is passed for the learning algorithm to learn from, and Output (or target values) are the ones that trained model will try to predict given new input X.

For the first prototype - target values will be *price_change* column, and rest of the features will be passed as input X.

4.4.2 SPLIT INTO *TRAIN* AND *TEST* SETS

Further step is to split X and Y matrices into training and testing datasets, so that generalization[62] of the model can be measured. Papers reviewed in the *State of the Art* (3.2) suggest training / testing data ratios from 70 to 95 percent for the training data, and rest for testing.

Prototype dataset has only 1432 entries, once tweets and prices were aggregated to 15 minute timeframes. Testing with larger timeframe will require much larger dataset of tweets.

First test was evaluated with 90-10 ratio - larger training dataset is preferred for fewer amount of data entries. Entries are not shuffled before splitting to keep the time sequence. Data is transformed using python's *pandas* or *numpy* libraries, that are mentioned in previous sections and elaborated in chapter 5.

4.4.3 STANDARDIZATION AND NORMALIZATION OF VALUES

By analyzing other projects source code it was observed that standardization and normalization of values is preferred or often required processing for data, that contains various different values and different scales e.g. followers count, price change.

In current dataset some of the features are already normalized[63] - *positive, negative, neutral mean* and *sum*. Normalization (also called MinMax Scaling) is a simple process which is done as follows - **y = (x - min) / (max - min)**

In regards standardization[64] - two columns in current dataset have gaussian distribution and deviation of 1 - *price change* (after it is normalized) and compound score. Others need to be

[62] Generalization shows how well model can predict on data it has never seen

[63] Normalization rescales attributes to the range 0 to 1

[64] Standardization rescales attributes to have mean of 0 and deviation of 1

standardized as well. Formula for standardization (also called Z-score normalization) **y = (x - mean) / standard deviation,** where:

- mean = sum(x) / count(x)
- standard deviation = sqrt(sum(sqrt(x - mean)) / count(x))

4.4.4 BUILDING A PREDICTION MODEL

Building a model for machine learning problem is a difficult task, as there is no right or wrong - best fit must be found over a lot of empirical testing for each specific use case. Many parameters must be tweaked until at least some sensible outcome is generated from the algorithm. This section will go through model building steps and parameter tuning decisions. Experiments and their implementation is inspired by some of the blogs on the internet[65][66].

Machine learning algorithm

Choosing machine learning algorithm is important step as every specific use case, requires different solution. This article [28] about time series stock price prediction, using twitter sentiment analysis provides a great summary of all algorithms that are suitable for such similar price + sentiment prediction problem.

First prototype is tested with two regression algorithms, as suggested in some of the relevant papers in State of the Art chapter.. Main reason to use regression instead of classification for this case is that the model is predicting continuous value - *price_change*. Here are the two algorithms used (results are presented in following section):

- Multiple Linear Regression (Ordinary Least Square). Very easy to implement, has no hidden layers and no activation function. Directly produces output from given inputs.
- Multilayer Perceptron. Has hidden layers and selected activation function. Can be easily tested with different various configurations.

Most development time was spent on MLP model, since Feature extraction step showed that VADER sentiment scores do not represent price change so well and linear regression is not the best fit. Further versions will be tested with more complex regression models, that were

[65] Link: https://www.kaggle.com/ironfrown/deep-learning-house-price-prediction-keras

[66] Link: https://towardsdatascience.com/create-a-model-to-predict-house-prices-using-python-d34fe8fad88f

suggested in other works, for example: ML algorithms like SVM or recurrent neural network like LSTM.

Structure of the network

This paragraph discusses building blocks of the neural network[67] that are used for prototype implementation. Arguments for some of the decisions in the architecture are also provided. Below (Figure 26) is high level structure of the first prototype:

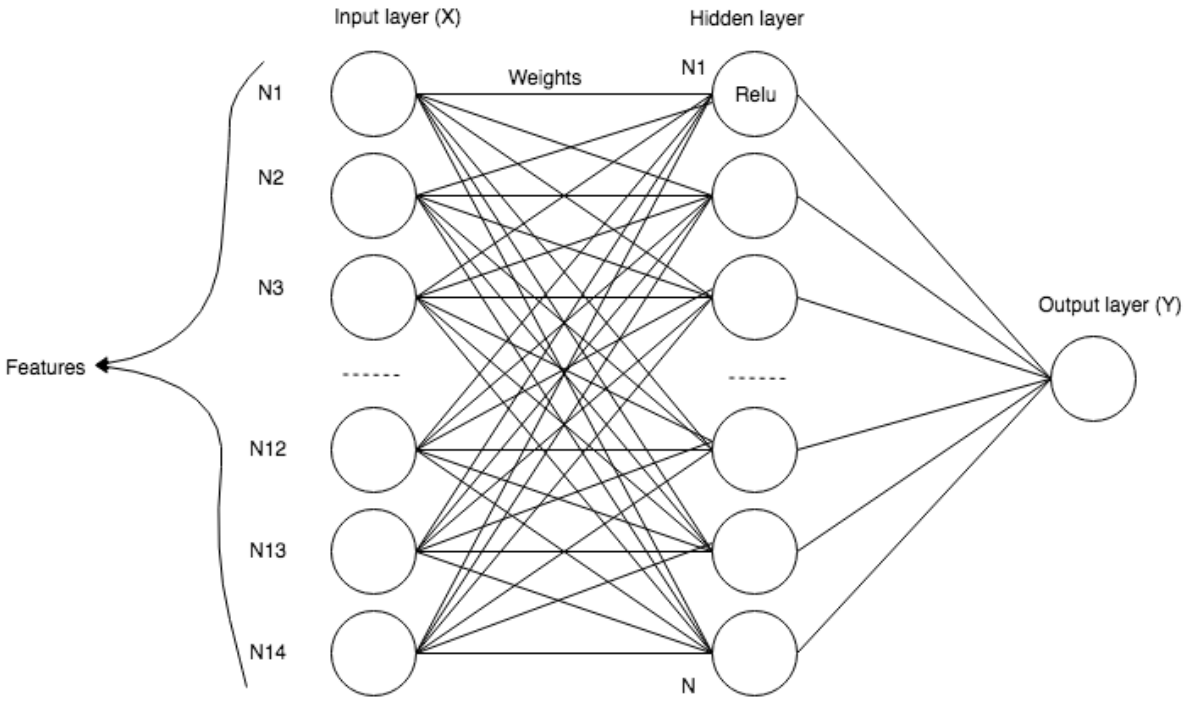

Figure 26: Deep neural network prototype. Source: our own

Hidden layers and neurons

As it can be seen in the diagram above - input layer has 14 neurons, that directly represents 14 features, that are passed in to the model. Whether these features have any predicting power will be seen in testing section.

Literature review and some *best practice* guides on the internet showed that there is no general rule how to select number of hidden layers and neurons in generic use case. It requires

[67] Linear Regression model has no activation function, and no training parameters.

empirical approach to determine best fit. However, based on best practices (discussed in 5.7) here is the configuration of prototype hidden layer:

- One hidden layer will be used in this experiment.
- Testing will be started with 7 neurons in the hidden layer.

It is important to note that it is questionable if deep neural network is even necessary for this use case or simple linear regression is better fit. Further testing will answer these concerns.

Activation function

Input layer does not need any activation function, especially when input data is already processed (standardized).

Selecting activation function for the hidden layer was straight forward, based on other researches and experiments (3.2). *ReLu and Leaky ReLu* are two most widely used functions. *Leaky ReLu* helps to prevent a problem that sometimes happens with original ReLu - neurons "die" in training process and never fire again, even if they are important for the prediction.

In the output layer no activation function is used, because output value is expected to be in range from -infinity to +infinity (positive or negative Bitcoin price change in percentage).

Loss and optimization functions

For regression problems it is common to use MSE (Mean Squared Error) function for evaluating the learning progress. It computes average squared deviation between predictions and targets, and allows to minimize the cost at each iteration.

Optimization function that will help reduce the loss and update weights in the network will be *Adam* (Adaptive Moment Estimation). It calculates different learning rate for every parameter, and is faster than other popular techniques i.e. SGD or *Adagrad*

Dropout

Another tool to optimize neural network performance and efficiency. Dropout should be used to deactivate only hidden layer neurons. Model will be tested with 20% dropout.

Target labels shift

It is assumed that some significant sentiment values in a timeframe [15min; 1h] reflects price change not directly on the following price candle, but rather an unknown amount of shift forward.

It takes time until some event or news spreads around enough, to make large amount of traders take actions in the market. Under this assumption model will be trained using a shift for the target labels (price_change) - this way model will learn to predict values with shifted timeframe (tweets from 13:00 -14:00 will predict price for 16:00).

However, it is very difficult task to find optimal shift for the target labels, one needs to make assumptions which spike in sentiment polarity could have influenced price change in the future.

Training configurations and results are presented in 6.2.

Regression model was built using *sklearn* library, and neural network using *Keras* framework. Implementation can be fou 5.7.

4.5 REQUIREMENTS

The requirement list (Figure 27 and Figure 28) shown below is only a small part of the whole system. MoSCoW were used to create the list in a descending order with Must requirements being in the top. The features that were seen to be critical for the prototype to work, were put as *Must* while the rest were prioritized lower and put in the other categories.

4.5.1 FUNCTIONAL

All the requirements were chosen based on the analysis, with the selection of twitter being the focus, sentiment score being extracted with VADER(3.5.4) and price collected from one popular price exchange.

