NUMERICAL AND TEXT DATA AUGMENTATION FOR FINANCIAL MARKET DATA

By

LIM SHIN CHYI

A project report submitted in partial fulfilment of the
requirements for the award of
Bachelor of Science (Hons.) Actuarial Science

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

APRIL 2019

DECLARATION OF ORIGINALITY

I hereby declare that this project report entitled "**NUMERICAL AND TEXT DATA AUGMENTATION FOR FINANCIAL MARKET DATA**" is my own work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : Lim Shin Chyi

ID No. : 1502795

Date : 5/4/2019

APPROVAL FOR SUBMISSION

I certify that this project report entitled "**NUMERICAL AND TEXT DATA AUGMENTATION FOR FINANCIAL MARKET DATA**" was prepared by **LIM SHIN CHYI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Hons.) Actuarial Science at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Dr Goh Yong Kheng

Date : _____

ACKNOWLEDGEMENTS

I would like to thank Universiti Tunku Abdul Rahman (UTAR) Lee Kong Chian Faculty of Engineering and Science (LKC FES) for giving me the opportunity to study and research on the topic "Numerical and text data augmentation for financial market data". I would also like to express my gratitude and sincere thanks to my supervisor, Dr Goh Yong Kheng for his supervision, and being always ready to help and support my work with patience. A special thanks to my family members for their constant support. Lastly, I would like to thank my friends for their help and encouragement.

LIM SHIN CHYI

NUMERICAL AND TEXT DATA AUGMENTATION FOR FINANCIAL MARKET DATA

LIM SHIN CHYI

ABSTRACT

Extensive studies and researches have been done on the prediction of financial market using text data such as news or numerical data such as stock prices and trading volume. However, lesser works can be found on doing prediction using the augmented text and numerical data. It is believed that better prediction can be obtained by taking into cosideration both text and numerical data. This project aims to identify the differences between the results obtained from text data model, numerical data model and augmented data model, and more importantly to verify the hypothesis that augmented data model will generate better result.

Web scraping and RSS can be used to collect online text data, particularly the news articles. On the other hand, numerical data, which are S&P 500 index prices and 30 stock prices, are obtained through Bloomberg. The text and numerical data is processed and used to train different type of models. Text data model, numerical data model and augmented data model are developed and analyzed. Classification algorithms such as logistic regression, support vector machine, naive bayes, neural network and decision tree are employed to build the models. The results obtained from this project indicated that model based on augmented data has the highest predicting power and accuracy as compared to the other two models. Hence, the hypothesis is verified and proved to be true. In future, more data over a longer period can be extracted and used to develop the models in order to obtain a more comprehensive and accurate results and predictions. Tuning of the algorithms employed to train the models can be further studied and investigated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Objectives

This project is designed to archive the following objectives:

i. To understand how to process textual and numerical data for augmentation purpose using Python.
ii. To identify how to augment numerical and textual data using Python.
iii. To build a model using only textual data, a model using only numerical data and a model using the augmentation of numerical and textual data to predict financial market sentiment.
iv. To evaluate the accuracy of the models in analysing and predicting financial market sentiment.

1.2 Problem Statement

Most financial sentiment analysis and models consider only the effect of numerical data such as stock price. However, textual information such as news is also a significant determinant of financial market sentiment. Therefore, there is a need to do some research about the financial analysis and model that involving both numerical and textual data. It is also important for us to determine the combination of data that can produce the best analysis and model. This is to ensure that a better prediction can be done by employing the most appropriate analysis and model. Clearly, this is of the utmost concern of financial institutions and investors.

1.3 Motivation

There are mainly two approaches to analysis and predict of financial market movement, which are fundamental approach and technical approach. Fundamental analysis relies on the financial data of the business such as dividends, ratios, management effectiveness and earnings to get some insight. The main source of financial data is the financial report while some may also refer to news and analysis report written by economics. Most of the financial data used by fundamental analyst is in text form. On the other hand, technical analysis relies on historical data such as past

price and trading volume, which are mostly in numeric form (Joshi, Bharathi and Jyothi, 2016). It employs various charts and indicators to identify trend in those historical data (Schumaker and Chen, 2009). Although both of the methods can be used of forecast market trend, it would be even better if we could apply both of them by incorporating both numerical and textual data.

