TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Marko Kaar 177030IAPM

MAPPING SUMMARIES OF ECONOMICAL NEWS TO CHANGES IN THE STOCK MARKET USING NATURAL LANGUAGE PROCESSING

Master's Thesis

Supervisor: Ahti Lohk
PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marko Kaar 177030IAPM

MAJANDUSUUDISTE KOKKUVÕTETE SEOSTAMINE AKTSIATURU KÕIKUMISTEGA KASUTADES LOOMULIKU KEELE MASINTÖÖTLUST

Magistritöö

Juhendaja: Ahti Lohk
PhD

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Marko Kaar

07.05.2019

Abstract

With great advancements in computer technology, a plethora of different fields have integrated computers in their day-to-day work. The financial sector is an excellent example of this, taking advantage of Moore's law. The noticeable growth in processing power per and the inexpensiveness of data storage, along with data availability, and electronic interconnectivity have noticeably changed the way financial markets operate [1].

Technology is used to try and predict the future of stock prices and there's plenty of software that visualise changes in the stock market, however there aren't many tools that, in simple terms, point out the possible associations of summarized news articles to changes in a company's stock price. The main goal of this work is to present a robust method for finding the possible associations of summarized news articles to changes in a company's stock price and presenting them to potential novice investors in a simple fashion. The method will include building a machine learning model which analyses sentiment data of summaries of news articles. The initial training set will be labelled by the author and the results will be compared to pre-labelled non-domain data. As a result, the data collected can be used to add additional dimensions to prediction algorithms in the future and potential new investors that use the product created in this thesis might have a better overview of why stock prices change. The method will be validated with an experiment where a random subset of changes in stock prices are evaluated by the author.

This thesis is written in English and is 52 pages long, including 6 chapters, 13 figures and 11 tables.

Annotatsioon

Majandusuudiste kokkuvõtete seostamine aktsiaturu kõikumistega kasutades loomuliku keele masintöötlust

Tänu suurtele edasijõudmistele arvutitehnikas on paljud erinevad sektorid integreerinud arvutid oma igapäevatöösse. Finants sektor on suurepärane näide sellest, kasutades ära Moore'i seadust. Arvutite võimekuse jõulise kasvu, andmete kättesaadavuse, ning arvutite omavahelise ühenduvusega on drastiliselt muutunud see, kuidas finantsturud töötavad [1].

Tehnoloogiat kasutatakse palju selleks, et proovida ennustada aktsiaturgudel tulevikku. Eksisteerib omajagu erinevaid tarkvaralisi lahendusi, mis visualiseerivad muudatusi aktsiaturul, kuid hetkel puuduvad head lahendused, mis väga lihtsal viisil, ning ka kõige võhiklikumale kasutajalegi ütleks, mis on põhjus mõne ettevõtte aktsiahinna tõusule või langusele. Töö autori põhieesmärk on pakkuda välja esmane meetod aktsiahindade muutuste potentsiaalsete seoste tuvastamiseks uudiste artiklitest, ning nende esitlemine viisil, et ka algaja aktsiatega kaupleja sellest informatsioonist aru saaks. Meetodis ehitatakse masinõppemudel, mis hindab uudiste artiklite kokkuvõtete tonaalsust. Treeningandmestik hinnatakse töö autori poolt ning selle tulemusi võrreldakse andmestikuga, mis ei ole konkreetse domeeniga seotud. Töö tulemusena, andmed kogutakse kesksesse allikasse, ning neid võiks saada tulevikus kasutada ka ennustavate masinõppealgoritmide sisendiks kasutada ning algajad investorid kes töös valminud toodet kaustavad võiksid saada parema ülevaate sellest, miks aktsiate hinnad muutusid. Töös pakutud meetodit valideeritakse eksperimendi käigus, kus juhuslikku alamhulka aktsia hindade muudatustest ning nende põhjustest hinnatakse autori poolt.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 52 leheküljel, 6 peatükki, 13 joonist, 11 tabelit.

List of abbreviations and terms

TA	Technical Analysis
FA	Fundamental Analysis
SVM	Support Vector Machine
ETL	Extract, Tranform, Load
NLP	Natural Language Processing
HTML	Hypertext Markup Language
NER	Named Entity Recognition
TF-IDF	Term Frequency – Inverse Document Frequency
S.E.C.	Securities and Exchange Commission
NLTK	Natural language toolkit
NASDAQ	National Association of Securities Dealers Automated Quotations

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Background

Predicting the stock market is extremely difficult due to the massive number of variables and factors involved. In a broad sense analyzing stock markets falls into two categories – Fundamental Analysis and Technical Analysis [2]. When looking at the profitability of a company and the future of their stocks based on the current financial performance, one deals with Fundamental Analysis. When finding trends from statistical figures the analyst deals with Technical [2]. In the world of machine learning, the more relevant of the two is Technical Analysis and this thesis, although not exactly following the practices of TA, will be more related to its outcomes than FA.

Technologically, the solution to associating changes in stock prices to summaries of news articles boils down to text classification, sentiment analysis and storing the processed data in a way that it can be retrieved efficiently later on. The author's hypothesis is that when stock prices drop, one can find articles that have a negative sentiment and when stock prices rise articles with a positive sentiment will appear.

1.2 Motivation

Understanding how the stock market works is an important part of being an investor. Knowing when to invest into a stock and when to sell can make the difference between being financially successful or financially ruined. Unfortunately, many novice investors don’t always have access to this kind of information.

As there are talks of the Estonian government planning to reform the second pension pillar, a big part of the responsibility of securing their future lands into the hands of each Estonian who decides to withdraw their second pension pillar to their investment account. There has been a large movement of people taking responsibility of their finances, with plenty of books being written by Estonians about investing, several blogs and groups existing that discuss the so-called financial independence target. These factors bring on new novice investors to the market that might know some theory about investing, but perhaps lack the tools to understand why changes happen.

1.3 The goal

The goal of this paper is to develop a robust method for linking summaries of news articles to changes in stock prices whilst comparing different machine learning tools, using APIs to gather data and present the outcome to the end-user in a simple and understandable fashion. During this process one of the questions that the author will investigate is how much result differ when sentiment analysis is performed on the full text of an article compared to the summary of the article. The author will label a subset of article sentiments and the result of the analysis will be compared to a pre-labelled dataset of movie reviews, investigating whether the contents of the training data make a difference in the model's accuracy. To the author's best knowledge, there hasn't been noticeable research into sentiment analysis of texts summarized by NLP methods.

When a customer searches for a name of a company in the user interface of the thesis outcome and selects a timeframe of events, the software should recognize whether the stock price increased or decreased and provide links to news articles that contain information about the fluctuation. The second functionality would be to alert the customer when news articles with a noticeably bad sentiment appear, giving the customer a heads up about stock prices falling.

The main challenges solved in the thesis are gathering test data with a custom crawler, implementing the keyword extraction, summarization, and sentiment analysis algorithms, and finding a suitable solution for data storage. It is clear that the data that's scraped by the news portal crawler will need to have some sort of structure to it, otherwise querying it later on will be very difficult. The data will be scraped, analysed and saved to a NoSQL database throughout the several phases of parsing. The algorithms for text classification and sentiment analysis will be written in Python, whilst querying and displaying the data will be written in Java with the frontend completed using the Angular framework. There will be various Python frameworks that will be used for data.

1.4 Research methodology

A combination of data will be analysed to provide valuable information to the end-user. Different approaches can be implemented to acquire meaningful outcomes of the analysis process such as the Naive Bayes' Theorem, Support Vector Machines, and multinomial regressions to name a few. Furthermore, Deep Learning has been validated to have improved performance as dataset sizes grow [3]. The aforementioned algorithms will be considered, as with smaller datasets the complexity of setting up prototypes will not require as much effort.

The question that arises when looking at the problem statement is how to know whether the analysed data is reliable. As mentioned earlier, to use some of the approaches described either pre-labelled datasets have to be acquired or a fair bit of manual work will have to be done to label the data that's collected. Building a classification model from the pre-labelled data and finding the accuracy will have to be investigated.

Two datasets will be used to analyze the sentiments of articles. The first training dataset will contain data from movie reviews, labelled as to being either positive or negative. The second dataset will consist of news article summaries labelled by the author. An investigation will be done to which training-test data split is the most optimal for classifying the sentiments with the aforementioned algorithms. From this it is visible how well the sentiment analysis algorithm works, as the software will provide a percent value for accuracy and precision.

Combining the precision value with the precision of the sentiment analysis result the precision of the product can be found as to how well it performs in providing connections between stock market changes and news article data. Previous research has shown a classification accuracy between 50% to 79% [4]. If the precision of the classification model is below 50%, then the product is largely unusable. Therefore, it has to be made sure that the combined accuracy of the classification and sentiment analysis is around the 70th percentile.

1.5 Thesis outline

In chapter 2, an overview of the current methodologies will be described, providing information about current areas of research and their findings.

In chapter 3, an analysis of different sentiment analysis methods will be provided with a comparative study between the main algorithms used in the thesis according to the research and knowledge gathered by the author.

In chapter 4, a detailed overview of the steps to link news articles and their sentiments proposed by the author will be described and explained.

In chapter 5, the experiment conducted to evaluate the results will be analysed.

In chapter 6, a summary of the findings will be presented along with the conclusive course of action to follow when developing the outcome of this thesis further on.

1.6 Summary of results

As a result of this paper, a fully functional news article summary analysis tool will be created. Furthermore, an analysis of different machine learning algorithms for sentiment analysis will be performed and evaluated, giving reasoning as to why the selected classification algorithm is the best algorithm to use when analysing sentiments of news articles, along with an analysis of the datasets used to train the sentiment analysis models. In addition to this, an analysis of summarization is performed, investigating whether it's possible to perform sentiment analysis on texts summarized by NLP models, without a noticeable loss of accuracy. A method is proposed for gathering and displaying the data to potential customers of the product. The method is evaluated with a random seed experiment, where a set of random news articles will be evaluated by the author, seeing whether the algorithm matches the sentiment that the author derives from the articles and seeing whether the sentiments match the changes on the stock market.

2 Background and Literature Overview

Attempts of predicting stock markets using sentiments derived from unstructured texts, whether it be social media or news articles is not necessarily novel idea. There are several papers that have attempted to perform such market prediction, mainly using the bag-of-words approach or the Naïve Bayes algorithm [3]. This paper combines several different natural language processing methods

2.1 Dataset Description

When investigating text classification problems, finding labelled documents can be fairly challenging as documents that have adequate labels are difficult to find whilst there is usually an abundance of data that is unlabelled. Labelling data by hand tends to be a time-consuming task, as the amount of labelled data needed for an accurate model is usually too large to get a working prototype. There are two pieces of data that are used for this thesis. The first is the training data used for training the sentiment analysis algorithms. For this procedure the author has chosen a dataset of movie reviews [6], which was a prelabelled dataset available on the internet with an even split of 2500 negative and 2500 positive sentiments as its labels. As previously mentioned, the author will review around 200 article summaries, creating 50-50 split in positive and negative articles. During the thesis the author is comparing the precision, recall and F-scores of each of the algorithms for the specific datasets. The data analyzed is a collection of news articles that are related to companies traded on the NASDAQ stock exchange. The data is scraped from different publishers by a python script written by the author in the process of this thesis.

Fetching news articles can be challenging, as news sites don’t have a clear index to them. It would be possible to try and navigate from article to article, following links and related stories within an article, however it’s also possible to end up in an infinite loop. As articles play a big role in the thesis, the author has decided to simplify the process of acquiring articles by using News API. News API is a simple HTTP REST API for searching and retrieving articles from various news publishers on the web in real time [4].