#	Description	Section
MUST		
F1	Must be able to collect tweet data based within selected period of time	4.1.2
F2	Must be able to collect live tweets	4.1.2
F3	Collected sentiment data must include: text, amount of spread (followers, likes, shares), date	4.1.2
F4	Must be able to collect Bitcoin historical data within selected	4.1.2

	period (from 01/01/2014 to current date)	
F5	Must collect price data from one of the 3 most popular exchanges(described in 3.4)	4.1.2
F5.1	Historical price date must include - open, high, low, close prices and volume per timeframe	3.5.3
F5.2	Must be able to show predicted price change in dollar	4.3
F6	Must be able to show predicted price change in percentage	4.3
F7	Must be able to aggregate data into selected timeframes[68]	4.3
F8	Must be able to apply text processing steps (section 4.2.2)	4.2.2
F9	Must be able to manipulate selected datasets (columns, fields, sorting, remove corrupted values) (section 4.2)	4.2
F10	Must be able to weight sentiment score with followers as a feature (section 4.3.2)	4.3.2
F11	Must be able to extract followers count for a interval	4.3.2
F12	Must be able to store collected data into a database	4.3.2
F14	Must be able to extract Sentiment feature with positive, negative, neutral and compound	4.3.1
F16	Must be able to extract Price change as a feature	4.3.2
F17	Must be able to split data into training and test sets	4.4.2
F18	Must be able to split data into input and output sets	4.4.1
F19	Must be able to use linear regression	4.4.4
F22	Must be able to measure performance of model training	4.4.4

[68] E.g. 15min; 1hour; 1day

F23	Must be able to predict value with shifted timeframe (section 5.7)	4.4.4
F24	Must use Multilayer Perceptron	4.4.4
SHOULD		
F25	Must be able to collect news articles from a news site.	4.1.2
F26	Must use keyword dictionary as a feature	3.5.5
F27	Must be able to use TFIDF as a feature	3.5.5
F28	Must be able to extract Bitcoin Market Cap change as a feature	4.3.2
F29	Must be able to collect live Bitcoin price data	4.1.2
F30	Must be able to collect historical market cap data	4.3.2
F13	Must be able to extract BTC market Volume change feature	4.3.2
COULD		
F29	Must be able to use Support Vector Machine	4.4
F30	Must be able to use LSTM	4.4
F31	Must be able to estimate the amount of followers at the time of posting of the tweet	4.1
F32	Must be able to collect Reddit data	4.1
F33	Must be able to collect live Reddit data	4.1
F34	Must be able to collect Facebook data	4.1
F35	Must be able to train the model on top with new data	
F37	Must be able to extract number of retweets as a feature	4.3.2

WONT		
F36	Must be able to collect news articles from government websites	4.1

Figure 27: Functional requirements of the prototype

4.5.2 NON-FUNCTIONAL

Only Must non-functional requirements were selected and shown in the Figure 28 below. Some of them were based on the functional requirements and limitations of the prototype while others were based on hypothesis on what is seen as acceptable for such a system.

#	Description	Functional req.
NF1	Must be able to collect tweet data faster than 1/10th of specified time period.[69]	F1
NF2	Crawler must not exceed more than 250mb of RAM memory[70]	F1
NF3	Live tweet collection should have 99.5% uptime	F2
NF4	Crawler must be able to recover from crash within 5 minutes with probability of 99%[71]	F2
NF5	Must not exceed exchange API rate limit	F4
NF6	Must collect price in 1 minute candles	F4
NF7	Price data must be taken from same exchange	F4, F5, F6
NF8	Must be able to aggregate data in periods from 1 minute to 1 day	F7
NF9	Must be able to aggregate within 1ms per tweet (1.4hrs/5mil)	F7

[69] The acceptable time to crawl 30 days of tweets should not exceed 3 days due to project's limitations, otherwise the analysis stage would be delayed.
[70] Average RAM usage by single instance of *Twint* crawler based on testing.
[71] It was calculated that in 5 minutes, 2000 tweets would be lost on average, which is acceptable amount.

NF10	Must be able to apply text-processing steps for datasets up to 20gb without loading whole dataset into RAM memory.	F8
NF11	Must be able to apply text processing steps within 1ms per tweet (1.4hrs/5mil)	F8
NF12	Must be able to extract sentiment data from tweet within 1ms (1.4hrs/5mil)	F8
NF13	Must be able to apply dataset transformations for up to 20gb sets without loading whole dataset into RAM memory.	F9

Figure 28: Non-functional requirements of the prototype

5 DESIGN AND IMPLEMENTATION

The goal of this project is to develop a functional prototype to test the hypothesis if Twitter can be used to predict bitcoin price changes (1.1). Two stages for the project has been identified in analysis. First, it is necessary to collect and process historical twitter messages and bitcoin prices from exchange. Secondly, to preprocess the data and to train a model using machine learning techniques. In this chapter the choices for software and prototype design and implementation are presented.

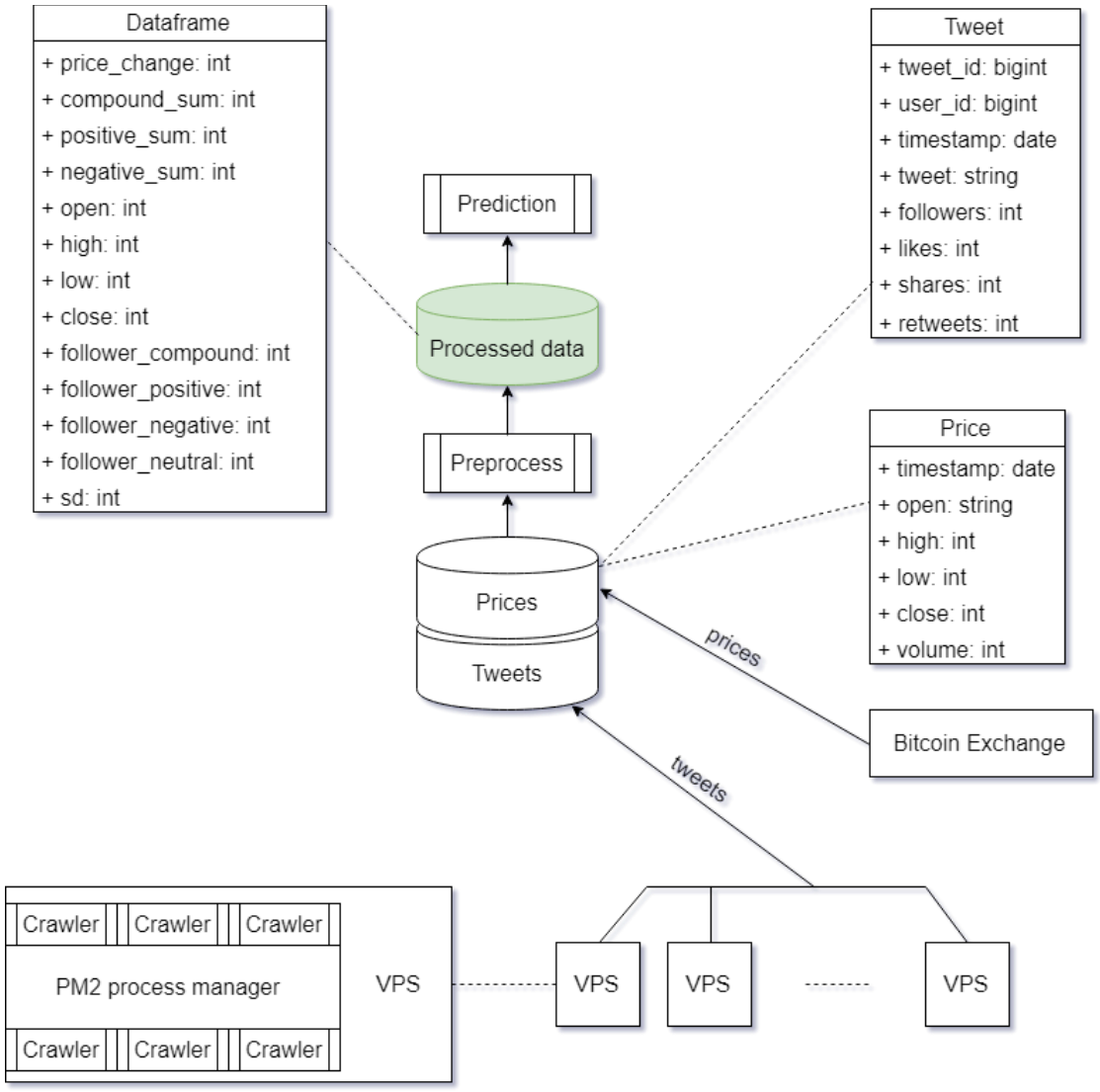

Figure 29: Flow diagram. Source: our own

The architecture as seen in Figure 29, displays the prototype diagram which consists of 9 Virtual Private Servers (VPS), running 6 crawlers each and one VPS running one real time crawler.

The crawlers are managed by PM2 process manager. Tweets are stored in local SQLite databases until all necessary historical data is crawled. When finished, SQLite databases are downloaded to and combined in the main database containing Tweets and Prices. Historical price data is downloaded from Exchange API and stored in the main database. Tweets and prices are then aggregated into specified time frames using *transform script (*Figure 56*)*. Processed data can then be used for training the model or making predictions.

5.1 COLLECTING HISTORICAL POSTS FROM TWITTER

One of the key requirements for data collection stage are collecting historical messages from twitter (F1, F3). Based on research in state of the art (3) and analysis (4.1) open-source crawler *Twint (*section 3.5.2*)* is used. Single instance of *Twint* is limited to one-month period and crawls backwards from specified most recent date. Single *Twint* instance running on VPS is able to gather one month of tweets in 3 days (NF1). In total there is 53 months (from 2014-01 to 2018-01) and while testing keywords with twitter search API, on average there are around 2 tweets posted per second which contain a keyword “bitcoin”. Therefore, based on our calculation 53 month period could contain up to 200 million tweets containing “bitcoin” keyword.

The command only crawl tweets, which includes the string “bitcoin” from January the 1st to February the 1st and then stores the tweets it in the database named: *year-month.db. The code is shown in* Figure 30:

```
python twint.py -s bitcoin --count --database 2014-01.db --since 2014-
01-01 --until 2014-02-01
```

Figure 30: Start Twint crawling. Source: our own

A single instance of *Twint* crawler can use up to 150 MB of RAM (NF2), which means that six instances within a single VPS can use up to 900 MB out of total 1000 MB of RAM[72]. The crawler can crash due to unhandled exceptions in the code or from network and VPS issues, therefore it is necessary to use a process manager.