In addition, financial market analysis is significant for investors to determine the performance of their securities, decide on how to invest their funds and plan their future investment. It can make prediction on some future values such as market index and stock price. Financial market is affected by the demand and supply, which are determined by investors and investors make decision based on various information. Examples of information affecting investors' decisions are news, journals, historical stock prices and social media posts. Among all the data that exist in this universe, only a small part of it is in numerical form while a lot more is in textual form. Besides, nuances and behavioral expression present only in textual data. Text has emotive contents, opinions and connections (Das, 2014). Hence, it plays an important role in data analysis specifically in financial market data analysis.

Financial market analysis is considered to be thorough and comprehensive if and only if it takes into accounts the effects of both numerical and textual data. Numerical data for financial market usually refers to the index prices while textual data refers to news and articles. Prediction of market movement is a key purpose of doing financial market analysis. This is extremely important for financial institutions such as investment firms and banks since an accurate and precise forecast will bring them profits and high earnings. Therefore, numerical and text data augmentation for financial market data is an important topic to research in order to develop a good model for analysis and prediction.

1.4 Project Scope

The textual data of this project is limited to be 300 articles from Wall Street Journal provided by Dr Goh Yong Kheng while the numerical data will be S&P 500 index prices. Both the data are taken from September 2017 to October 2017. This project is

only for the analysis and prediction of S&P 500 index prices and not for the other financial market instrutments. Although the data collection of articles is performed, it is not used to develop any model due to the limitation of time. It is only for the demonstration of data collection steps and methods.

Including this chapter, this report is divided into five parts. The second chapter discusses on the past studies and researches related to this project. In Chapter 3, the methodology used in this project is explained and elaborated. Chapter 4 presents and discusses the results obtained from the three models. Model using text data, model using numerical data and model using augmented text and numerical data are compared and analysed in this report. The last chapter draws some conclusive remarks and addresses the limitation faced in this project. It discusses the possible improvements and suggests new direction of future works.

CHAPTER 2: LITERATURE REVIEW

In this chapter, researches regarding the prediction of stock market are investigated in Section 2.1. Section 2.2 presents some works related to the development of models such as the data used and algorithms employed. Section 2.3 discusses past studies about the collection of data. Data collection is divided into three parts which are the collection of data using web scraping, collection of data using RSS and the use of cloud computing to ease data collection.

2.1 Prediction of Stock Market

Early studies on stock market prediction were based on two theories, which are random walk theory and efficient market hypothesis (EMH). The random walk theory was introduced by Fama (1965) and Malkiel (1985) whereas the EMH was introduced by Fama (1970). EMH assumes that stock market prices at any point in time "fully reflect" all information available and this indicates that financial market movement are driven by "new" information such as news and current events rather than present or past prices. Since news and current events are unpredictable, financial market prices will follow a random walk pattern and thus impossible to be predicted with more than 50% accuracy (Qian & Rasheed, 2007). In the research of Qian and Rasheed (2007), several theories opposing the EMH and random walk model had been presented and it is proven that prediction is possible through the achievement 65% prediction accuracy.

Besides, Fortuny et al (2013) argued that stock price movement could be predicted based on technical indicators, news data and their sentiments. Fortuny et al (2013) presented that models derived from these information tend to perform better than random guessing. Schumaker and Chen (2009), and Mittermayer (2006) also developed models on the basis of textual information to perform directional predictions of stock movement for instance up or down instead of the actual values.

Extensive studies on stock price movement prediction have been done based on either textual data such as news or numerical data such as stock prices. However,

lesser work can be found on using both textual and numerical data. According to Schumaker and Chen (2009) and Li et al (2011) as cited by Fortuny et al (2013), models that used only textual data are too limiting. Schumaker and Chen (2009) showed that better prediction was obtained using articles (news) and stock price at the time of article released as compared to using only article data. Therefore, prediction on index (S&P 500) price movement based on only text data, only numerical data, and the augmented text and numerical data, is performed in this project.