2.2 Sentiment analysis

Sentiment analysis is an area that has been research since the early 2000s. It has been an area of computer science that has been growing at an exponential speed, hence keeping up with all the activities in the area can be challenging. When initially sentiment analysis was more related to analysing products, in recent years the focus has moved to social media such as Twitter and Facebook. It has been used in detecting the popularity of politicians along with many other similar market analysis applications. Alongside this, topics such as product reviews, stock markets, elections and many other utilize sentiment analysis [5].

Due to its popularity, sentiment analysis has been applied to predicting the price changes of certain stocks using different machine learning algorithms such as self-organizing fuzzy neural networks [8] and others, however this paper differs from the aforementioned article and other similar sentiment analysis research articles [9] [10] due to the fact that it does not plan on predicting the price of a stock, but rather trying to display the possible associations of the change after the fact.

2.3 Data storage

Working with large amounts of texts and querying data from them is a challenge that many companies face. As unstructured data has become the norm in today's society, several different methods of data storage for this type of data are available. Companies tend to turn towards data redundancy and keeping multiple copies of documents with complex solutions such as data lakes or complexly built relational databases. For this paper, there are two criteria for data storage – the data must be easily storable and queryable, and the response time must be short without putting a lot of effort into indexing the data. MongoDB is a flexible NoSQL database that has fits these criteria as it's lightweight, easy to set up and usable in both Python and Java, which are the two main programming languages used for this thesis.

3 Comparative study of sentiment analysis algorithms

Sentiment analysis is a field of study in natural language processing that analyses people's sentiments, opinions, and emotions from text-based documents. In fact, sentiment analysis is one of the most popular research in natural language processing and is also very relevant and deeply studied in the context of data mining, in both web-based texts and any other parseable documents. Research into sentiment analysis has diverged to areas outside of computer science. As business and society play a big role in the applications of sentiment analysis, among other management sciences [8]. As data has become more available with social media platforms growing, such as product reviews on Amazon, forum discussions, Twitter, Facebook, TripAdvisor and many others, in addition to blogs, micro-blogs and more the analysis of it all has become more and more important. Big Data has become a standard in the world of computers and the huge volume of opinionated data is perfect input for sentiment classification algorithms.

There are two main ways that sentiment analysis is implemented throughout the thesis. The first method being Bayes classifiers, which can be used to rate the article's positivity using a pre-trained dataset of reviews. During the thesis a dataset of movie reviews [12] was used and other works have used training data provided by Yelp [9] to train their sentiment analysis models, however for this paper this data was not used. Furthermore, SVMs can also be used for sentiment analysis and in most cases, they give better accuracy compared to data analysed with Naive Bayes [10].

For this thesis the main effort for sentiment analysis will be done using SVMs, however the Naïve Bayes algorithm will be evaluated to see how well the algorithms perform with the provided training data.

3.1 Naïve Bayes

Naïve Bayes is a probabilistic machine learning algorithm that is mainly used for a wide range of classification tasks such as spam filtration, sentiment analysis and in the case of this thesis - classifying news article. The Naive Bayes algorithm makes the assumption that the occurrence of a certain feature is independent of other features. This means that, as an example, if an article is about basketball and contains the words crowd, hoops, and arena, then regardless of all of the other words in the article, all these words contribute to the article being about basketball. The equation for the Naive Bayes assumption is as follows [11].

$$P(x|y\ =\ c) = \prod_{i=1}^{D} P(x_i|y)\ =\ c$$

This is the Naive Bayes assumption which, unfortunately, isn't true in the most real-world situations. Despite this, Naive Bayes often performs classification very well [12].

The Naive Bayes algorithm is based on the Bayes Theorem, in which the probability of an event A happening can be found, given that an event B has occurred, also known as *posterior probability*. This can be calculated by finding the probability of event B occurring, given that event A has occurred multiplying it by probability of event A occurring, and dividing the product by the probability of event B occurring.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

The Naive Bayes algorithm has to be used to understand what sector the article is best suited towards. The sectors will be predefined according to the sectors and industries registered in.

How this works from a technical standpoint is that first the words in the article will have to be tokenized after which the data gets placed into a data frame and the data is normalized. During normalization, a lemmatized list of words is found. After this step stop words should be removed. A stop list, or negative dictionary is a device used in automatic indexing to filter out words that would make poor index terms [13]. Generally, stop lists include the most frequently occurring words of a specific dictionary. There are several different language dictionaries implemented in different natural language processing frameworks such as Spacy. In practice, however, stop lists have tended to include infrequently occurring words, and have not included many frequently occurring words [13].

After filtering the words in the stop list a cleaned data frame is created and a model can be trained. The model will vectorize the cleaned data using Term Frequency - Inverse Document Frequency (TF-IDF).

TF-IDF works by, essentially, finding the relative frequency of words in a text, comparing it to the inverse proportion of that specific word over the entire dictionary of documents that have been collected into a corpus [14].

When the model has done its work, the result will be saved into the NoSQL database in JSON format, so that sentiment analysis can be run on the now improved dataset.

3.2 Support Vector Machines

Support vector machines (SVMs) have in the past shown themselves to be very effective at traditional text categorization, outperforming Naïve Bayes is a plethora of applications [15]. SVM is one of the most popular classification techniques which aims to minimize the number of misclassification errors directly. An SVM model is a representation of the documents as points in space. This can be compared to principal component analysis, which is a topic of statistical multivariate analysis, where the points are mapped so that a maximally large gap is created between the categories. For all new documents a prediction is made, placing the document onto the same space, checking into which classification they fit better [16]. As seen on Figure 1, the SVM model works in a two-dimensional space where each axis is a specific feature. The model compares all the features, two features at a time and then draws conclusions from the data. If there are several hyperplanes that can classify the data, then the hyperplane which leaves the most margin from both classes is chosen.

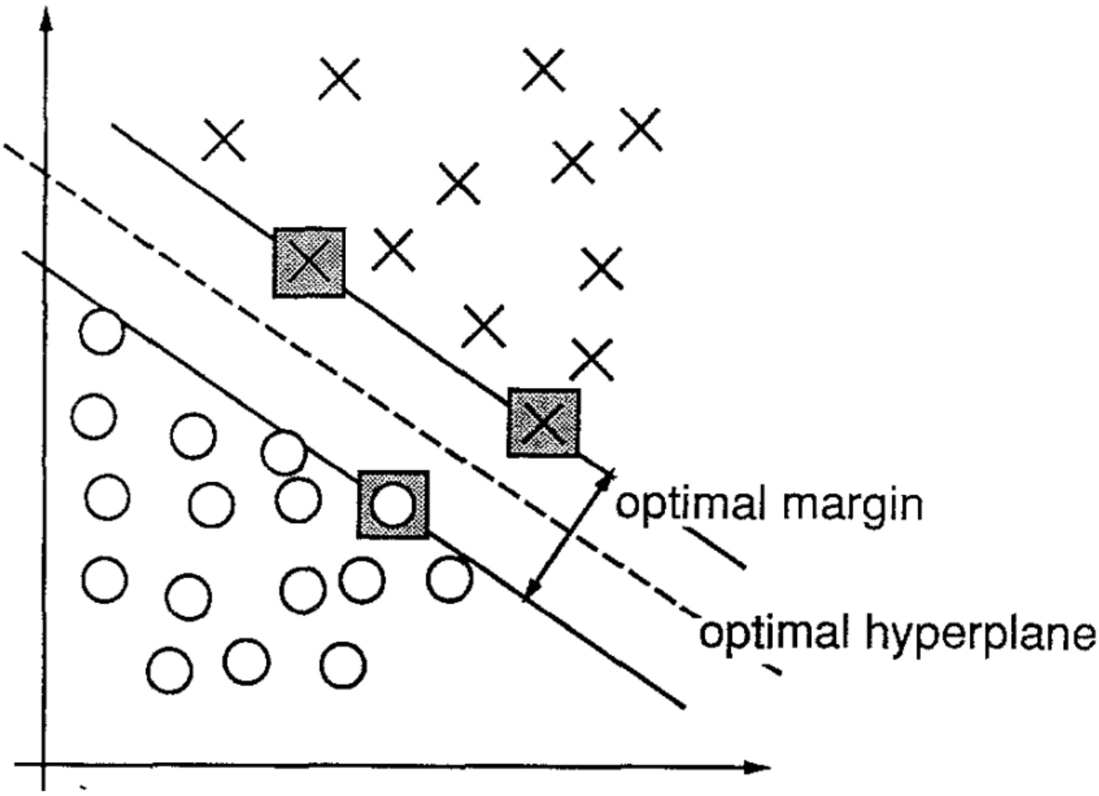
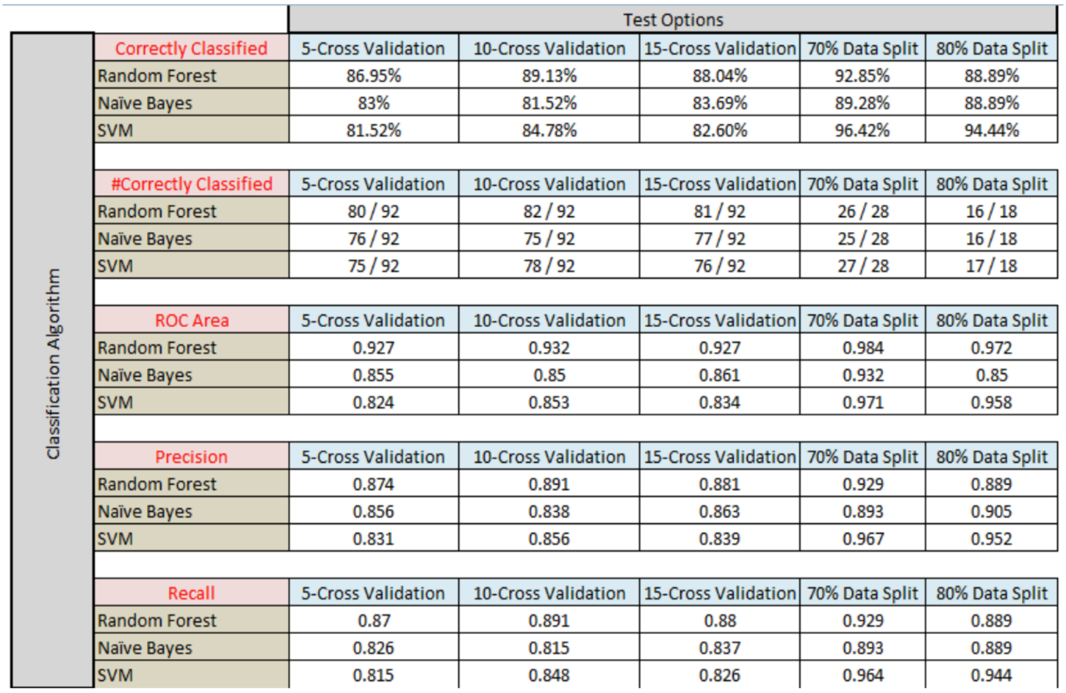


Figure 1 Visualization of a separation between two categories of data in a 2-dimensional space. The grey boxes mark support vectors – showing the largest margin between the data.

Source [17].

The procedure, as is standard in supervised machine learning tasks, consists of training a classifier on pre-tagged training data and then evaluating the performance of the classifier on a held-out set of test data [18]. When looking at an SVM, at its core, in its simplest, linear form, is a hyperplane that separates a set of values with maximum margin where the split region is called the margin. The samples by the margin are called the support vectors [19].

When comparing the accuracy of either of the algorithms, the trend tends to be that when there's more training data, and the training versus test data split is around 80%, the SVM is the most accurate algorithm to predicting sentiment.