PM2 is primarily a node.js process manager however, it is fully capable of managing python processes as shown in Figure 31. It allows to monitor the logs, memory usage and automatically restart python scripts if crashed.

[72] 1000MB is the limit of cheapest single VPS instance offered by Digital Ocean.

App name	id	mode	pid	status	restart	uptime	cpu	mem	user	watching
crawler_realtime	0	fork	13361	online	587	4D	0%	29.4 MB	root	disabled

Figure 31: PM2 Interface. Source: our own.

PM2 interface contains **application name**, **process id** in Linux system, **mode** in which process is running (fork/cluster), **status**, how many times the process was **restarted**, **uptime** since last restart, current **memory** usage and location for **error logs**.

```
pm2 start twint.py --name crawler201401 --interpreter /usr/bin/python3 -
- "-s bitcoin --count --database 2014-01.db --since 2014-01-01 --until
2014-02-01"
```

Figure 32: Command required to start crawler application within PM2 process manager with monitoring function

Within each VPS six *Twint* crawlers are spawned in *PM2* environment as shown in Figure 33. Each crawler creates a local SQLite database to which it connects and stores the tweets. In case of crash *PM2* restarts the process.

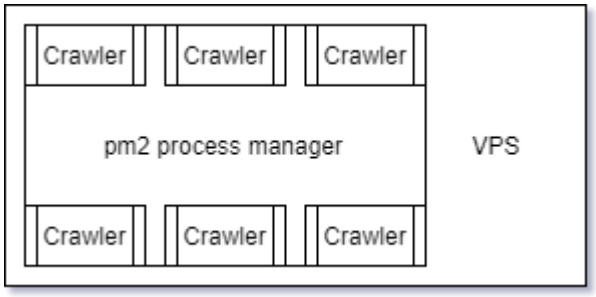

Figure 33: Virtual Private Server. Source: own

This architecture allows to setup crawlers on demand and scale horizontally. Digital Ocean allows a snapshot function where of one VPS can be used to create new machines as a blueprint. This functionality allows the system to easily request a new VPS if additional crawling or processing power is necessary. To fulfill requirements F32, F33, F34, F36 the crawler would need to be improved and generalized, but the scaling problem would be solved by the same VPS blueprint structure.

During initial testing *Twint* has crashed or restarted due to network issues. *Twint* is stateless therefore when *PM2* restarts the process it passes same arguments and the application starts crawling from the beginning. Due to how Twitter crawling working as infinite list, it is not easy to start crawling again from the same spot. To recover from crash the latest crawled tweet is found

in the Database and start from that day. In worst case it is necessary to crawl through 1 day of tweets to recover.

Code snippet allowing to recover from crashed crawler is shown in figure 32

```
earliest_tweet = """
            SELECT `_rowid_`,date FROM `tweets` ORDER BY `_rowid_` ASC
LIMIT 1;
            """
        cursor.execute(earliest_tweet)
        resume_date = cursor.fetchone()
        if resume_date != None:
            global resume
            resume = datetime.datetime.strptime(resume_date[1], "%Y-%m-
%d").date()
            resume = resume + datetime.timedelta(days=1)
```

Figure 34: Restore crawling from latest crawled date. Source: own

Using this setup, the crawler collected up to 70% of total amount of tweets containing a string "bitcoin" from 2014-01 to 2018-05. Some of the tweets were not collected due to Twitter repeatedly not returning results for some months or other undefined reasons.

5.2 COLLECTING REAL TIME TWITTER DATA

The requirement to crawl real-time data (F2) is fulfilled by using official Twitter API and a library *Tweepy.* It is managed by the PM2 process manager to ensure that application is restarted if it crashes due to network or other issues.

Command to launch Tweepy real time crawler is shown in Figure 35.

```
pm2 start tweepy.py --name crawler_realtime --interpreter
/usr/bin/python3
```

Figure 35: starting Tweepy. Source: our own

The example below (Figure 36) shows how to connect to the stream of real time tweets and receives callback with tweets which contain the specified keywords. Other statistics like number of followers, tweet id, user id is also returned by the crawler.

Code snippet below (Figure 36) shows the returns tweets containing specified keywords from Twitter real-time stream.

```
stream.filter(track=['btc','bitcoin','xbt', 'cryptocurrency'],
languages=['en'])
```

Figure 36 Specifying filter for Tweepy. Source: our own.

Tweepy by default does not store information in database, therefore it was necessary for a short solution which is shown in (Figure 37) below. The structure of Tweets is the same as described in database section (5.4) for both historical and real time crawler. To avoid storing Unicode characters and emoji in the database, emoji[73] library which changes emoticons into text is used during crawling process.

```
hashtags =  ",".join(re.findall(r'(?i)\#\w+', status.text,
flags=re.UNICODE)) #extract hashtags
text = emoji.demojize(status.text) #remove emojis
entry = (status.id, status.user.id, date, time, status.user.time_zone,
status.user.screen_name, text, 0, 0, 0, hashtags,
status.user.followers_count, status.user.friends_count,
status.user.listed_count, status.user.favourites_count,
status.user.statuses_count)
cursor.execute('INSERT INTO tweets
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)', entry) #insert tweets into
Database
```

Figure 37: Code snippet to store Tweets into database. Source: our own

Tweepy enabled efficient real time tweet collection, in addition it can search for multiple keywords at once. Lastly, the tool put less stress on Twitter platform compared to *Twint* because search operations are more intensive in processing power compared to stream filter method.

5.3 COLLECTING BITCOIN HISTORICAL PRICE DATA

Historical Bitcoin price data is crucial for this project (F4), as it provides target labels for predictions and some extra technical analysis features to make those predictions more accurate (volume, standard deviation etc.). As mentioned in the 4.1.2, historical price data is collected using CCXT library. It provides smooth and unified API interface with various endpoints for the developer.

For this experiment only one function is used - *fetch_OHLCV* (F5). Signature from library documentation is displayed in the Figure 38.

[73] Link: https://pypi.org/project/emoji [accessed 22/05/2018]

```
fetchOHLCV (symbol, timeframe = '1m', since = undefined, limit = undefined, params = {})
```

Figure 38: Fetch Open High Low Close Volume from exchange. Source: our own

This is one of the most important configuration variables, that were used for the first data collection attempt:

```
exchange = ccxt.binance()
```

Figure 39: exchange instance is created. Source: our own

In Figure 39 exchange instance is created (NF7), which will allow calling uniform CCXT endpoints and get data from the exchange API. Binance exchange is used for the prototype, since it offers rate limit of 1200 requests per minute, while the most popular exchange at the time of the paper being written - Coinbase, only offers 60 requests per minute. Binance enables much faster historical data collection.

```
symbol = 'BTC/USDT'
```

Figure 40: Selecting currency. Source: our own

Currency pair is selected. Since Binance exchange do not work with FIAT currencies[74], it uses cryptocurrency USD Tether[75] as an equivalent for USD. Every USDT unit is backed by US Dollar, and its rate is always equal to one dollar.

```
from_timestamp = date_to_unix(string_to_date(sys.argv[1] if len(sys.argv) > 1 else '01/01/2014'))
to_timestamp = date_to_unix(string_to_date(sys.argv[2])) if len(sys.argv) > 2 else current_time
```

Figure 41: Defining period for price collection. Source: our own

Two variables that define in which period price data should be collected can be passed manually as a script argument, or else default values will be used. Not many exchanges have stable BTC data before 01/01/2013.

```
timeframe = '1m'
```

Figure 42: Choosing timeframe. Source: our own

This variable defines in what timeframe OHLCV data should arrive (NF6). While exchange APIs offer larger timeframes, which could be directly used for machine learning experiments, it is

[74] https://en.wikipedia.org/wiki/Fiat_money

[75] https://www.cryptocompare.com/coins/guides/what-is-usdt-and-how-to-use-it/

convenient to collect data in the smallest unit possible, and aggregate data into larger ones, when desired.

```
batch_limit = 1000
```

Figure 43: Choosing batch limit. Source: our own.

Most of exchanges has limit on how many historical prices can be retrieved in one request (in most cases it is 1000). This variable is not only passed to the request but also used to calculate how many batches / requests will be needed to retrieve data for selected period.

API rate limit and batches

If requested amount of price candles are larger than the maximum batch size (1000 entries = 1000 minutes), several batches and requests are necessary to be made. Script snippet below (Figure 44) shows how batch count is calculated:

```
def batch_count():
    diff = secs_to_min(to_timestamp - from_timestamp)
    batch_count = math.ceil(diff / batch_limit)
    return batch_count
```

Figure 44: Batch count. Source: our own

batch_count tells how many requests should be made to fetch all the data in selected period. This leads to another implementation problem - executing number of requests might reach exchange's API limit, in which case IP address is banned with exponentially growing time penalty. Even though Binance documentation says limit is 1200 requests per minute, during test, API would return limit error if more than one request was made per 1 second (NF5). Script below (Figure 45) handles rate limitation and makes sure that batch window is moved 1000 minutes forward for each following request.

```
def get_historical_prices():
  count = batch_count()
  prices = []
  seconds_left = from_timestamp
  print('batch count: ', count)

  for x in range(count):
    print('#', x, ' | ', x / count * 100, '%')
    print('Batch from ' + timestamp_to_date(seconds_left) + '\n')
    batch = exchange.fetch_ohlcv(symbol, timeframe, seconds_left * 1000, batch_limit)
    prices = prices + batch
    seconds_left += 1000 * 60
    time.sleep(1)

  return numpy.array(prices)
```

Figure 45: Historical prices. Source: our own

The timeout period slows down the execution of the script significantly; however, in ideal scenario historical prices should only be collected once.