Although news most certainly influences investor sentiments and stock market prices, public mood states and sentiments also play an equally important role in influencing the stock market prices. There are numerous studies regarding stock price prediction based on public sentiment data extracted from online social media such as Twitter. Nowadays social media has become a prefect representation of public sentiments and feelings on an incident or event. Twitter is the social media platform used by a lot of researchers to study public sentiments. Bollen, Mao and Zeng (2011) introduced a model to predict stock market using public mood states from tweets posted in Twitter. Besides, Ranco et al (2015) and Pagolu et al (2016) showed the existence of strong correlation between the financial market and the Twitter sentiment. Ranco et al (2015) investigated the effects of Twitter sentiment on stock price returns. Pagolu et al (2016) also developed a sentiment analyzer that is used to determine the sentiment of a tweet for predicting the rises and falls in stock market. Prediction of financial market sentiment based on social media data is an interesting area of study. However, research regarding this topic is not done in this project due to the limitation of time.

2.2 Model Development

Variety of models used to predict financial market have been developed based on different types of data. Schumaker and Chen (2009) presented model using extracted article terms and stock price at the time of article released, model using only extracted article terms, and model using extracted terms and a regressed estimate of the +20 minute stock price. For textual representation, they showed that Proper Nouns (hybrid go-between for Noun Phrases and Named Entities) performed better as compared to

Bag of Words, Noun Phrases and Named Entities.

Schumaker and Chen (2009) used linear regession and support vector machine (SVM) to develop models in their research. Fortuny et al (2013) also employed Support Vector Machine (SVM) in building their model as it has been proven to be successful for text mining by Cohen & Hersh (2005) and Tang et al (2009). Moreover, Naïve Bayesian classifier is trained by Gidófalvi (2001) to predict the movement of the associated stock price.

On the other hand, Wong, Liu and Chiang (2015) proposed a unified latent factor model to characterize the joint correlation between stock prices and news articles for predictions on individual stocks. They used sparse matrix factorization to formulate model learning. Akita et al (2016) predicted stock prices by using Long Short-Term Memory (LSTM) to regress from textual and numerical data while taking into account the correlations between multiple companies in the same industry.

According to Sidana (2017), machine learning has seven type of classification algorithms which are linear classifiers (logistic regression and naive bayes classifier), support vector machines, decision trees, boosted trees, random forest, neural networks and nearest neighbour. Since model predicting financial market sentiment is a classification type model, all the algorithms trained in this project are classification type. Models using logistic regression, support vector machine, naive bayes, neural network and decision tree are developed in this project.

2.3 Data Collection

Due to the advancing of internet and evolution of World Wide Web (WWW), a lot of users from different backgrounds exchange, share and store infromation online as they can easily and fastly get connected to their target audience (Saurkar, Pathare and Gode, 2018). Since tons of data available online, there is a need for researchers to change their source of obtaining data. This section discusses about the past works regarding the collection of online data, particularly text data.

Web scraping and RSS are widely used methods to obtain online data. Besides, cloud computing is used to ease the process of web scraping and collection of data through RSS as it allows a scipt to run continuously.

2.3.1 Web scraping

Saurkar, Pathare and Gode (2018) claimed that web scrapping is the technique to handle and obtain useful information in least efforts from the infinite data available on the Internet. Hoekstra, Bosch & Harteveld (2012) proved the possibility of data collection through web scrapping. They stated that it could increase the quality, frequency and speed of data collection leading to improved learning.

There are a few application of web scrapping to facilitate data collection and analysis. It is used to collect prices from online retailer and construct daily price indexes to determine online inflation rate in five Latin American countries (Cavallo, 2013). At the European level, web scraping is employed to automatically collect consumer prices online (Polidoro et al, 2015). Hessisches Statistisches Landesamt (2018) stated that German has increasingly used web scraping as part of its pricing statistics. European Statistical Systems Network (ESSnet) employed web scraping on job vacancies and enterprise characteristic. Web scraping job vacancies involved the automated extraction of information from job portals and company websites while web scraping enterprise characteristic involves automated search, store, structuring and linking of company websites with official statistics' database (Hessisches Statistisches Landesamt, 2018).