		Test Options				
Classification Algorithm	Correctly Classified	5-Cross Validation	10-Cross Validation	15-Cross Validation	70% Data Split	80% Data Split
	Random Forest	86.95%	89.13%	88.04%	92.85%	88.89%
	Naïve Bayes	83%	81.52%	83.69%	89.28%	88.89%
	SVM	81.52%	84.78%	82.60%	96.42%	94.44%
	#Correctly Classified	5-Cross Validation	10-Cross Validation	15-Cross Validation	70% Data Split	80% Data Split
	Random Forest	80 / 92	82 / 92	81 / 92	26 / 28	16 / 18
	Naïve Bayes	76 / 92	75 / 92	77 / 92	25 / 28	16 / 18
	SVM	75 / 92	78 / 92	76 / 92	27 / 28	17 / 18
	ROC Area	5-Cross Validation	10-Cross Validation	15-Cross Validation	70% Data Split	80% Data Split
	Random Forest	0.927	0.932	0.927	0.984	0.972
	Naïve Bayes	0.855	0.85	0.861	0.932	0.85
	SVM	0.824	0.853	0.834	0.971	0.958
	Precision	5-Cross Validation	10-Cross Validation	15-Cross Validation	70% Data Split	80% Data Split
	Random Forest	0.874	0.891	0.881	0.929	0.889
	Naïve Bayes	0.856	0.838	0.863	0.893	0.905
	SVM	0.831	0.856	0.839	0.967	0.952
	Recall	5-Cross Validation	10-Cross Validation	15-Cross Validation	70% Data Split	80% Data Split
	Random Forest	0.87	0.891	0.88	0.929	0.889
	Naïve Bayes	0.826	0.815	0.837	0.893	0.889
	SVM	0.815	0.848	0.826	0.964	0.944

Figure 2 Comparison of three sentiment classifiers against different test options.

Source: [20]

Figure 2, shows that in previous research SVM has performed better in classifying sentiments correctly than the Naïve Bayes algorithm in tests where the test data was split 70% and above. Furthermore, the precision was better in those scenarios.

When looking at cross validations, the Naïve Bayes algorithm performs better in some scenarios, therefore a further investigation between the NB and SVM algorithms should be done using the dataset collected for this thesis.

The Random Forest classification algorithm wasn't investigated in this thesis and will not be addressed further.

When looking at the model training times of either of the algorithms the SVM model training time was around 160 seconds and for the Naïve Bayes model it was noticeably longer. For the dataset of 5000 movie reviews the Naïve Bayes algorithm didn't finish in a reasonable amount of time.

4 Method

In this section the author presents a viable method for parsing time-series data in the form of news articles, storing it, and performing text pre-processing and sentiment analysis on the texts. In addition, the author proposes a method for storing the data in a structured way, that can be queried later on. The method consists of the following parts:

1. Fetching articles using NewsAPI
2. Parsing articles and extracting keywords
3. Text summarization
4. Analysing sentiments

Acquiring data for classification is one of the main challenges as the final solution doesn't provide much value without up-to-date information. To retrieve the necessary data in the required format crawlers will first be developed that ingests news articles from publishers such as the New York Times, the Wall Street Journal, BBC, and TechCrunch. Access to some news outlets may be restricted to crawlers as the outlets have implemented CAPTCHA tests that can't be passed by the simple crawlers, however the outlets that can be crawled contain helpful metadata about the nature of the articles. As news outlets want to be found by search engines, the pages hold keywords about the article that is used for search engine optimization. These keywords can be helpful with categorizing the data. More information about the crawler will be provided in the methodology section of this paper. Initial examples will be set up using already existing datasets and the data that the crawler produces will be structured similarly to the existing datasets. Knowing whether the results are accurate or not may require some manual work and an experiment can be conducted, taking a random set of data and validating it. The goal of validation would be to get as high accuracy as possible within the provided time frame.

In general, there are two main flows to the thesis. The first flow contains scraping the data, analysing it, and saving it to a local database. The second flow deals with displaying the data to the customer. Figure 3 explains these flows in further detail.

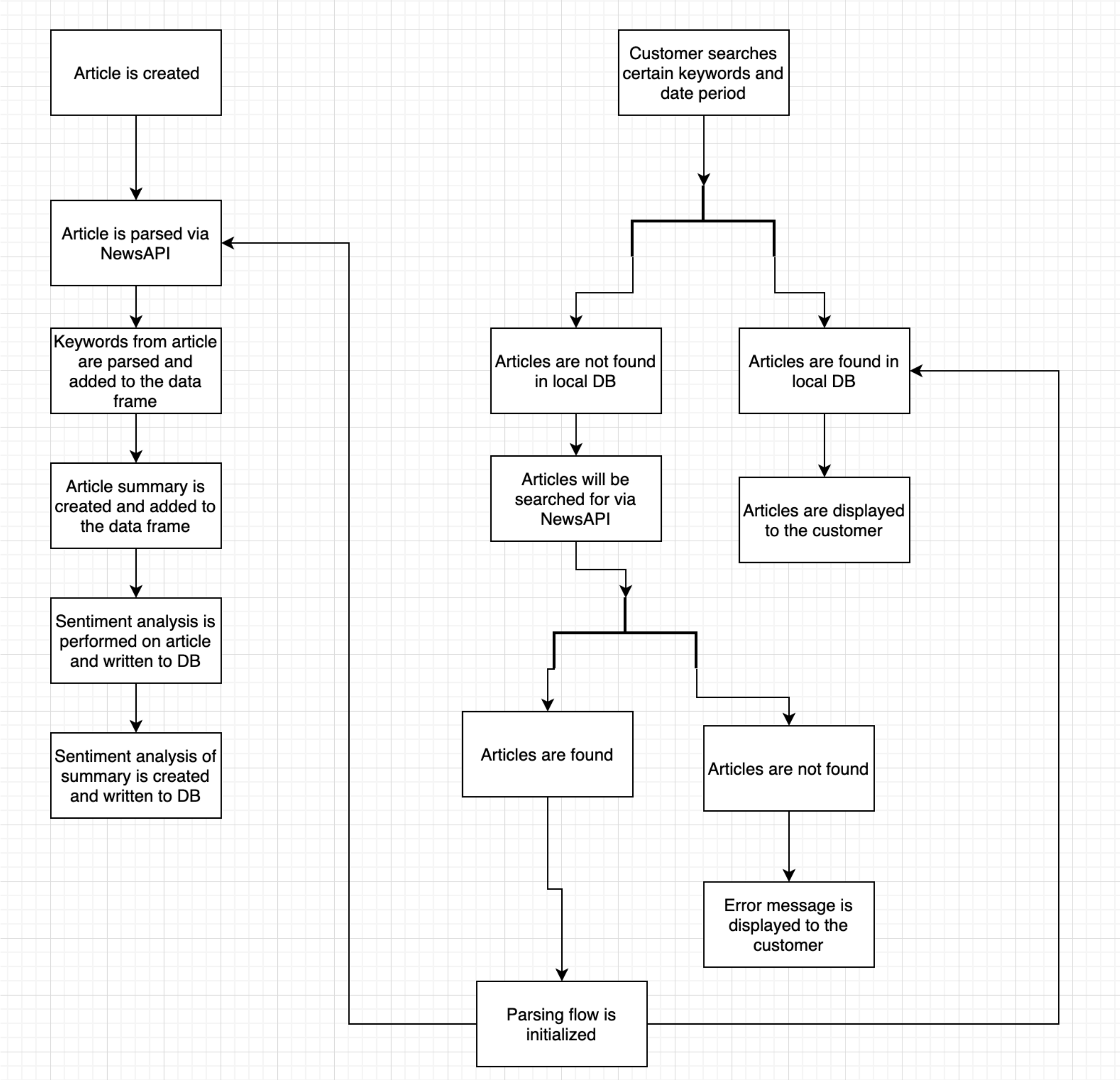

Article is created
Article is parsed via NewsAPI
Keywords from article are parsed and added to the data frame
Article summary is created and added to the data frame
Sentiment analysis is performed on article and written to DB
Sentiment analysis of summary is created and written to DB
Customer searches certain keywords and date period
Articles are not found in local DB
Articles are found in local DB
Articles will be searched for via NewsAPI
Articles are displayed to the customer
Articles are found
Articles are not found
Error message is displayed to the customer
Parsing flow is initialized

Figure 3 Description of the two main flows

When no relevant articles are found from NewsAPI, an error message is displayed to the customer, that there were no articles in the chosen news portals related to the keywords they entered.

4.1 Fetching articles using NewsAPI

To acquire data in a simpler fashion a news API will be used. As mentioned in the introduction, the API that will be used for collecting the initial list of articles is called NewsAPI. NewsAPI has a fairly expensive subscription plan for enterprise use, hence the free development plan was used. The development plan has a lot of limitations to its use such as limiting the number of requests to 500 requests per day, truncating the content of the articles, and delivering articles only a month old. Despite these limitations, it is a useful source of articles, as it allows specific sources to be defined, and that's one of the main criteria of this thesis – to retrieve articles from reliable news outlets.

The query to NewsAPI will return an array of news article objects containing information about the source of the article – which news outlet it originates from, the author of the article, the title of the article, a truncated description, a URL to the article, the date the article was published, and a truncated content of the article. Considering that News API has the limitations mentioned in the last paragraph, the author had to implement a parser for each of the news outlets. For simplicity's sake, we'll focus on the New York Times parser.

Each NewsAPI query returns a list of the latest 100 articles found in all of the news outlets specified in the query. These 100 articles get saved into a local MongoDB database. MongoDB allows saving the data in pure JSON form, as it is returned by NewsAPI. This type of data storage allows the author to further manipulate this data, without having to worry about losing the data. In each step of the sentiment analysis flow, a new table is created, similarly to an ETL process where a new stage is created with the modified data throughout the pipeline. Parsing articles is an infinite process, where a set of 100 articles is queried every 20 minutes. This fits into the 500 query per day limit, allowing customers to make an additional 428 queries for information that wasn't initially found in the database. The research showed that on average, approximately 6 articles are written every 20 minutes during the day times over all of the news outlets being monitored, although not all of them relevant to the change in price of stocks.

4.2 Parsing news articles and extracting keywords

Once the articles are saved into the local database a second script iterates through the articles, queries the news article site and downloads its full HTML code. All of the parsers are written in the Python language using different frameworks such as urllib and BeautifulSoup for web scraping. For keyword extraction tools such as nltk and pandas will be used. For sentiment analysis different NLP tools are used such as nltk, scipy, sklearn, numpy, pandas and many others, this subject will be discussed in a later chapter.

After the article HTML is downloaded, the initial data frame is built, including all the variables that will be included during the search. The data frame object gets passed through each stage of the pipeline and is enriched by relevant values on each step. From Table 1 the final result of the data frame can be seen.

Table 1 Columns of the database table for searching articles and sentiments

URL	Publishing Date	Keywords	Meta Keywords	Full text	Summary	Full Text Positive Sentiment Score	Full Text Negative Sentiment Score	Summary Positive Sentiment Score	Summary Negative Sentiment Score

The URL value is added at the first step of the pipeline, when all articles are iterated over, and the scraping pipeline begins. During the second step of the pipeline the keywords and the publishing date are extracted from the HTML code.

For keyword extraction the author has combined three methodologies. One is to parse it from the metadata of the article website, as in most cases it's included there for search engines to find relevant articles. In addition to this, automated keyword extraction on articles using NLP, strongly influenced by an article of the same name, will be performed.

Alongside the methodologies stated in the article, named entity recognition (NER) will be performed on the text, picking out people, organisations and products that might not be part of the most frequent. The purpose of this is to make sure that even if the metadata on the news portal is lacking, there are still keywords that can be searched when the customer makes queries. The high-level approach to keyword extraction can be seen on Figure 4.