I.e. time for getting one year's price data can be calculated like this:

- 525,600 minutes / 1000 *batch_limit* = 526 *batches_count*
- 526 seconds = **approx. 9 minutes.**

Here is an example of one data entry in the *fetch_OHLCV* response:

```
[
    [
        1504541580000, // UTC timestamp in milliseconds, integer
        4235.4,        // (O)pen price, float
        4240.6,        // (H)ighest price, float
        4230.0,        // (L)owest price, float
        4230.7,        // (C)losing price, float
        37.72941911    // (V)olume (in terms of the base currency), float
    ],
    ...
]
```

Figure 46: Price Values. Source: our own

Price values can be used to calculate different technical indicators and timestamp is used to aggregate data into diff. timeframes.

Prototype was only trained and tested with historical prices, which are stored locally in a .csv file and manually loaded to the python scripts. However, production version would require price prediction on live data (F29). This use case requires some infrastructure:

- Python script with CCXT library must be hosted on a cloud, in order to keep it running 24/7.
- Prices should be stored in either local or remote database

For real-time data collection, *fetch_OHLCV* function could be called every minute, with *batch_limit* set to 1 and this way receive latest minute candle.

5.4 Database

To fulfill requirement (F12) database is necessary to store collected tweets. Initially it was planned to use central SQL database to which each crawler would connect and store data, however during the testing, it was not able to support storing data generated by 53 crawlers. This problem would most likely be solved by dedicating more resources to the server. Due to limited resources and as a temporary workaround it was chosen to use local SQLite databases for each crawler. This required additional step of downloading all the databases from servers, merging them with a script shown in Figure 47 and importing them into single MySQL database.

```
while True:
    rows_cur1 = cur1.fetchmany(10000)
        if not rows_cur1:
           print("migration completed")
           break
        for row in rows_cur1:
           try:
              cur2.execute("INSERT INTO tweets({0}) VALUES
({1})".format(columnNamesStr, ques), row)
           except sqlite3.IntegrityError: # tweet is already in the db
```

Figure 47: Snippet from merge.py script for merging all SQLite databases into one

SQLite databases are great at handling data from single data source and a few connections, the structure is lightweight and simple. Crawler itself is stateless and when running, it checks if a database exists and if so, it starts storing data otherwise it creates a table with necessary fields and stores the data. Crawling is sequential from newest tweet within specified period to oldest, therefore the database allows crawler to identify the last tweet which was crawled.

Tweets table structure within local SQLite databases is simple as displayed in Figure 48. Tweets are indexed by id and contains id and the username of the poster. The tweet also contains other information when the tweet was posted, which time zone and other statistics.

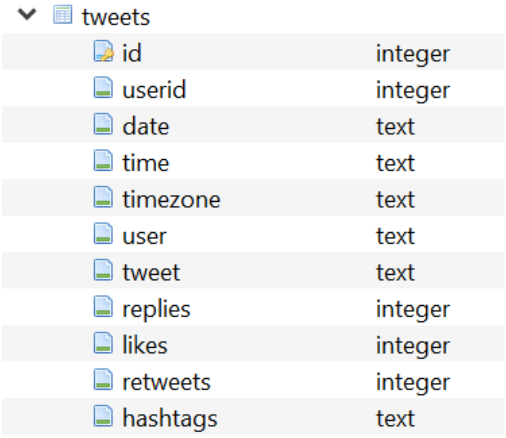

Figure 48: SQLite database structure. Source: our own

Central database is necessary to ease the processing and other operations. After collection the tweets, they are then imported from local databases with the prices, which are gathered from exchange API into a single database. In current prototype the import is scheduled manually, however in the next iteration, the crawlers could write data directly into central database. Raw tweet and price data is separated from processed data because some operations performed during data processing stage are not reversible.

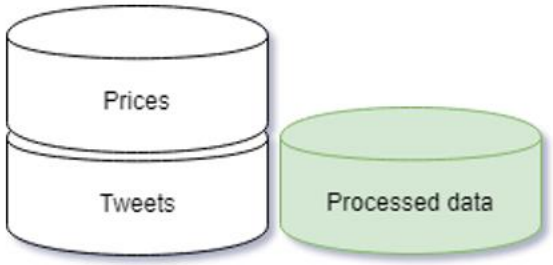

Figure 49: Prices, Tweets and Processed data databases. Source: our own

Central database contains Tweets and prices in a structure as shown in Figure 49. Processed data in the database contain Tweets and Prices (shown as Dataframe in Figure 50) which are aggregated on specified period and contain sentiment and price data. The details of data processing are discussed in next section.

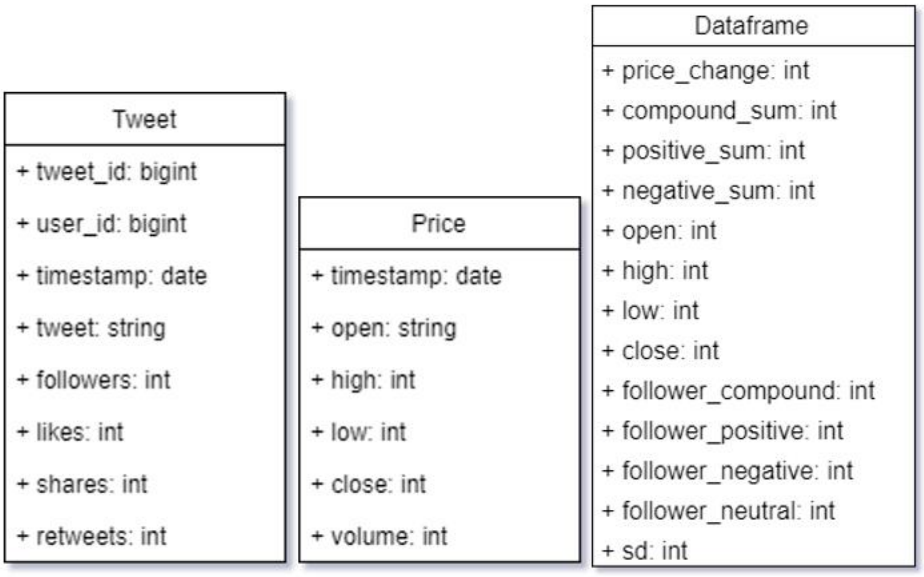

Figure 50: Database structure. Source: our own

To sum up, the structure of database should be improved in the next iteration. Crawler part of prototype was implemented in early stages of the project due to time limitations, as it was necessary to collect enough data for further analysis. This decision made data handling process more manual and time consuming. However, without data it would have been difficult to perform any analysis and train the model.

5.5 SERVER

Virtual Private Server (VPS) supplied by Digital Ocean is the preferred choice due to familiar development experience of project members, flexibility and on demand scaling. VPS sold as a service contains an Operating System and provides a super user access to the machines. In total nine VPS (1 GB 1 vCPU 25 GB 1 TB)[76] were setup for the crawling process, costing $0.0007/hour each. Running nine crawlers for five consecutive days would cost around $1 for servers and $5 for bandwidth. Each VPS is setup with Ubuntu Linux Operating system, PM2 process manager and six crawler instances as shown in Figure 51.

[76] Link: https://www.digitalocean.com/pricing [accessed 22/05/2018]

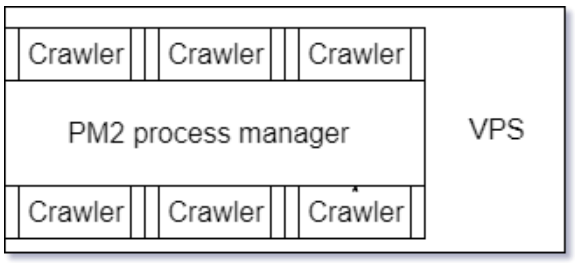

Figure 51: VPS structure. Source: our own

To sum up, VPS is a good choice for the crawling task, for next iteration of prototype, each server can be requested and setup through Digital Ocean API. In addition, price per hour billing allows very efficient use of resources. The process of crawling can be scaled horizontally by requesting more machines instead of more RAM and CPU power. Horizontal scaling is considered preferred method due to CPU and RAM scaling being more expensive in general.

5.6 Data Preprocessing

Once raw data is collected it must be cleaned up, transformed and merged into one feature vector. Input and output of this process are well presented in Figure 50 in previous section. This section will elaborate on this process). Few code snippets will help present critical implementation details. For all data processing tasks, it is necessary to process the files in chunks to comply with (OBJ) and (NF13OBJ requirements. The file size can exceed the total amount of RAM in the system because the files are processed in chunks, code sample is shown iOBJOBJFigure 52.

```
while True:
    df = pd.DataFrame(cursor.fetchmany(1000))  # read only 1000 tweets
at once
    if len(df) == 0:  # we are done if there are no data
        break
    else:
        df.to_csv(f, header=False)  # write to the file
```

Figure 52: Code snippet showing how to process the data in chunks. Source: our own

Data received from *Tweepy* is imported into *Azure Workbench* platform, which helps remove unnecessary columns or corrupted entries using visual interface and no coding. Filtered dataframe is saved and passed into *prep_tweets.py* script as .csv file. The script applies cleaning steps that are defined in chapter 4.2.2 (F8) and then extracts sentiment scores with VADER. Below is the python snippet that processes tweet string (NF11).

```
def clean_tweet(text):
    # 1. Lower case
    processed_text = text.lower()
    # 2. Remove mentions
    processed_text = re.sub(r"@[\w\S]+", '', processed_text)
    # 3. Remove hashtag symbols
    processed_text = re.sub('#', '', processed_text)
    # 4. Remove 'rt' from retweeted tweets
    processed_text = re.sub(r'rt ', '', processed_text)
    # 5. Remove numbers and special characters
    processed_text = re.sub(r'\s[\W\d:]+|[\W\d:]+\s', ' ',
processed_text)
    # 7. Remove links
    processed_text = re.sub('https?://[A-Za-z0-9./]+','',
processed_text)
    # 8. Remove extra white spaces that were introduced in previous
steps
    processed_text = re.sub(r' +', ' ', processed_text)
    return processed_text.strip()
```

Figure 53: removing unwanted symbols and strings from tweets. Source: our own

In the same loop, processed tweet is passed to VADER algorithm (F14). Returned scores for each tweet are appended as new columns to the dataset (F9) as shown in Figure 54 below.

```
data['compound'] = pd.Series(compound)
data['positive'] = pd.Series(positive)
data['neutral'] = pd.Series(neutral)
data['negative'] = pd.Series(negative)
```

Figure 54: Assigning new sentiment scores. Source: our own

After preprocessing is done dataframe is outputted to *data_clean.csv* file. Cleaned data file is loaded into *prepare_cols.py* script, which will extract some new features (Figure 55):

```
data['followers_compound'] = data['followers'] * data['compound']
data['followers_positive'] = data['followers'] * data['positive']
data['followers_neutral'] = data['followers'] * data['neutral']
data['followers_negative'] = data['followers'] * data['negative']
```

Figure 55: Multiplying sentiment scores by amount of followers. Source: our own

Script also prepares some new columns with sentiment scores, so that in the following aggregation step (Figure 56), these columns can be aggregated into features for a selected timeframe, e.g. *compound_sum* or *compound_mean*. Also *tweet* column with all the tweet texts is removed, since the actual tweet is not used anymore, once sentiment score is extracted. It was discovered unintentionally that string removal from dataset improves processing speed dramatically.