However, some problems do arise from the use of web scraping. Krotov and Silva (2018) stated that the issue regarding the legality and ethics of web scraping remains to be a "grey area" with no definite answer. Therefore, it is necessary for web scraping users to comply with some legal and ethical requirement (Krotov and Silva, 2018). Mattosinho (2010) explained that requesting of data automatically at high speed through web scraping might cause Denial-of-Service attack to the requested server. This is because substantial amount of requests triggered and sent to the server in a short period of time. This has been taken in consideration when web scraping is employed to extract data in this project.

2.3.2 Really Simple Syndication (RSS)

RSS which stands for Really Simple Syndication or Rich Site Summary can also be used to obtain online data. It is widely used by news sites to publish articles' information and publishers to collect data automatically (Hurtado, 2015). RSS allows users to syndicate and aggregate online content, particularly the frequently updated content such as news, blog entries and HTML (O'Shea and Levene, 2011). Hurtado (2015) explained that RSS users can receive and syndicate updated data from data sources automatically.

Study of Brick Factory in 2007 on America's top 100 newspaper websites, with the title of "American Newspapers and the Internet: Threat or Opportunity?", showed that 96 out of 100 America's top online newspapers employed RSS technology. Li et al (2007) stated that there are about 75,000 new RSS feeds created and 1.2 million new stories posted daily by referring / according to the survey of Technorati.

Bross et al (2010) mapped and extracted data from blogosphere using RSS feeds. They developed feed crawler software, which is implemented in Groovy, a dynamic Java programming language. Hurtado (2015) used feedparser and web crawler guided by RSS feed to collect data/information from RSS feeds and HTML pages. In this project, feedparser, a Python library for parsing feeds is employed in this project to collect data using RSS feed.

2.3.3 Cloud computing

Abdulhamid (2019) stated that Eric Schmidt is probably the first to introduce the word "cloud computing" in his talk on Search Engine Strategies Conferences in 2006 as cited in Qian et al (2009). Qian et al (2009) described cloud computing as a kind of computing technique that provides IT services with low-cost computing units connected by networks. Kumar and Goudar (2012) mentioned that cloud computing is a Pay-per-Use-On-Demand mode to users. Users can use the modalities whenever demanded and only pay for the services they used (Priyanshu and Rizwan, 2018). National Institute of Standards and Technology (NIST) as cited in Kratzke (2018)

defined cloud computing based on three basic services, which are Infrastructure as a service (IaaS), Software as a service (PaaS) and Platform as a service (PaaS).

Cloud computing have some features and advantages. For instance, its scalability, on-demand services, quality of service, user-centric interface, autonomous system and pricing as mentioned by Prasad, Naik and Bapuji (2013). However, there are also issues and challenges in adopting cloud computing. Prasad, Naik and Bapuji (2013) claimed that the issues and challenges in adopting cloud computing are security, reliability, privacy, open standard, performance, bandwidth cost, long-term feasibility and legal issues.

Cloud computing is used in this project to run python script continuously for a long period of time. Without using cloud computing, local computer is required to turn on and run non-stop for an extensive amount of time. This is not healthy for a normal local computer. Other tasks to be done using the computer might also be interfered.

CHAPTER 3: RESEARCH METHODOLOGY

Python programming language is used throughout the entire project. In order to achive the objectives of this project and accomplish it, there are some important methods and techniques to be employed at different stages. Some methods or processes might need to be applied a few number of times for those needed unprocessed data when developing models consisting of numerical data and augmented data.

This chapter is divided into six parts as follows: Section 3.1 discusses on the collection of textual data (articles) and numerical data (index and stock prices). Section 3.2 ans 3.3 explains the processing steps for text data and numerical data respectively. The prepearation of data for modelling is presented in Section 3.4. Section 3.5 explains the development of models with machine learning. Lastly, Section 3.6 discusses on the evaluation of models using accuracy.