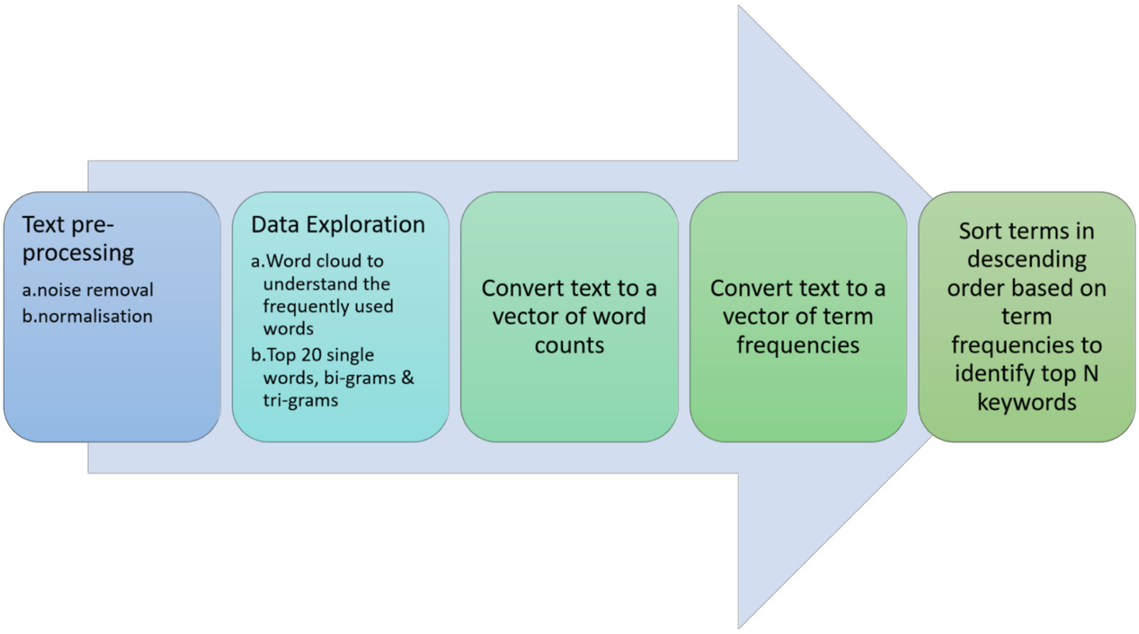

Text pre-processing
a.noise removal
b.normalisation
Data Exploration
a.Word cloud to understand the frequently used words
b.Top 20 single words, bi-grams & tri-grams
Convert text to a vector of word counts
Convert text to a vector of term frequencies
Sort terms in descending order based on term frequencies to identify top N keywords

Figure 4 High-level approach to keyword extraction from text.

Source: [25]

To display meaningful data in the context of this thesis, an article from the New York times will be analysed: https://www.nytimes.com/2019/04/03/business/tesla-sales-elon-musk.html. In this article the author explains that Tesla stocks have been on a decline, summarizing that the demand in electric cars in the United States has been more modest than predicted earlier, and that Tesla has produced less cars than predicted. The article also mentions Elon Musk and his challenges with the Securities and Exchange Commission. When looking at the most common words it is visible that most of them are what are called stop words. Stop words are words that are commonly used, such as any articles "the", "a", along with prepositions such as "in" and "to". In the specific article, the word counts appear as follows:

Table 2 Most common words in the article

Word	Word Count
the	37
in	19
to	19
of	17
a	12
and	12
Tesla	11

Table 2 shows, that Tesla, a key subject of the article, is only 7th in the list of most mentioned words. In order to avoid the keywords being words that don't hold value in the context of the article, pre-processing has to be performed on the text. The text pre-processing consists of four main steps. First noise removal will be performed on the text, meaning that all punctuations and HTML elements will be removed. Compared to the article by Vivek [25], this process is somewhat redundant, as during the scraping process the scraper only keeps texts that are part of the article. During the normalization process, data is split into tokens. For cleaning the split words in this thesis, the author has used lemmatization. Lemmatization is a technique that finds the base, dictionary reference, of a word. The purpose of lemmatization is to remove inflectional forms of a word. Finally stop words are removed, as shown on Figure 5.

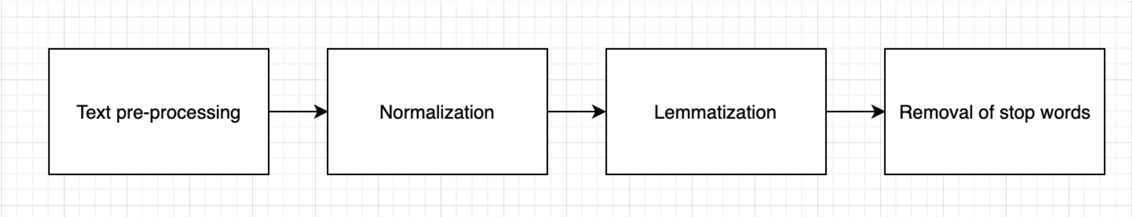


Figure 5 Text pre-processing flow

When stop words are removed, the list of most common words appears as follows:

Table 3 Most common words after removing stop words

Word	Word Count
tesla	13
car	10
musk	8
model	8
would	6
delivery	6
expected	5

The values represented in Table 3 are more relevant than the words with stop words. There are still, however, words that don't make too much sense in the context of keywords such as the verbs "would" and "expected". Furthermore, the keywords have no information about the S.E.C. that the article mentions. When creating a word cloud of the article, as seen on Figure 6, it is also visible that the more repeated words carry more weight, dealing with the delivery of the tesla models.

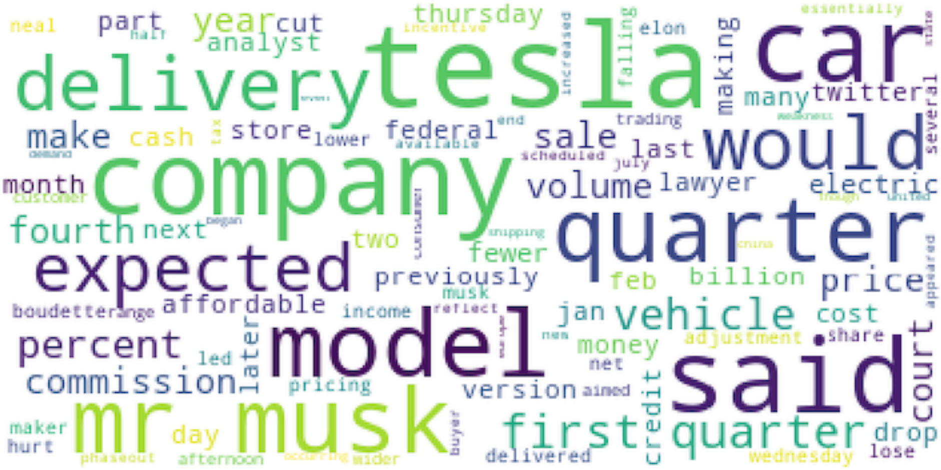


Figure 6 Word cloud of the most common words in chosen article

The final goal of text extraction is to find TF-IDF values for each of the words. TF-IDF is a numerical statistic that shows how important a word is in a document or a corpus. TF-IDF consists of two components. TF is term frequency, mathematically described as follows:

$$TF = \frac{Frequency\ of\ a\ term\ in\ a\ document\ or\ text}{total\ number\ of\ terms\ in\ the\ document}$$

IDF is inverse document frequency, described as follows:

$$IDF = \frac{\log\ (Total\ documents\ or\ texts)}{Number\ of\ documents\ or\ texts\ with\ the\ term}$$

TF-IDF scores don’t provide any meaningful results when being found for a single document, as they usually require a corpus of texts for the IDF value. When only one document is used then it can be seen that log(1) / 1 = 0, which gives no meaningful contribution to the calculation. At the time of writing this paper the article list was still being collected, therefore there wasn’t a great corpus of articles to compare against. To solve this problem the dataset of movie reviews used for the sentiment analysis was used for keyword extraction. Once the database of articles grows, the articles processed can be used as a reference in the future. There were 5000 movie reviews used for the TF-IDF corpus. The results contained more relevant keywords than simply using the word frequencies as can be seen in Table 4.

Table 4 TF-IDF results with scores

Word	TF-IDF score
tesla	0.689
model	0.282
car	0.264
delivery	0.204
percent	0.169
vehicle	0.141
expected	0.135
sale	0.127
volume	0.122
court	0.110

To complement the list of keywords acquired from the previous method, a very simple NER function was created using spacy. Spacy is a python library that is used for advanced natural language processing. Spacy has several statistical neural network models for the English language, allowing for software to recognize people, organizations and even products from unstructured, random texts. The following keywords were extracted from the article about Tesla:

```
{
  “people” : [“Elon Musk”, “Musk”],
  “organizations” : [“Tesla”, “the Securities and Exchange Commission”,
“S.E.C”, “Twitter”],
  “products” : [“Model 3”, “Model S”, “Model X”, “the Model 3”]
}
```

These keywords were then combined and saved into an array in the NoSQL database.

4.3 Text summarization

There are methods to summarize text such as abstractive summarization, extractive summarization, and many others.

Abstractive methods select words based on sentiment understanding, even when those words did not appear in the source documents. At the time of writing this paper, there are no simple ways to perform abstractive summarization, therefore extractive methods will be used.

Extractive methods attempt to summarize articles by selecting a subset of words that retain the most important points. This approach weighs the important part of sentences and uses the same to form the summary. Different algorithm and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other [8]. The process of extractive summarization is similar to keyword extraction, using word frequencies to score sentences and their importance. The full flow of text summarization can be found on Figure 7.

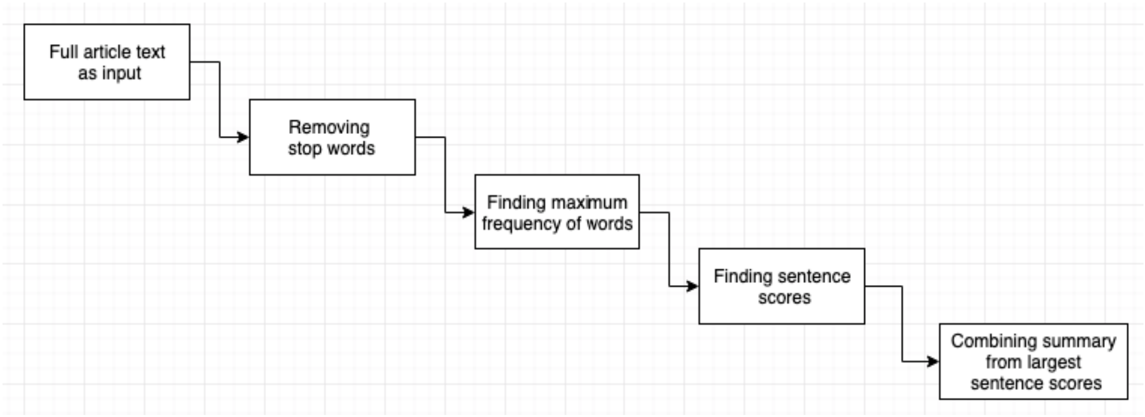

Full article text as input
Removing stop words
Finding maximum frequency of words
Finding sentence scores
Combining summary from largest sentence scores

Figure 7 Flow of summarizing texts

Removing stop words is done by using an English stop word corpus provided in the NLTK library. This process leaves the most important words in the context of the article. After this step, the maximum frequency of each word is found using the following formula.

$$frequency\ of\ word\ X = \frac{\sum word\ X\ in\ text}{maximum\ value\ of\ \sum word\ in\ text}$$

Once the word frequencies are found, the sentence scores can be calculated using the following formula.

$$sentence\ score = \sum frequency\ of\ words\ in\ sentence$$

After this is found, the largest sentence scores are taken, and a summary is combined from the sentences with the respective scores.

One of the questions that this paper investigates is whether summarizing the article text will provide a drastically different result for sentiment analysis and keyword extraction. When looking at the summarization of the example article about Tesla, it can be seen that overall the main message is retained, however some key sentences are ignored when simply looking at word frequencies. The summarized text makes no mention of the S.E.C investigation, which might play a major role in the change of stock prices. The full article text and its summary can be found in Appendix 1.