Finally, last processing step for the twitter data is aggregating cleaned data into selected timeframes. This can be achieved by performing *resample* method on a Pandas dataframe object:

```
df.resample(timeframe).agg({'datetime': 'first', 'followers': 'sum',
'compound_sum': 'sum', 'compound_mean': 'mean', 'followers_compound':
'sum', 'positive_sum': 'sum', 'positive_mean': 'mean',
'followers_positive': 'sum', 'neutral_sum': 'sum', 'neutral_mean': 'mean',
'followers_neutral': 'sum', 'negative_sum': 'sum', 'negative_mean': 'mean',
'followers_negative': 'sum'})[order]
```

Figure 56: Aggregate the scores into specified timeframe. Source: our own

As shown in Figure 56, *resample* is executed on a dataframe with timeframe arguments passed in. *timeframe* argument is a string of time format, i.e. *15min, 60min, D* (for day) etc. (F11, NF8). Also *agg* function perform specific actions with the values of a selected column. For example, values of *compound_sum* column will be summed, and *compound_mean* values will be averaged. Some of these features reflect very similar properties of a dataset and after testing, some of these will probably be removed.

Aggregated dataset is saved to *data_aggregated_[*chosen timeframe*].csv* file, which will be concatenated with processed price dataset.

And now regarding price data aggregation - all columns from OHLCV price data are used for feature extraction and aggregation, there is no need for any column removal. However, timestamp needs to be converted to Datetime format (e.g. 2014/01/01 12:01:00). Once converted it is seen that dataset is not ordered by date, this can also be achieved with *Workbench* preprocessing feature (as shown in).

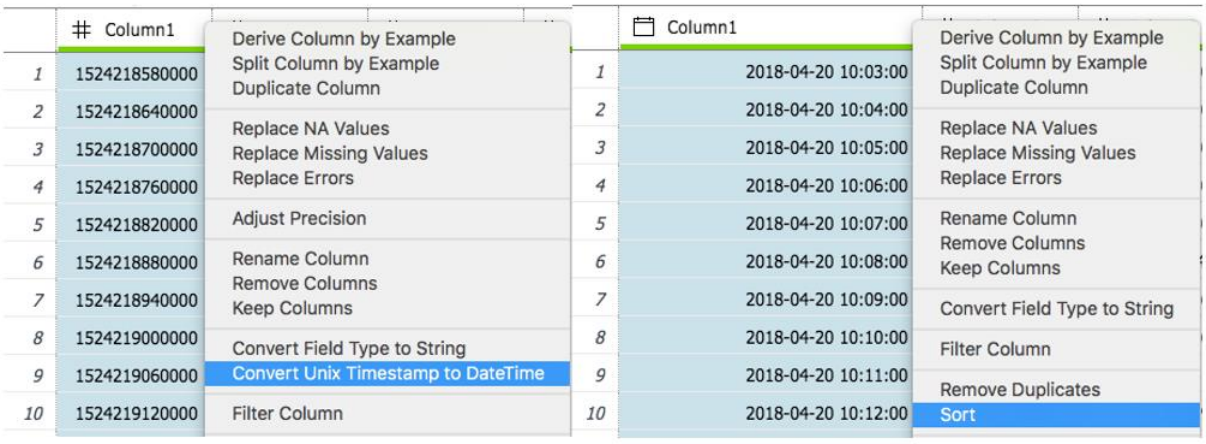

Figure 57: Converting from Unix timestamp into DateTime and sorting. Source: our own

Price data comes with no column names, which makes it difficult to work with dataframes. Column names - *datetime, open, high, low, close, volume,* are added using workbench interface and then sorted by date, as shown in Figure 57.

When price dataset has been prepared to work with, it must be aggregated into selected timeframe. This is done very similarly as it was done with tweet data aggregation:

```
df.resample(timeframe).agg({'open': 'first', 'high': 'max', 'low':
'min', 'close': 'last', 'volume': 'sum', 'standard_deviation': lambda x:
round(np.std(x), 2)})[order]
```

Figure 58: Aggregating price data. Source: our own

During the aggregation, extra feature is extracted from the dataset - *standard_deviation.* It is calculated by taking each minute's closing price and applying *numpy's std* function.

Last touch for the price dataframe before it is saved, is to extract target labels and append *price_change* (F16) column to the dataset. Price change percent is calculated using Open and Close price difference of a single entry

```
# Calculate delta percentage and add column to df
delta = np.array([])
for o, c in zip(df['open'], df['close']):
 delta = np.append(delta, get_delta_percent(o, c))
df['price_change'] = delta
```

Figure 59: calculating price change based on open and close prices. Source: our own

After this step, dataframe is saved to *[selected timeframe]_agg_btc_prices.csv* file.

Final step of preprocessing is concatenating two cleaned up feature vectors to a single dataframe. This is easily done through Azure Workbench interface:

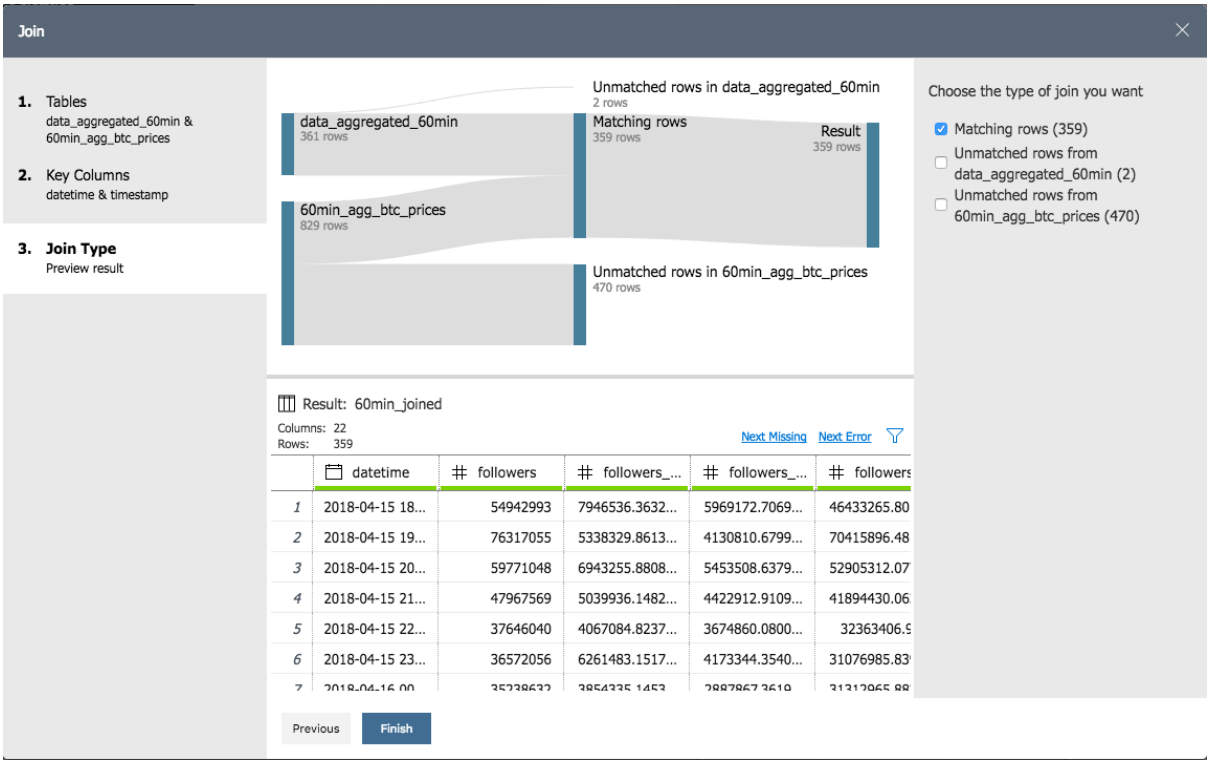

Figure 60: Joining tables in Azure Workbench. Source: our own[77]

After this step, dataframe is again saved to final .csv file - *[chosen timeframe]_final.csv*

5.7 PRICE CHANGE PREDICTION MODEL IMPLEMENTATION

Model implementation is based on some of the guides and tutorials found on the internet[78][79]. This section will show some of the code that is crucial for the experiment.

Splitting data into input (X) and output (Y) vectors (F18)

First, prepared dataframe is split into input (X) and target (Y) vectors using selected *shift* value (read more in 4.4.4) (F23). During this process X data is also scaled using *sklearn scale* function, that centers the data (mean = 0). Process shown below (Figure 61):

[77] Appendix E. full size
[78] Blog: https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90 [accessed 22/05/2018]
[79] Article: https://machinelearningmastery.com/make-predictions-long-short-term-memory-models-keras [accessed 22/05/2018]

```
shift = 4
# Split dataset into X and Y
X = df.loc[:,'followers':'standard_deviation']
X = preprocessing.scale(X)
X_forecast = X[-shift:]
X = X[:-shift]
Y = df['price_change'].shift(-shift)[:-shift]
```

Figure 61: Shifting rows for prediction. Source: our own.

In cases when K-Fold cross validation is not used, data will be split into X_train, X_test, y_train, y_test sets using *train_test_split* from *sklearn* (F17).

Linear regression model (F19)

The LR model is built and trained in two simple python lines (Figure 62):

```
reg = LinearRegression()
reg.fit(X_train, y_train)
```

Figure 62: Building Linear regression model with sklearn library. Source: our own

LinearRegression algorithm is built with *sklearn* library (F19). Code sample shown in Figure 63.

Multilayer Perceptron (F24)

Structure of the first prototype of neural network is defined in section 4.4.4. It is built using Keras framework.

```
# define base model
def baseline_model():
  model = Sequential()
  model.add(Dense(15, input_dim=y_shape))
  model.add(Dropout(0.1))
  model.add(Dense(7, activation='relu'))
  model.add(Dense(1))
  # Compile model
  print(model.summary())
  model.compile(loss='mean_squared_error', optimizer='adam')
  return model
```

Figure 63: Building MultiLayer Perceptron in Keras. Source: our own

Function in Figure 63 creates a baseline model, it uses main building blocks of Keras library:

- *Sequential* model is a linear stack of layers
- First Dense layer is an input layer and must include *input_dim* property. It is defined by number of target labels in Y.
- Last Dense layer is output layer, and contains only one neuron (predicted value)

- *model.compile* function prepares model for training. Loss and Optimizer functions are required.

In the final step, predictions are made by fitting test data - *X_test* to the trained model. K-Fold cross validation of 10 folds was applied on the data to see how well the model can generalize on new data. Performance is measured using MSE (results found in 7.2).

Process implemented with python and shown in Figure 64:

```
pipeline = Pipeline(estimators)
pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)
kfold = KFold(n_splits=10, random_state=seed)
results = cross_val_score(pipeline, X, Y, cv=kfold)
predictions = cross_val_predict(pipeline, X, Y, cv=kfold)
print("K-Fold Standardized: %.2f (%.2f) MSE" % (results.mean(),
results.std()))
print("Fitted model MSE: %.2f" % mean_squared_error(y_test, y_pred))
```

Figure 64: K-fold with Keras library. Source: our own

6 TESTING

The main goal of testing is to make sure that the prototype is functional and fulfills the requirements defined in Analysis section. The prototype will be considered successful if it passes 99% of defined test cases and requirements (see in 4.5). Secondary goal is to test if chosen machine learning models can predict Bitcoin price changes with sufficient accuracy. Due to our limited knowledge in machine learning and experimental nature of this field it is necessary to test different machine learning techniques, models, and then to compare the results. It is common to start with simpler models like Linear Regression to have baseline score to which other models can then be compared to.

6.1 TEST CASES

It is necessary to test if implemented prototype meets the requirements. However, only the most important requirements are listed (4.5). Overall, the test cases were focused on main requirements for data collection, data processing and training model. The results are shown in the test case (Figure 65) below.

Req.	Test Steps	Actual Result	Pass/Fail
F1	Collect tweets posted during 2018-05 containing keyword bitcoin	Collected 4579098 tweets	Pass
F2	Collect real time tweets containing keyword bitcoin	Real time tweets are being received	Pass
F3	Tweets must include: text, followers, likes, shares, date	Follower count is missing in historical tweets	95%
F4	Collect bitcoin price data within 2018-05	Collected bitcoin price data for 2018-05 period	Pass
F7	Aggregate data into 15 min periods	Aggregated data into 15min periods	Pass
F8	Apply text processing steps	Applied text processing steps	Pass
F9	Manipulate dataset (columns, fields, sorting)	Successfully applied dataset manipulations, columns, fields, sorting	Pass

F19	use Linear Regression	Used Linear regression	Pass
F24	use Multilayer Perceptron model	Used Multilayer perceptron	Pass
NF1	Collect tweet data faster than 1/10th of crawling period	Collected 30 days of tweets within 3 days	Pass
NF2	Watch memory usage of Crawler within VPS	Average RAM memory used is 200mb by single instance	Pass
NF9	Aggregate 4579098 tweets to 60 minute intervals	1.92 seconds	Pass
NF10	Apply text processing without loading whole dataset into RAM memory	Processed data by 256mb chunks each processing step	Pass
NF11	Apply text processing steps for 4579098 tweets	~13 minutes	Pass
NF12	Extract text sentiment data from 4579098 tweets	~20 minutes	Pass
NF13	Apply data transformations without loading whole dataset into RAM memory	Processed data by 256mb chunks each processing step	Pass

Figure 65: Test Cases. Source: our own

6.2 Testing Machine Learning Models

This section will investigate results of different machine learning approaches used to train models and predict Bitcoin's future price changes. First baseline model was built based on knowledge from Analysis chapter (4). From then onwards some different configurations were tested to see if they improve MSE score and plotted predictions actually correlates with real price changes.

Best results of different configurations are presented below (results are provided for both tested timeframes – 15 and 60 minutes). Graphs illustrate price change predictions versus real price changes.

Model	**Linear Regression**
Shift	**4**
MSE	**0.16 [15min]** **0.49 [60min]**

Figure 66: Linear regression results. Source: our own

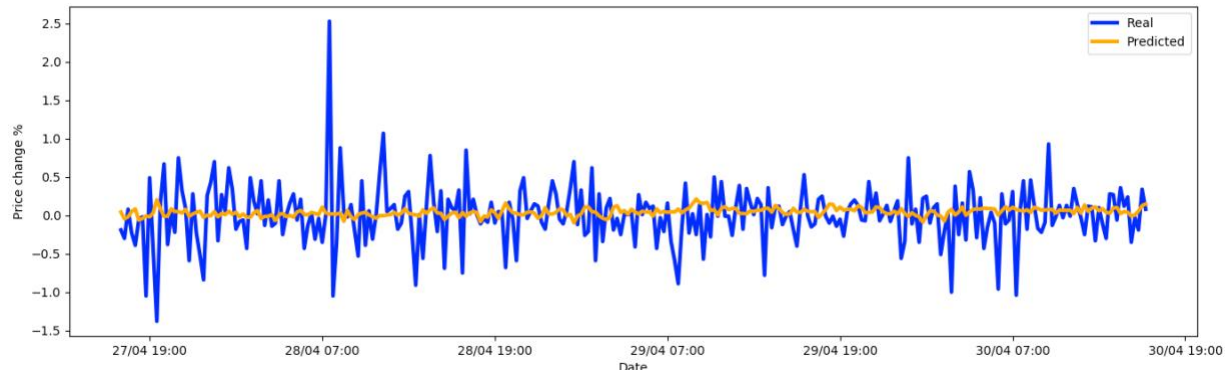

Figure 67: LR - 15 minutes timeframe. Source: our own[80]

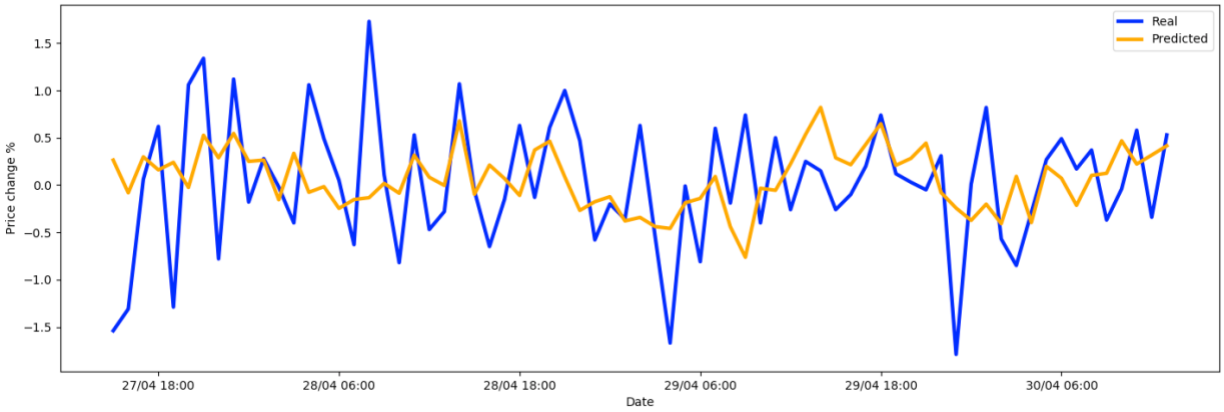

Figure 68: LR - 60 minutes timeframe[81]

While 15 minutes timeframes gives better MSE score (Figure 66), looking at Figure 67 it can be seen that predicted price change variance is very little comparing to real values. As discussed in section 4.3, there is only very weak correlation between sentiment features and price changes and this can be seen in 60 minutes prediction graph above (Figure 68).