Table 5 Text summarization results

Text type	Number of words	Number of sentences	Number of relevant keywords
Original article text	343	49	11
Article summary	86	11	6

Table 5 shows that even though the article content will be just about 4/5th smaller sentence wise, the number of relevant keywords also falls drastically. This is fine when presenting summaries to the customer, however when looking for articles about a company, the number of keywords retained should be maximal so the search can provide a variety of different articles related to the change in stock price.

4.4 Analysis of sentiments

Sentiment analysis is investigated via two main methods, the Naïve Bayes algorithm and SVM, the steps of the latter being described on Figure 8.

The primary objective of this paper has been to find a method to perform sentiment analysis on news articles, using the outcome of this analysis to support the hypothesis of discovering news articles with the relevant sentiment when stock prices change. During the thesis, the author has chosen to train the Naïve Bayes and SVM models using a dataset of articles whose sentiments were labelled by the author and compare it to a dataset of pre-labelled movie review sentiments.

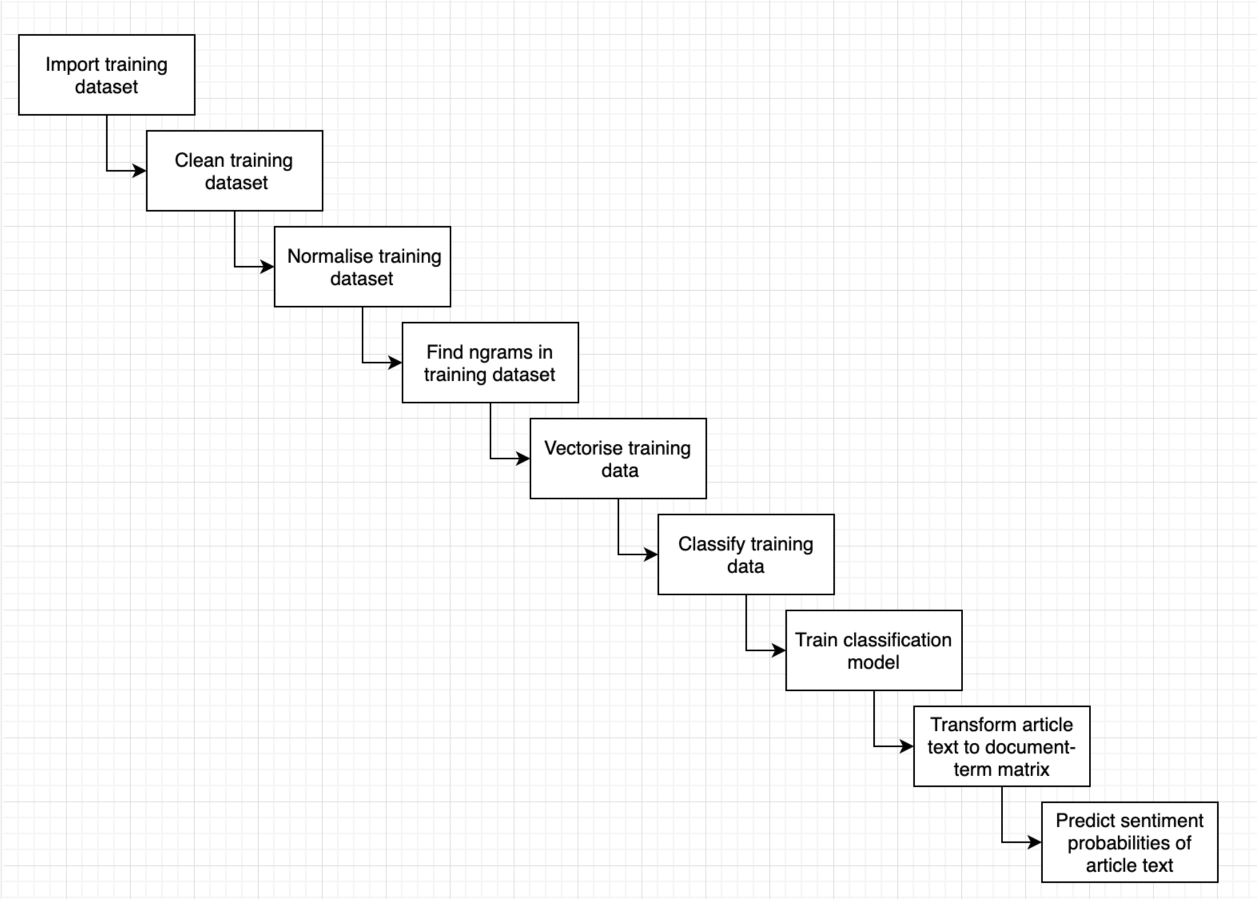

Import training dataset
Clean training dataset
Normalise training dataset
Find ngrams in training dataset
Vectorise training data
Classify training data
Train classification model
Transform article text to document-term matrix
Predict sentiment probabilities of article text

Figure 8 Sentiment analysis flow

The first step was to find a dataset that could be used to sufficiently evaluate the sentiments of various news articles from different publishers. As no pre-labelled dataset containing news article sentiments was found, the author labelled 200 articles, which isn't a perfect amount, however it does give an initial overview of how well a model could perform. The movie review dataset used for comparison was a set of 5000 files, where each file contained a review for a movie. There was also a file with the labelled sentiments, being either positive or negative. Once the data is imported, it has to be cleaned. The raw text data is put into a Pandas framework data frame that provides some structure to the initial data, which in turn makes data analysis and manipulation simpler.

Using NLTK and a wordnet lemmatizer the data is split into word-based tokens, removing all symbols and numeric values that might interfere with the semantic analysis model. Similar tools are used for keyword extraction. During the normalization process stop words are removed from the tokenized words list and lemmas of the tokens are appended to the data frame for further analysis. In addition to lemmas the ngrams for each lemmatized sentence are found. This is due to the fact that single words in most cases can be deceiving without context, however when combined into bigrams and trigrams, the context becomes clearer and the sentiment analysis process takes negation and subjects into consideration when estimating the sentiments.

When building the model, the count vectorizer method is used as it provides a simple way to build a vocabulary of known words, encode the documents into a vector form, and provide the count of each word in a document with the *fit_transform* method. These vectors can then be used for the machine learning model. The vectors are an input to the support vector classifier along with the sentiment labels. Before the sentiment labels can be input to the classifier model, they have to be mapped to integer values due to the fact that the labels are "positive" and "negative" not integer values 0 and 1.

Before training the models, the author took 200 of the scraped article summaries and labelled them. Unfortunately, the split of articles that were directly relevant to the stocks traded on the NASDAQ stock exchange was relatively small in contrast to articles about politics, sports and various other categories, therefore the training set was left to 50 articles, 25 with a positive sentiment and 25 with a negative sentiment.

The sentiment classification model is trained using 80% of the data as training data and 20% of the data as test data. The classification was run on over the manually labelled data of 50 hand-labelled articles and 4000 movie reviews in three different train-test scenarios. The first was with a train-test split of 60% training data and 40% test data, the second was with 70% training data 30% test data, and the third scenario, which also provided the most accurate results was with 80% training data and 20% test data.

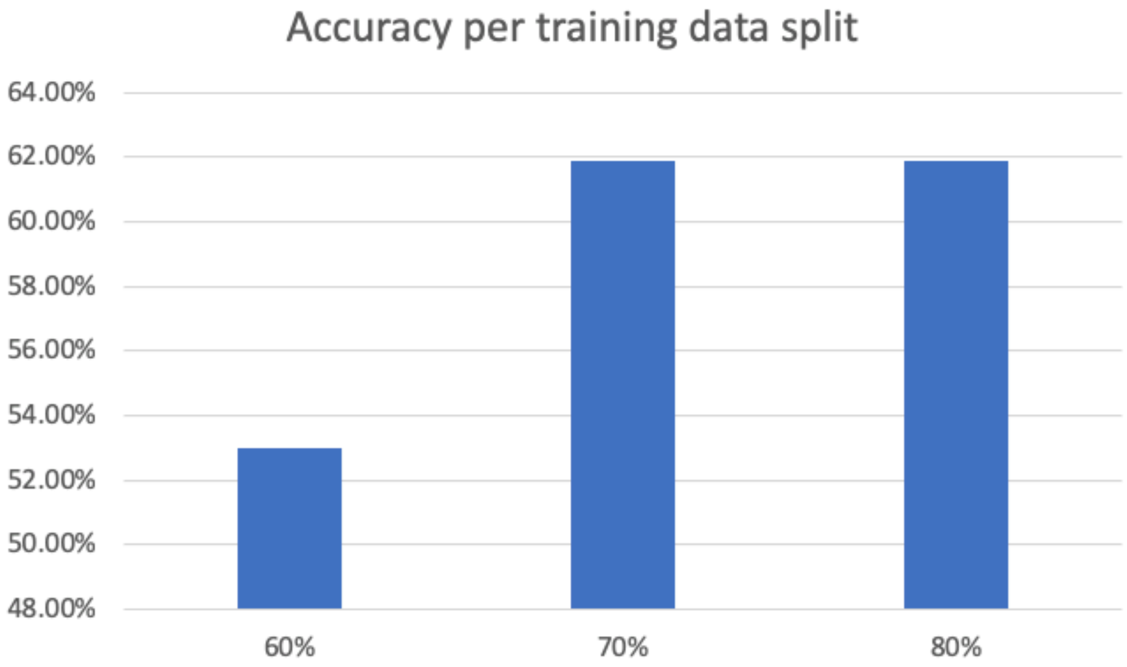


Figure 9 Average accuracy of manually labelled training data percentage using SVM classifiers

From the aforementioned splits, Figure 9 shows that the mean accuracy on the given test data and labels was 53%, 61.9% and 61.9% respectively. The accuracy could be improved significantly if the number of labelled articles was larger, unfortunately the sample of relevant articles scraped was too low.

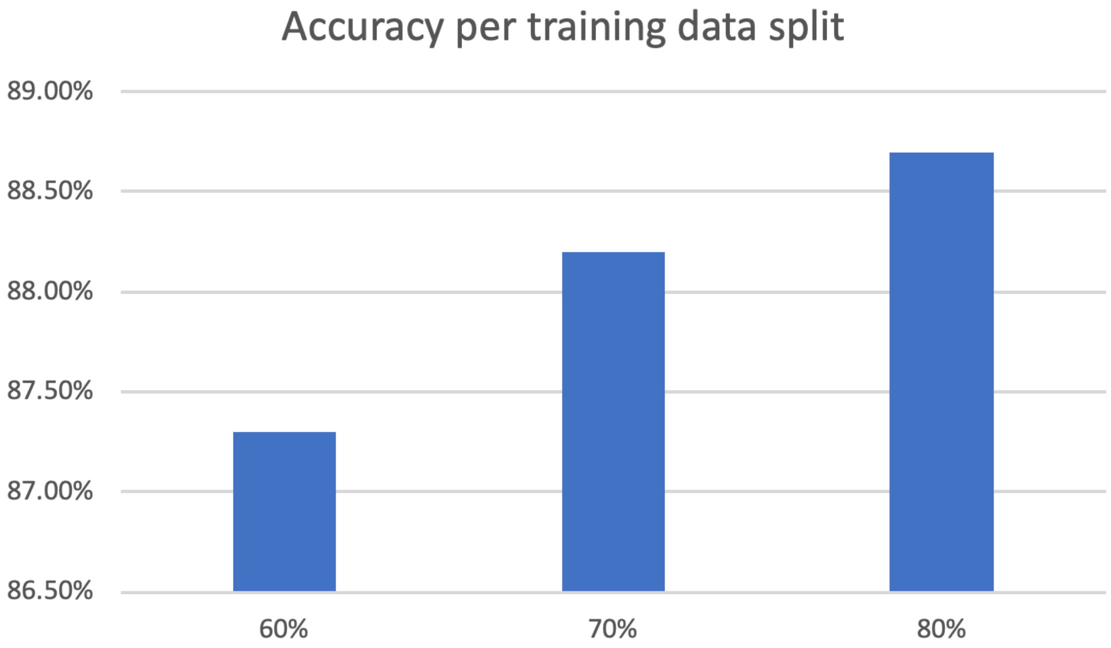


Figure 10 Average accuracy of movie review training data percentage using SVM classifiers

Comparing these results to the accuracy of the movie review dataset classifications on Figure 10, being 87.3%, 88.2% and 88.7% respectively, it can be seen that if the dataset was to be increased, the results would be more accurate. Generally, it could be seen that when a larger test data split was used, the sentiment scores of articles were less error prone. Interestingly when performing the classification using the Naïve Bayes algorithm the accuracy was best using a 70% train-test split, as seen on Figure 11.