Model	**Multilayer Perceptron**
Shift	4

[80] Appendix F. full size
[81] Appendix G. full size

Epochs	1000
Dropout	10%
Batch size	128
Hidden neurons	7
Activation func.	ReLu
Loss func.	MSE
Optimizer	Adam
MSE	**0.16 [15 min]** **0.54 [60 min]**

Figure 69: Multilayer Perceptron results. Source: our own

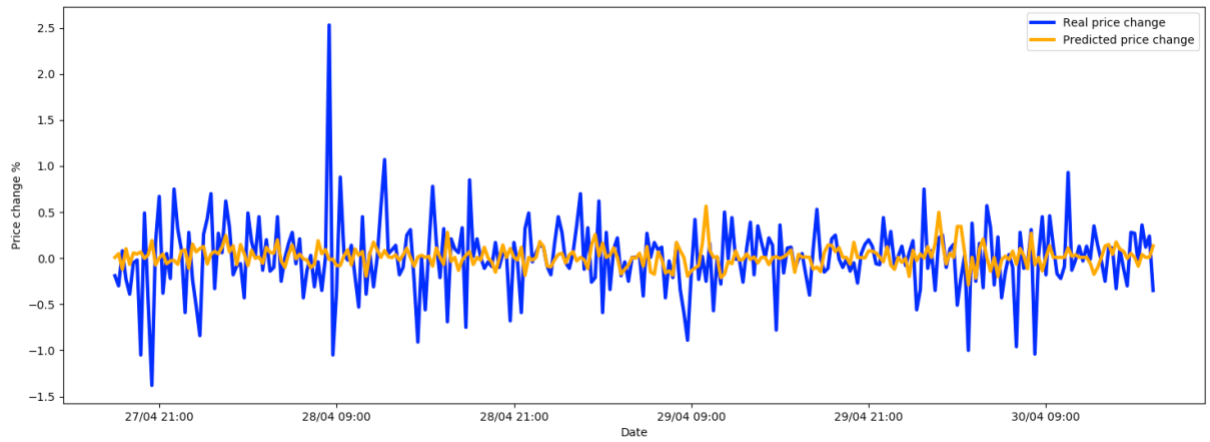

Figure 70: MPL - 15 minutes timeframe. Source: our own[82]

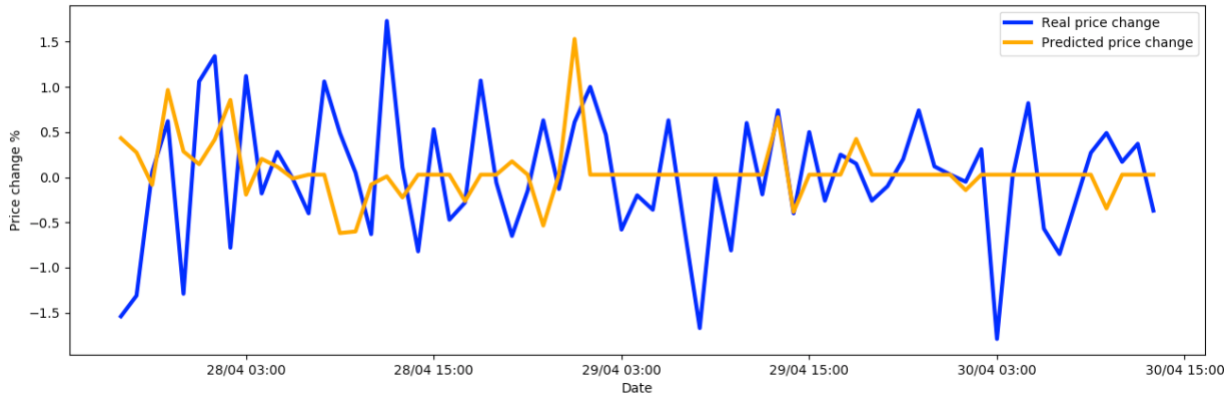

Figure 71: MLP - 60 minutes timeframe. Source: our own[83]

[82] Appendix H. full size
[83] Appendix I. full size

While testing MLP models (Figure 70 and Figure 71), it seemed that number of epochs did not have much influence on MSE score, however, with more epochs, prediction graph would fit real price changes better. Even if some predicted spikes look similar to price movements, it seems that there was little predicting power in MLP.

Model	**Long Short-Term Memory**
Shift	1
Epochs	10000
Batch size	72
Loss func.	MAE
Optimizer	Adam
MSE	**0.46 [15 min]** **0.65 [60 min]**

Figure 72: LSTM network config. and results. Source: our own

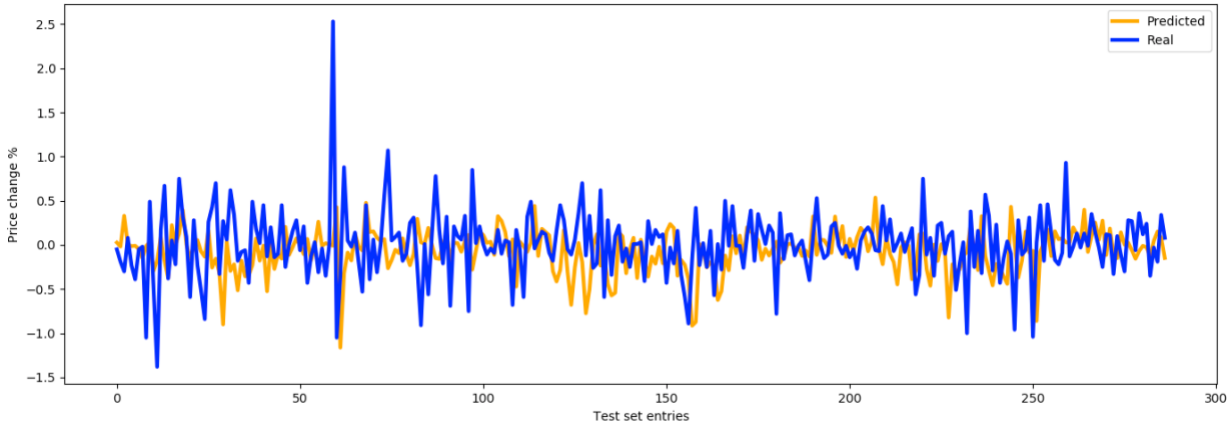

Figure 73: LSTM - 15 minutes timeframe. Source: our own[84]

[84] Appendix J. full size

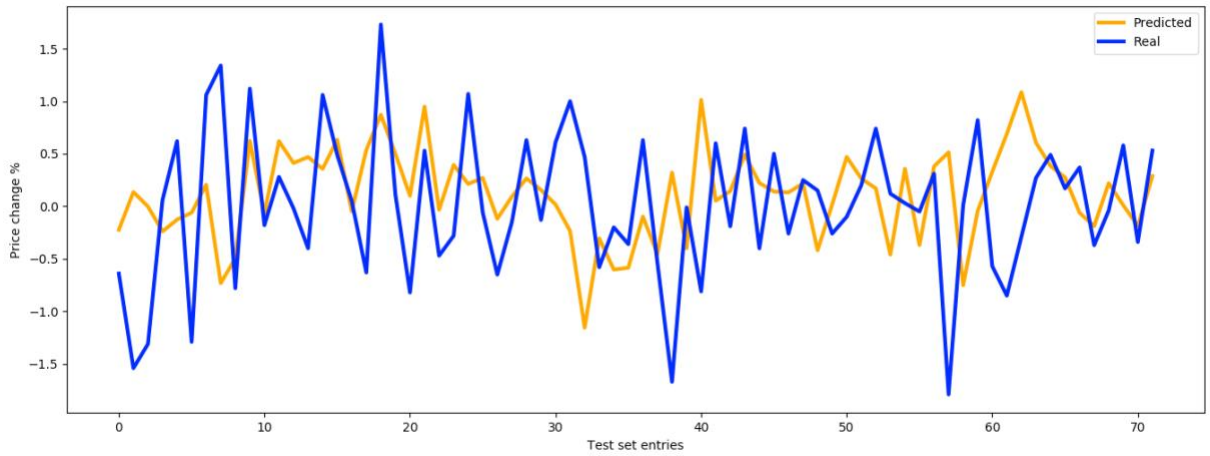

Figure 74: LSTM - 60 minutes timeframe. Source: our own[85]

LSTM produced similar MSE scores like other models (Figure 72), however both prediction graphs of 15 (Figure 73) and 60 (Figure 74) minutes seems to have a lot of correlation with real price changes, which is promising. 15-minute timeframe also produced low MSE scores both on training and testing datasets, which indicates that 10.000 epochs did not cause overfitting. Larger number of epochs proved to fit better real price changes, same as with MLP model above.

To sum up different machine learning models testing - it can be said that predicted results very vaguely show some correlation and seem not to be completely random. In particularly 60-minute timeframe seemed to outperform 15 minutes, so even larger timeframe could be tested in future iterations. In general, it is difficult to evaluate the performance of the training because of lack of knowledge and time for this project. However Discussion chapter (7) will go into how prediction capabilities can be improved in the future.

[85] Appendix K. full size

7 DISCUSSION

This chapter will go through some of the decisions that were made, and what it meant for the result of the project. It will also go through some changes that might have happen in the next iteration or ideas that could improve the prototype.

7.1 DISTRIBUTED CRAWLER

The conclusion of many research papers presented in State of the Art (3.2) is that they could have used more data. Therefore, the next iteration of prototype should focus heavily on increasing the quality and amount of data.

Current version of distributed crawler involves many manual tasks such as downloading all the databases from different servers and combining them. Each VPS had to be setup manually from snapshot. The monitoring and control is not centralized, therefore it is necessary to connect to each server to make sure that crawler is running and was started with correct parameters. The progress for each crawler is not clearly shown, therefore checking the logs is only way to make sure that crawling task is completed. To solve these issues the crawler controller can be separated from child processes. The main controller could distribute the tasks to child processes, allocate resources by requesting required number of servers from Digital Ocean through API. The monitoring and database should also be centralized.

Another issue is that real-time tweets are collected as soon as they are posted, this choice restricted the prototype from gathering vital data such as Likes and Dislikes (read more in 4.1). Likes and dislike are live data, and are accumulated over time, at the point of creation, there are none of them. Argument can be made that newly created post would have no readers yet, this would also mean that the message at this point has had no impact on the world yet. A way to get around the problem is to delay the prediction, by waiting with the crawling until the post is a few days old. This way it allows for users to read and dislike/like it, giving part of the live data.

Collecting historical data from social media could be incorrect if we use the number of followers the person has at the time of crawling data instead of the time when the message was posted. Especially if the timeframe is as large as a few years. This problem could be solved by using other services which track historical follower growth of Twitter accounts or by estimation.

Through the process of crawling the data, a pattern was shown, that tweets were created more and less in specific times on the day. The pattern was that tweets were being created more in the morning compare to noon, this could have influenced the 15 minutes of tweet aggregation that were done, because 15 minutes of tweet aggregation in the morning would have much fewer tweets than 15 minutes in the noon. Another argument was the psychology factor, with more positive tweets being sent in the morning and more negative in the night, in correlation with cloudy day having impact on people being more negative than on a sunny day [25].

Overall the crawler for first prototype has performed well, no problems preventing larger scale data collection have been found.

7.2 Predictions and Models

Some of the few things that could have improved on the result, is by having more data to work and train with. This could have been done by crawling more data from multiple social media platform, however a more important aspect was to gather more data that are meaningful and useful, instead of having to remove 2,5 million tweets out of 5 million, because half of the data being "empty". After reading through the data, it was seen that half of the tweets were made by bots with a hyperlink and hashtag attached to the tweet [24]. With the processing of the data to clean it, the hyperlinks would be removed since they are not useful for the sentiment score, which led to a tweet that are completely empty. Another project [1] managed to avoid a big part of it, by manually looking into hundred thousand of tweets to locate the users who created these bot tweet messages and blacklisted them from their crawler. Another way to increase the data gathered, were to not only crawl the data which had specific hashtags related to bitcoin, but other words that might indirectly effect bitcoin, such as news or big events as mentioned in 4.1. These words could be found using TFIDF, to see which word were common and use them as both hashtag to crawl for more data and features to weight if the word would be good or bad for the bitcoin.

VADER were tested with random tweets to see how it performed, however that was an oversight, since the score received from the testing were VADER's score on the sentiment of the tweets. Since the tweets were randomly selected, it was impossible to identify how accurate VADER's sentiment score was, except for other projects (section 3.2) saying it was a competent tool. This could had been avoided or rather self-tested by running it through known tweets to see if the score matches up with what the expected outcome would have been, such as a high

positive tweet to be close to a high score while a highly negative tweet would be as low as possible etc.

A decision to use workbench was made early in the process, however that proven to be a mistake. Workbench had problems communicate what it was doing, during high amount of data input. This became a problem when waiting for the data to become processed only to find out later on, that it had crashed or did nothing for the last few hours.

8 CONCLUSION

To conclude this project, it is needed to re-connect with the research question (see in 1.1) and evaluate the overall progress of this research. This paper studied whether Bitcoin price change can be predicted using machine learning approach on historical price and sentiment data from social media and news sources. Research sub-problems were concerned with the whole process from data collection, processing to price change prediction and evaluation.

As can be found in Testing chapter (6), the prototype that was built for this experiment has not shown a lot of predictive power. However, it was only the first iteration of experiment, which can be improved drastically. Every step from data collection to training must yield higher performance in order to produce high prediction accuracy. All *Must* requirements from *MoSCoW* table (see in 4.5) were implemented, and test cases showed good results (in 6.1), however lack of knowledge and time blocked from achieving stable predicting power in the first iteration.

On the other side, building complete and production ready ML model was not the main goal of this study and the conclusion is different from the results of first prototype. In-depth State of the Art analysis (chapter 3) produced important knowledge about the research area. Quite few research groups have trained models with sentiment data from either Twitter or other social media/news sources and successfully predicted prices or price changes (accuracy 70 – 90%). This helped to build baseline workflow for data collection, processing and feature extraction with steps that were tested and proved by other researchers (sub-questions in 1.1).

This paper confirms that there is a correlation between sentiment changes and price movements. However, it also shows that it is very difficult problem and there is no right way to achieve high prediction accuracy. Low-level machine learning knowledge must be gathered to properly evaluate the models and achieve better results in next iterations. Some of the difficulties in the process and possible improvements can be found in previous Discussion chapter (7).

9 REFERENCES

[1] E. Stenqvist and J. Lönnö, "Predicting Bitcoin price fluctuation with Twitter sentiment analysis," Stockholm, 2017.

[2] K. YB., K. JG., K. W., I. JH., K. TH., K. SJ. and e. al., "Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies," PLoS One, 2016.

[3] D. Garcia, C. Tessone, P. Mavrodiev and N. Perony, "The digital traces of bubbles: feedback cycles between socio-economic signals in the Bitcoin economy," *Royal Society Interface,* vol. 11, 2014.

[4] K. L., "BitCoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the Internet era," Scientific reports, 2013.

[5] https://bitcointalk.org/, "Bitcoin Forum: Simple Machines," 30 March 2016. [Online]. Available: https://bitcointalk.org/ . [Accessed 30 March 2016].

[6] forum.ethereum.org, "Ethereum," 30 March 2016. [Online]. Available: forum.ethereum.org. [Accessed 30 March 2016].

[7] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[8] G. CW., H. B-N. and Y. C-W., "A bivariate causality between stock prices and exchange rates: evidence from recent Asian flu," *The Quarterly Review of Economics and Finance,* vol. 40, no. 3, pp. 337-354, 2000.

[9] W. GI., B. JR. and W. Z., "Not so naive Bayes: aggregating one-dependence estimators," *Machine Learning,* vol. 58, pp. 5-24, 2005.

[10] M. Makrehchi, S. Shah and W. Liao, "Stock Prediction Using Event-based Sentiment Analysis," in *2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT)*, 2013.

[11] B. O'Connor, R. Balasubramanyan, B. R. Routledge and N. A. Smith, "From tweets to polls: Linking text sentiment to public opinion time series," 2010.

[12] T. Wilson, J. Wiebe and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Human Language Technology and Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, 2005.

[13] S. Asur and B. A. Huberman, "Predicting the future with social media," in *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Washington, DC, USA, 2010.

[14] L. Hong, O. Dan and B. D. Davison, "Predicting popular messages in twitter," in *20th international conference companion on World wide web, WWW '11*, New York, NY, USA, 2011.

[15] J. Bollena, H. Maoa and X. Zengb, "Twitter mood predicts the stock market," *Journal of Computational Science,* 2011.

[16] R. Bandari, S. Asur and B. A. Huberman, "The pulse of news in social media: Forecasting popularity," in *CoRR*, 2012.

[17] E. Gilbert and K. Karahalios, "Widespread worry and the stock market," in *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[18] X. Zhang, H. Fuehres and P. A. Gloor, "Predicting stock market indicators through twitter "I hope it is not as bad as I fear"," in *The 2nd Collaborative Innovation Networks Conference - COINs2010*, 2010.

[19] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff and S. Patwardhan, "OpinionFinder: a system for subjectivity analysis," in *HLT/EMNLP on Interactive Demonstations*, Stroudsburg, PA, USA, 2005.

[20] D. McNair, J. P. Heuchert and E. Shilony, "Profile of mood states," 1971.

[21] J. Rocchio, "Relevance Feedback in Information Retrieval.," 1971.

[22] R. B. Go and L.Huang, "Twitter Sentiment Classification Using Distant Supervision," Stanford University, 2009.

[23] L. Barbosa and J. Feng, "Robust Sentiment Detection on Twitter from Biased and Noisy Data," *COLING,* pp. 36-44, 2010.

[24] V. A. Kharde and S. Sonawane, "Sentiment Analysis of Twitter Data: A Survey of Techniques," *International Journal of Computer Applications,* 2016.

[25] M. Nofer, The Value of Social Media for Predicting Stock Returns, Darmstadt, Germany: Springer, 2015.

[26] K. XIAO, Q. LIU, C. LIU and H. XIONG, "Price Shock Detection With an Influence-Based Model of Social Attention," 2017.

[27] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press., 2008.

[28] R. Pimprikar, S. Ramachandran and K. Senthilkumar, "USE OF MACHINE LEARNING ALGORITHMS AND TWITTER SENTIMENT ANALYSIS FOR STOCK MARKET PREDICTION," *International Journal of Pure and Applied Mathematics,* vol. 115, 2017.

10 ABBREVIATIONS

AdaGrad - Adaptive Subgradient Methods

Ads – Advertisements

Adam - Adaptive Moement Estimation

AI - Artificial Intelligents

AIM - Affect Infusion Model

AODE - Averaged One-Dependence Estimators

API - Application Programming Interface

AWS - Amazon Web Services

BTC – Bitcoin

CCXT - CryptoCurrency eXchange

CMC - Coin Market Cap

CPU - Central Processing Unit

CV - Cross Validation

EE – Easter Egg

GB - Gigabyte

GPU - Graphical Processing Unit

GUI - Graphical User Interface

HTTP - HyperText Transfer Protocol

IBM - International Business Machines

ICO - Initial Coin offering

ID - Identity

IP - Internet Protocol

KTH Royal Institute of Technology - Kungliga Tekniska Högskolan Royal Institute of Technology

Log - Logarithm

LSTM - Long Short-Term Memory

Max – Maximum

MB - MegaByte

Min – Minumum

ML - Machine Learning

MoSCoW - Must, Should, Could, Won't

MSE - Mean Squared Error

Neg - Negative

Neu – Neutral

NLP - Natural Language Processing

OHLC - Open High Low Cost

OHLCV - Open High Low Close Volume

PCDSA - Periodic Cumulative Degree of Social Attention

POMS - Profile of Mood States

Pos – Positive

PoS - Proof of Stake

PoW - Proof of Work

RAM - Random Access Memory

ReLU - Rectified Linear Unit

SGD - Stochastic Gradient Descent

SMI - Sentiment Mood Index

SoTA - State of The Art

std - Standard deviation

SVM - Support Vector Machine

S&P - Standard and Poor

TB - Terabyte

TF-IDF - Term Frequency-Inverse Document Frequency

TU Darmstadt - Technical University Darmstadt

TWINT - Twitter Intelligence Tool

URL - Uniform Resource Locator

USD – United States Dollar

USDT – United States Dollar Tether

UTF-8 - Unicode Transformation Format 8

VADER - Valence Aware Dictionary for sEntiment Reasoning

VPS - Virtual Private Server

XBT – Bitcoin