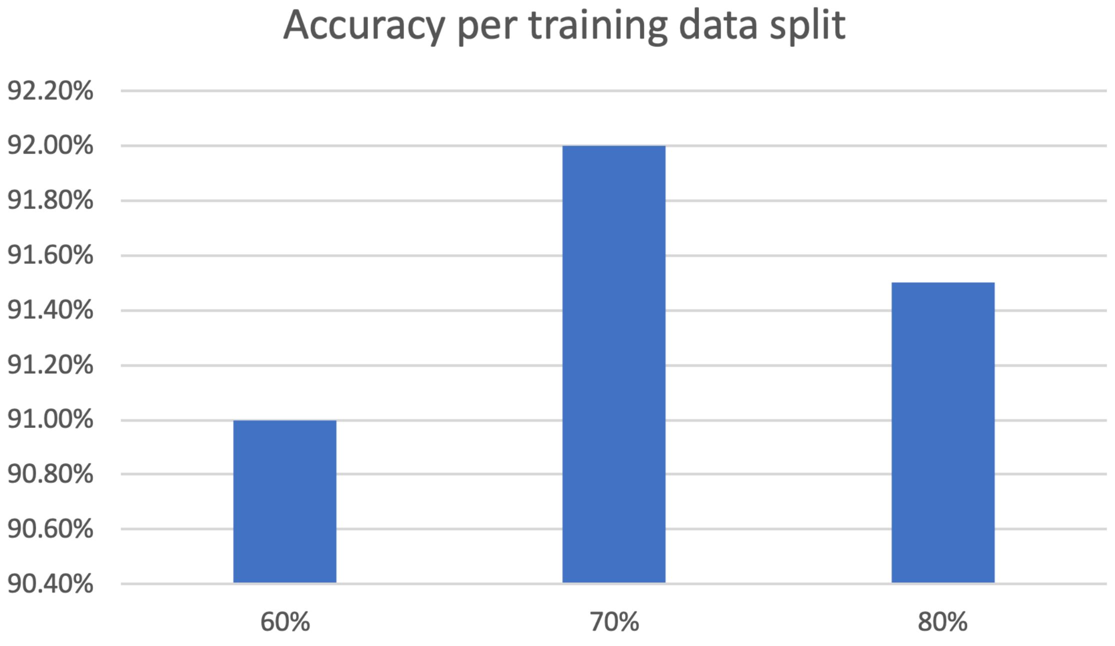


Figure 11 Average accuracy of movie reviews training data percentage using Naïve Bayes classifiers

When comparing this result to the Naïve Bayes classifications of the manually labelled article data on Figure 12, it can be seen that at such a small sample of labelled data, the accuracy is significantly lower, and the best result was obtained when the 80% of the data was used for training the model.

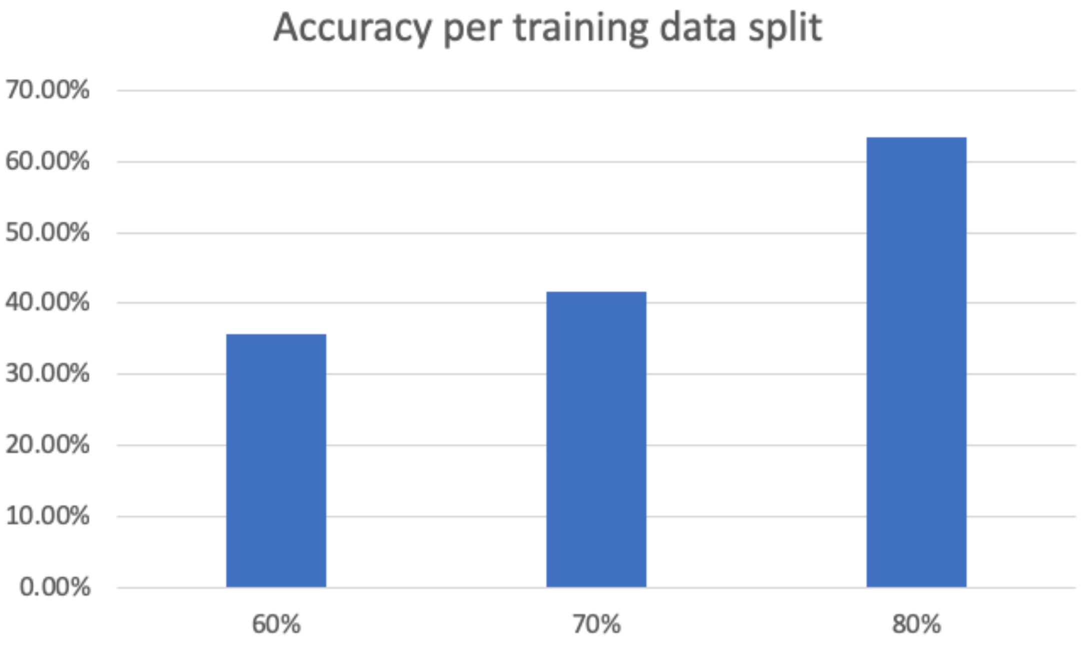


Figure 12 Average accuracy of manually labelled training data percentage using Naïve Bayes classifiers

As was stated in the previous chapter, one of the questions that this paper investigates is whether summarizing texts has an effect of the sentiment scores of articles. The author investigated the analysis results of several thousand articles using SVM and Naïve Bayes for sentiment analysis, in Table 6, a sample of 7 articles are presented with results from the SVM classification.

Table 6 Comparison of full text and summary sentiment scores for SVM classifiers

Full text positive sentiment score	**Full text negative sentiment score**	**Summary positive sentiment score**	**Summary negative sentiment score**
0.9968	0.0032	0.9271	0.0729
0.8311	0.1689	0.783	0.217
0.9428	0.0572	0.5	0.5
0.4532	0.5468	0.3332	0.6668
0.5	0.5	0.5	0.5
0.3404	0.6596	0.5	0.5

Full text positive sentiment score	Full text negative sentiment score	Summary positive sentiment score	Summary negative sentiment score
0.7661	0.2339	0.4666	0.5334

Table 6 shows that when sentence summaries are created, the sentiment score can change in a variety of different ways. From a sample of 1360 articles, it was found that 780 articles were identified as to having a positive sentiment. When looking at summaries of articles, it was found that only 770 percent of summaries had a positive sentiment. Of those 1360 articles 576 article full texts were found to have a negative sentiment, with 574 article summaries being found to have a negative sentiment.

When investigating the results of the Naïve Bayes classification on the full article text and the summary for a random set of 7 articles, Table 7 shows a little bit more consistency to the scores of sentiments in full texts compared to the scores of sentiments in summaries.

Table 7 Comparison of full text and summary sentiment scores for NB classifiers

Full text positive sentiment score	Full text negative sentiment score	Summary positive sentiment score	Summary negative sentiment score
0.454	0.546	0.399	0.601
0.548	0.452	0.747	0.253
0.391	0.609	0.272	0.728
0.722	0.279	0.514	0.486
0.508	0.492	0.593	0.407
0.489	0.521	0.619	0.381
0.424	0.576	0.576	0.424

At a first glance, the precision percentages look promising, however on further inspection, looking at all the articles with further detail and looking specifically at true-positives and false-positives, it can be seen that there were 338 articles that had a positive full text and summary sentiment score, 589 articles that had a negative full text and summary sentiment score, 185 articles that had a positive full text sentiment score, but a negative summary sentiment score, 178 articles that had a negative full text sentiment score, but a negative summary sentiment score, 16 articles that were falsely summarized as having a neutral sentiment. Considering that almost 40% of the summaries were classified to have a wrong sentiment in the context of the article, it can be concluded that drawing trustworthy sentiment analysis from summaries is possible, however quite error prone with the current solution.

Table 8 Comparison of sentiment analysis scores between SVM and NB classifiers

Article ID	**SVM positive sentiment score**	**NB positive sentiment score**	**SVM negative sentiment score**	**NB negative sentiment score**
5cccadbe2f90c0254c174283	0.998	0.601	0.002	0.399
5ccd8a4b2f90c02a4952a761	0.943	0.501	0.058	0.499
5ccdae4b2f90c02bcf0a1054	0.729	0.631	0.271	0.369
5cccc10c2f90c025a9f51a37	0.015	0.32	0.985	0.68
5ccec2052f90c007a28ed216	0.528	0.486	0.472	0.514
5ccfe2442f90c01583df3054	0.655	0.804	0.345	0.196
5ccca9742f90c0254c174217	0.471	0.49	0.529	0.51
5ccf2a8a2f90c010b106c4c2	0.985	0.67	0.015	0.33
5ccd8a212f90c02a4952a75d	0.3748	0.5867	0.6252	0.4133
5ccdadbf2f90c02bcf0a1046	0.5	0.391	0.5	0.609

Comparing the results of the SVM classification to the Naïve Bayes classification the general trend is that both of the algorithms classify the texts to having a similar sentiment. In Table 8, looking at 10 random articles, it can be seen that the percentages for positive and negative sentiments for both algorithms vary somewhat from each other, however the sentiment is the same. In this table if the positive sentiment score is over 0.5 then the article is considered to have a positive sentiment, if the negative sentiment score is over 0.5 then the article is considered to have a negative sentiment and if the scores are exactly 0.5 then the article is considered to have a neutral sentiment.

Table 8 also shows that even though there are a few values that have been classified differently by the two algorithms, there are a lot more classifications that have been classified to have the same sentiment. However, on further analysis it was found that from 4846 articles there were 2358 articles that had the same sentiment score in both SVM and NB classifications, 548 articles where the SVM classified the sentiment to be positive, but NB classified the article sentiment as negative, 1661 articles where the SVM classified the sentiment as negative whilst NB classified the sentiment as positive, and 279 articles where the SVM classified the article to have a neutral sentiment and NB classified the article to be either positive or negative.

4.5 User interface

To allow the user to find the articles related to a stock price a user interface was created. The user interface consists of a Java backend application that serves two purposes – one is to query for articles based on the keywords that the customer enters and the second is to pass the data to the frontend application. The backend application also determines which articles to present, based on the stock price trend in the date period the customer selected. If the trend shows that stock prices are on a decline, the articles with a negative sentiment score will be displayed, if the stock price trend is ascending, articles with a positive sentiment score will be displayed. In addition to this, the backend application also queries for data and executes the python-based sentiment analysis scripts when no data has been found in the local NoSQL database.

5 Experimental setup

To validate whether the hypothesis of news articles appearing when stock prices change is true, a simple observative experiment can be performed. As the time to gather articles was fairly limited, at first a list of articles was investigated. From those articles a few companies and products were extracted, and visual confirmation was done. One of the products investigated was Bitcoin. When looking at the price of Bitcoin, specific changes in stock price were noticed. For this experiment, stock price in the range of April 22nd to April 30th were investigated.

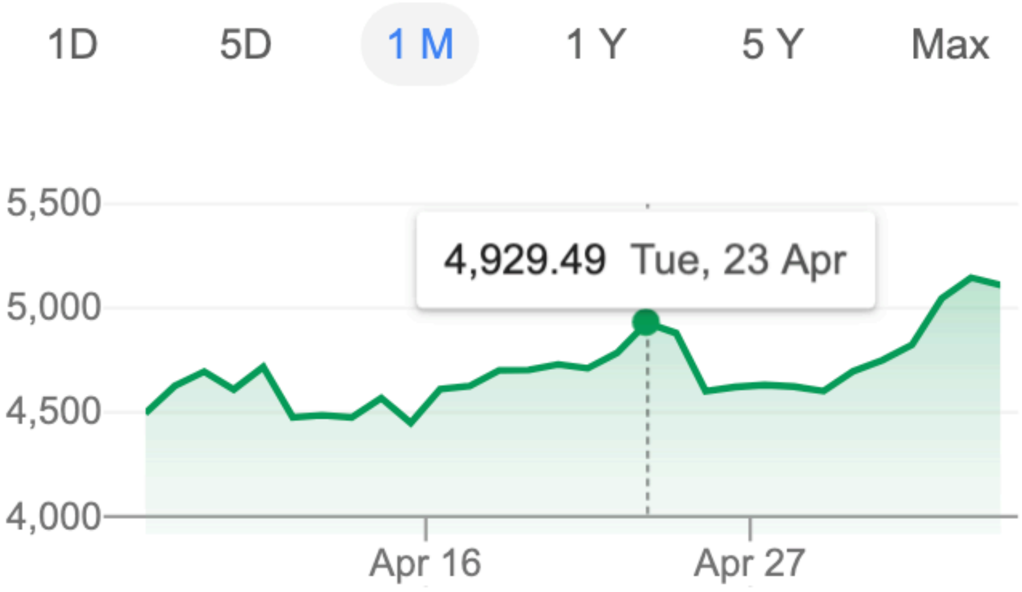


Figure 13 Price of Bitcoin in April 2019

From Figure 13 it can be seen that the price of Bitcoin had a fairly steep decline around April 23rd that stabilizes around April 25th. Within this period of time, it is expected that articles about Bitcoin exist with a negative sentiment.

Table 9 News articles about Bitcoin - sentiment data

Article Date	Negative Sentiment Score	Positive sentiment score
2019-04-23	0.99	0.01
2019-04-23	0.987	0.013
2019-04-23	0.646	0.354

Article Date	**Negative Sentiment Score**	**Positive sentiment score**
2019-04-24	0.524	0.476
2019-04-24	0.994	0.006
2019-04-26	0.5	0.5

When investigating the data in Table 9, it can be seen that two articles with a very high negative sentiment and one article with a mildly negative sentiment appear on April 23rd. On April 24th another article appears with a highly negative sentiment, and one appears with a semi-neutral sentiment. This is around the time that the stock price stabilises, as can be seen on Figure 13, between the 25th of April and the 28th of April the change in stock price is rather flat, supported by the article on the 26th of April, with a completely neutral sentiment score. When looking at the summaries as to understand why the drop happened there are a lot of different keywords in the articles. For the specific Bitcoin articles some of the main points have been its use in criminal activities. Although this specific case may seem like cherry-picking, there are other companies in the data that follow the trend.

Considering that the sentiment analysis model is a major factor to whether the customer is shown correct results, it is important to know how accurately the model, based on the movie reviews training data, evaluates the sentiment in the news articles. To measure this there are several calculations that are done.

To measure the accuracy of the models the F-score, precision, and recall were found using the equations below:

$$F_1 = \left(\frac{recall^{-1} + \ precision^{-1}}{2}\right)^{-1} = 2 * \frac{precision * recall}{precision + recall}$$

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

where *tp, fp, fn* are the number of True Positives, False Positives and False Negatives respectively [13].

When comparing the accuracy and precision between the two algorithms investigated the results surprisingly show that the Naïve Bayes algorithm has a better F_1 score as seen on Table 10.

Table 10 Precision, recall and F_1 scores of the Naive Bayes algorithm

	Precision	**Recall**	**F_1 score**	**Support**
1	0.9064	0.9246	0.9154	199
weighted avg	0.9152	0.9150	0.9150	400

Comparing this to the result of the precision, recall and F_1 score of the SVM presented in Table 11, it can be seen that the Naïve Bayes algorithm performs better with the movie review dataset, however it does take a significantly longer amount of time to train the model. A similar result was found when comparing the results of the manually labelled data. In manually labelled data the NB F_1 score was 0.633 compared to the SVM F_1 score of 0.411. The same split of training data was used for both models.

Table 11 Precision, recall and F_1 scores of the support vector classifier

	Precision	**Recall**	**F_1 score**	**Support**
1	0.9028	0.8711	0.8867	512
weighted avg	0.8866	0.8860	0.8860	1000

The author also investigated how well the classifier algorithms classify summary sentiments in comparison to full article text sentiments using the manually labelled data. The results were that when comparing the predicted full text sentiment to the predicted summary sentiment, the SVM classifier predicted the correct sentiment 61% of the time. The Naïve Bayes classifier however predicted the correct sentiment 79% of the time.

In addition to the previous methods a Stanford CoreNLP text analyzer was evaluated. StanfordNLP provides the base forms of words, their parts of speech, named entity recognition functionality, data normalization, understanding syntactic dependencies, analyzing sentiments, and information extraction among other functionality[24].

In the context of this paper, every article text gets parsed by Stanford CoreNLP's sentiment analysis module and the results are compared to the results of the sentiment analysis implemented for this paper. There is one challenge however – the sentiment analysis model in Stanford CoreNLP is trained to evaluate the sentiment of each sentence in an article whilst the sentiment analysis model for this paper is trained to evaluate the sentiment of the entire text. Therefore, an algorithm had to be created to summarize the sentiment provided by Stanford CoreNLP.

Initially the arithmetic mean was found and in the case of articles with negative sentiment the results were very similar between the sentiments evaluated by both models, however in the case of articles with a positive sentiment, the Stanford CoreNLP model classified a suspiciously small amount of articles to having a positive sentiment, between 1% - 2% of all the articles. One way to improve the result of the Stanford CoreNLP sentiment score for the whole text would be to rank the article to sentences that provide meaning the context of the article and sentences that provide less meaning would have less weight on the sentiment.

However, even with the weighed sentiment scores the results varied too much and the investigation showed that this method of validation is unreliable. In the end, the validation method that was tested was cherry-picking companies and seeing whether associations of stock change exist, and a comparative method between the main analysis results. As analysis was done based on two different datasets, each with two different algorithms, looking at the summaries that receive the same sentiment score gives the author validation that the concept works to some extent. It's not perfect, but it's a start.

6 Summary

In current times where the amount of information a person receives every day is growing exponentially, it’s quite difficult to read every single news piece and to stay in the loop of what’s happening in the ecosystems of companies. This can have a severe impact especially in areas where finances are involved. As people tend to favor finding their own way towards wealth, every year new investors enter the stock market without having a clue as to why stock prices change.

There already exists plenty of methods to predict how stock prices are going to change, along with methods to analyze texts and find their sentiments, however there aren’t very many methods that perform these analyses on summarizations of texts. Linking together several NLP techniques such as keyword extraction, text summarization, and sentiment analysis and applying these techniques to a practical use case in the financial sector are hard to find. Doing this analysis in real time allows the users of such news analysis systems to make better decisions about their investments.

The main goal of this thesis was to evaluate which of the chosen sentiment classification algorithms would perform better on previously summarized text articles. Alongside this goal, the question that had to be answered was whether random sentiment datasets can be used to train the models and in the case of the movie review dataset, it seems like the results are varied – there are a noticeable number of articles where the sentiment classification is questionable, however as the experiment showed, the data does match the hypothesis in many scenarios. The results of the comparison between the Naïve Bayes classification and SVM classification accuracy was also surprising. According to previous research by Kalyani et al [20] the hypothesis was that SVM classifiers would be more accurate, however the opposite was discovered, that NB classifiers performed more accurately, however training the model took up to 5 times longer with the same size of training data.

To summarize, during this thesis the author wrote a news article crawler that collects news articles from the internet using an API to gather a list of articles, and a python script that iterates over the articles and parses the content of the article from the full web page. The author then implemented an automated keyword extraction algorithm, expanding it with NER functionality. Furthermore, the author labelled a small dataset of news article summaries and investigated, which of the two chosen classification algorithms analyzes sentiments better with small summarized texts, finding that the Naïve Bayes classification algorithm is better suited for analysis of such texts. The author also programmed an API and a user interface to display the collected data in a sensible fashion.

The author hypothesized that when creating a summary of an article text, the sentiment might change due to the loss of information. This did happen, however in the case of the Naïve Bayes classifier, it only happened 21% of the time which is an acceptable result. The author feels that if the manually labelled dataset had been larger, the accuracy would have been better also, however due to time constraints this hypothesis wasn't investigated further.

The key goals of this thesis were met, the results somewhat surprising, but mostly expected. A certain method was proposed to parse news articles in real time, extract keywords from the articles using NLP methods, summarize the text using NLP methods, and perform sentiment analysis on full article texts along with the summaries of the full texts using Naïve Bayes classifiers over their SVM counterparts. Finally, this data was presented to the customers in a user interface and summarizations were presented to explain the associations of a certain change in stock price.

Bibliography

[1] A. A. Kirilenko and W. A. Lo, "Moore's law ver-sus murphy's law: Algorithmic trading and itsdiscontents.," *Journal of Economic Perspec-tives,* pp. 51-72, 2013.

[2] A. Singh, "Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)," 25 October 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/.

[3] A. Conneau, H. Schwenk, L. Barrault and Y. Lecun, "Very Deep Convolutional Networks for Text Classification," arXiv:1606.01781, 2016.

[4] N. Tabari, P. Biswas, B. Praneeth, A. Seyeditabari, M. Hadzikadic and W. Zadrozny, "Causality Analysis of Twitter Sentiments and Stock Market Returns," in *Proceedings of the First Workshop on Economics and Natural Language Processing*, Melbourne, Australia, 2018.

[5] L. Xiaodong, X. Haoran, C. Li, W. Jianping and D. Xiaotie, "News impact on stock price return via sentiment analysis," *Knowledge-Based Systems,* pp. 14-23, 2014.

[6] News API, "NewsAPI," 29 04 2019. [Online]. Available: https://www.newsapi.org.

[7] M. V. Mäntylä, D. Graziotin and M. Kuutila, "The evolution of sentiment analysis: a review of research topics, venues, and top cited papers.," *arXiv preprint,* p. 1, 2016.

[8] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis," *Standford University, CS229,* vol. 15, 2012.

[9] T. H. Nguyen, K. Shirai and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications,* vol. 42, no. 24, pp. 9603-9611, 2015.

[10] J. Bollen, H. Mao and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science,* vol. 2, no. 1, pp. 1-8, 2011.

[11] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies,* vol. 5, no. 1, pp. 1-167, 2012.

[12] L. Harper, "Sentiment Analysis Analysis Part 2 —Support Vector Machines," 01 08 2017. [Online]. Available: https://medium.com/nlpython/sentiment-analysis-analysis-part-2-support-vector-machines-31f78baeee09. [Accessed 24 03 2019].

[13] Y. Xu, X. Whi and Q. Wang, *Sentiment Analysis of Yelp's Ratings Based on Text Reviews,* Stanford University, 2014.

[14] B. e. a. Agarwal, "One-class support vector machine for sentiment analysis of movie review documents," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering,* vol. 9, no. 12, pp. 2458-2461, 2015.

[15] K. P. e. a. Murphy, *Naive bayes classifiers,* vol. 18, University of British Columbia, 2006, p. 60.

[16] I. Rish, "An Empirical Study of the Naïve Bayes Classifier," *IJCAI 2001 Work Empir Methods Artif Intell,* vol. 3, 2001.

[17] C. Fox, "A Stop List for General Text," *SIGIR Forum,* vol. 24, no. Fall 89/Winter 90, pp. 19-21, 1989.

[18] J. e. a. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003.

[19] B. Pang, L. Lee and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 2002.

[20] A. Ben-Hur, D. Horn, H. Siegelmann and V. N. Vapnik, "Support vector clustering," *Journal of Machine Learning Research,* vol. 2, pp. 125-137, 2001.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[22] M. Gamon, "Sentiment Classification on Customer Feedback Data: Noisy Data, Large Feature Vectors, and the Role of Linguistic Analysis," in *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, 2004.

[23] P. M. Domingos, "A few useful things to know about machine learning," *Commun. acm,* vol. 55, no. 10, pp. 78-87, 2012.

[24] J. Kalyani, H. N. Bharathi and R. Joythi, "Stock Trend Prediction Using News Sentiment Analysis," *International Journal of Computer Science and Information Technology,* pp. 67-76, 2016.

[25] S. Vivek, "Automated Keyword Extraction from Articles using NLP," 17 December 2018. [Online]. Available: https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34.

[26] P. Dubey, "Understand Text Summarization and create your own summarizer in python," 23 December 2018. [Online]. Available: https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70.

[27] J. Bartczuk, "Text Summarization In Python," 16 January 2019. [Online]. Available: https://semantive.com/text-summarization-in-python/.

[28] S. CoreNLP, "Stanford CoreNLP – Natural language software," Stanford University, 05 October 2018. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/. [Accessed 29 March 2019].

[29] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Appendix 1 – Article text of Tesla article

Full text:

Tesla delivered fewer vehicles than expected in the first quarter, the company said Wednesday, with deliveries falling 31 percent to 63,000 vehicles from the fourth quarter of 2018. The electric-car maker, led by Elon Musk, said the lower-than-expected delivery volume and several pricing adjustments would hurt its first-quarter net income. The company previously said it would lose money in the first quarter. The company's shares were down about 8 percent in afternoon trading on Thursday. The price cuts were aimed at making Tesla cars affordable to a wider range of customers. The cost to consumers was essentially increased on Jan. 1 when a federal tax credit available to buyers of electric vehicles was cut in half, to $3,750, as part of a scheduled phaseout of the incentive. The credit will drop to $1,875 after July 1 and end next Jan. 1. The drop in deliveries appeared to reflect new weakness in demand in the United States, occurring even though Tesla began shipping cars in volume to Europe and China in the quarter. Deliveries of its Model 3 sedan, its most affordable car, totaled 50,900, 20 percent fewer than in the fourth quarter. Deliveries of its Model S and Model X luxury models fell 56 percent to 12,100 cars. [Hints of the slump were evident in registration data.] Tesla produced 77,100 vehicles in the quarter, down from 86,555 in the fourth quarter. It reiterated that it expected to deliver 360,000 to 400,000 cars this year, and said it had finished the quarter with “sufficient” cash on hand, without offering further details. Analysts have raised concerns about Tesla’s cash supply. It started the first quarter with $3.7 billion, but used nearly $1 billion to make a payment to bondholders last month. Next month, Tesla hopes to begin volume deliveries of a $35,000 version of the Model 3, the car that many of the company’s fans have been waiting for. Analysts are uncertain whether Tesla can make money selling the Model 3 at that price, however. In a Feb. 28 conference call, Mr. Musk said Tesla would close most of its stores around the country and rely on online sales as part of an effort to slash costs so it could sell the $35,000 car profitably. But 11 days later, Tesla reversed course and said it would keep many of the stores, and reopen some that had already closed. It also said it would increase prices on all its cars except the $35,000 version of the Model 3. In the coming weeks, Tesla is expected to report a loss for the first quarter. The company was profitable in the final two quarters of 2018 as Model 3 sales rose. There have been other challenges for the company, including a renewed legal battle between Mr. Musk and the Securities and Exchange Commission. The S.E.C. asked a federal court in February to hold Mr. Musk in contempt for violating a settlement that he and Tesla reached with the commission last year regarding scrutiny of his statements. In a motion to the court, the S.E.C. said Mr. Musk had not sought approval from company lawyers, as required by the agreement, before making a sales prediction on Twitter that varied from previous announcements. The court is expected to rule on Thursday. In the tweet, which was posted Feb. 19, Mr. Musk said Tesla would make 500,000 cars this year, 100,000 more than the company had forecast in January. Tesla’s general counsel left the company two days later. In response, Mr. Musk’s lawyers said the commission was overreaching based on comparable precedents. They contend that Mr. Musk was acting within his discretion in his Twitter posting, and that it contained no material information beyond what the company had previously stated in public.

Summary:

The company previously said it would lose money in the first quarter. It reiterated that it expected to deliver 360,000 to 400,000 cars this year, and said it had finished the quarter with “sufficient” cash on hand, without offering further details. In the tweet, which was posted Feb. 19, Mr. Musk said Tesla would make 500,000 cars this year, 100,000 more than the company had forecast in January. said Mr. Musk had not sought approval from company lawyers, as required by the agreement, before making a sales prediction on Twitter that varied from previous announcements. It started the first quarter with $3.7 billion, but used nearly $1 billion to make a payment to bondholders last month. Deliveries of its Model 3 sedan, its most affordable car, totaled 50,900, 20 percent fewer than in the fourth quarter. Next month, Tesla hopes to begin volume deliveries of a $35,000 version of the Model 3, the car that many of the company’s fans have been waiting for.

Appendix 2 – Code of the product

The final product can be found on TalTech Gitlab.

There are 2 projects – one for the Python code, and one for Java and Angular code.

Python code performs sentiment analysis, keyword extraction and article parsing.

https://gitlab.cs.ttu.ee/Marko.Kaar/masterthesis

Java and Angular code deliver a visual user interface along with querying for the data from the local MongoDB database.

https://gitlab.cs.ttu.ee/Marko.Kaar/masterthesis-java

The entire codebase along with the training data can also be found in a zipped format.

https://drive.google.com/open?id=1AUsOnctaeuHh7ol0Ipw_-_kVU3NjST78

Appendix 3 – Example of analysed data

```
Falsely Classified:
{
    "_id" : ObjectId("5ccd7c722f90c02a4952a615"),
    "url" : "https://www.bbc.co.uk/news/technology-48036912",
    "date" : "2019-04-24T09:51:02Z",
    "keywords" : [ "paint", "digital", "program", "window", "artwork",
"tool", "included", "computer", "would", "urge", "Miranda Lorikeet", "Ms
Lorikeet", Microsoft Windows", "Pat Hines", "Matt Hancock", "Microsoft
Paint", "Microsoft", "Paint", "BBC News", "MS Paint", "the Conservative
party", "Windows", "Windows 10", "Captain Redblood"
    ],
    "fulltext" : "Fans of low-fidelity art app Microsoft Paint are rejoicing
after it was confirmed it would remain a part of the Windows operation system
\"for now\". In 2017, Microsoft had said that Paint would be deprecated but
it survived. Confusion returned in recent weeks as users questioned whether
Paint would be part of the next Window 10 update, which launches in May. A
…
 \"The limitations are kind of what dictates my aesthetic with it, it's why I
chose it as my primary medium. \"If I had my way with it it would just be
included on every computer.\" Yet another aficionado is Concha Garcia Zaera,
an 88-year-old Spanish woman who regularly updates her Instagram account with
charming digital artworks, all produced in Paint. Ministers urge the
Conservative party to unite - while Matt Hancock urges Tory MPs \"to
compromise\". ",
    "shorttext" : "Fans of low-fidelity art app Microsoft Paint are rejoicing
after it was confirmed it would remain a part of the Windows operation system
\"for now\". \"It's a crappy tool at the end of the day, it's not good, and I
think I like using something that is genuinely a rubbish tool to make
artwork.\" Confusion returned in recent weeks as users questioned whether
Paint would be part of the next Window 10 update, which launches in May. A
Microsoft developer confirmed that Paint would be included - to the relief of
many. \"If I had my way with it it would just be included on every
computer.\" This would be unfortunate, added Ms Lorikeet, who praised the
\"accessible\" nature of the program. Yet another aficionado is Concha Garcia
Zaera, an 88-year-old Spanish woman who regularly updates her Instagram
account with charming digital artworks, all produced in Paint.",
    "negative_sentiment_score" : 0.7827,
    "positive_sentiment_score" : 0.2173,
    "summary_negative_sentiment_score" : 0.7665,
    "summary_positive_sentiment_score" : 0.2335
}
```

Correctly classified:

```
{
    "_id" : ObjectId("5ccd65f22f90c028f65e54f9"),
    "url" : "http://techcrunch.com/2019/04/23/waymo-picks-detroit-factory-to-
build-self-driving-cars/",
    "date" : "2019-04-23T17:03:56Z",
    "keywords" : ["vehicle", "driving", "service", "self", "driver", "build",
"begin", "car", "level", "american", "Waymo", "Gretchen Whitmer", "American
Axle & Manufacturing’s", "American Axle & Manufacturing", "SAE", "Waymo",
"the Michigan Economic Development Corporation", "MEDC", "Magna", "Jaguar",
"Chrysler", "Pacifica Hybrid"
    ],
    "fulltext" : "Waymo,  the self-driving vehicle technology startup under
Alphabet, is setting up shop in a Detroit factory on American Axle &
Manufacturing’s campus. Waymo said Tuesday it will partner with American Axle
& Manufacturing to repurpose the existing facility, which was most recently
used as a sequencing center for a local parts supplier. The goal is to begin
moving into the facility by mid-2019 and begin preparing the site for
manufacturing Level 4 autonomous vehicles. Level 4 is a designation by SAE
that means the vehicle handles all of the driving under certain conditions.
“By choosing to establish its new facility in Detroit, Waymo is continuing
the city’s momentum and further cementing Michigan as a leader in mobility
and the epicenter of advanced automotive manufacturing,” Governor Gretchen
Whitmer said in a statement. In January, the Michigan Economic Development
Corporation voted to approve Waymo’s plan to set up a manufacturing
facility in the state to build its self-driving vehicles. The MEDC approved
an $8 million grant for the project. Waymo has partnered with Magna to build
thousands of self-driving cars at the factory, including autonomous versions
of the all-electric Jaguar I-PACE and Chrysler  Pacifica Hybrid minivan, in a
bid to deploy its ride-hailing service at scale. In December, Waymo launched
a limited commercial robotaxi service in the Phoenix area, dubbed Waymo One.
The Waymo One self-driving car service, and accompanying app, still has
Waymo-trained test drivers behind the wheel. The safety driver will
eventually be removed from the vehicle. The service has slowly opened up to
more people. ",
    "shorttext" : "The goal is to begin moving into the facility by mid-2019
and begin preparing the site for manufacturing Level 4 autonomous vehicles.
In January, the Michigan Economic Development Corporation voted to approve
Waymo’s plan to set up a manufacturing facility in the state to build its
self-driving vehicles. Level 4 is a designation by SAE that means the vehicle
handles all of the driving under certain conditions. Waymo,  the self-driving
vehicle technology startup under Alphabet, is setting up shop in a Detroit
factory on American Axle & Manufacturing’s campus. The Waymo One self-driving
car service, and accompanying app, still has Waymo-trained test drivers
behind the wheel. In December, Waymo launched a limited commercial robotaxi
service in the Phoenix area, dubbed Waymo One. The service has slowly opened
up to more people.",
    "negative_sentiment_score" : 0.4123,
    "positive_sentiment_score" : 0.5877,
    "summary_negative_sentiment_score" : 0.469,
    "summary_positive_sentiment_score" : 0.531
}
```