3.1 Data Collection

3.1.1 Textual data

For textual data, 300 articles from The Wall Street Journal obtained using web scraping are provided by Dr. Goh Yong Kheng. The 300 articles are from all categories of news in The Wall Street Journal from September 2017 to October 2017 (2 months). Besides, methods to collect data from the web such as web scraping and rss feed have been employed. Cloud computing is also used in some of the data collection methods.

3.1.1.1 Web scraping

Acoording to Saurkar, Pathare and Gode (2018), web scraping extracts and transforms unstructured data from the web into structured comprehensible data such as spreadsheets or comma-separated values (CSV) files. It then saves it into a file system or central database for future use of visualisation and analysis. Salerno and Boulware (2003) as cited in Draxl (2018) claimed that web scraping is the process of querying a source through Uniform Resource Locator (URL), retrieving the results page (HTML)

and parsing the page to obtain the results.

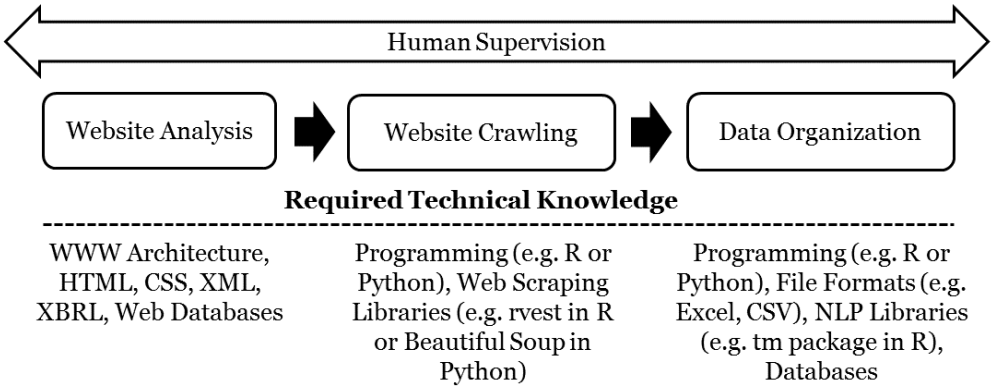

Figure 3.1.1: Phases in Web Scraping (adapted from Krotov and Tennyson (2018) as cited in Krotov and Silva (2018)).

Web scraping comprises three main phases that can be intertwined, which are website analysis, website crawling, and data organization as shown in *Figure 3.1.1*. However, some degree of human supervision is still needed throughout the whole process as it often cannot be fully automated.

Krotov and Silva (2018) explained that website analysis is the examination of a website's underlying structure in order to understand how and where the required data is stored, for later retrieval. A basic understanding is needed on the architecture and mark-up languages of World Wide Web such as HTML and XML, and several Web databases such as MSSQL and MySQL. For web crawling, a script to browse website and extract required data automatically is developed and run. The crawling script is usually developed using Python and R programming languages due to the availability of libraries or packages that allow automatic crawling and parsing of Web data. For instance, the Beautiful Soup library in Python and the "rvest" package in R. The last phase in web scraping is data organization. Cleaning, preprocessing and organization of the parsed data is needed to ensure that it allows further analysis. This is often done through programmatic approach using libraries and function. The Natural Language Processing (NLP) library and data manipulation functions in R and Python are useful for this purpose (Krotov and Silva, 2018).

In this project, a few python libraries such as requests, beautiful soup, selenium, time and json are needed to collect article data using web scraping. Two methods are used to extract online data due to the restrictions of websites. For websites that allow the use of requests (get) to extract data through URL, selenium is not employed as it requires a lot of computation powers and creates heavier task to the source. For instance, “https://www.fool.com”. Selenium is only used when the retrieval of page source or data by sending request is prohibited. For instance, “https://www.bloomberg.com/markets”. Requests and selenium libraries are used alternatively but not simultaneously. Beautiful Soup Core Development Team (2004) (as cited in (Hurtado, 2015)) stated that beautiful soup is a Python parser library that helps to remove HTML tags from contecnt and provide a clean text. Time module allows the script to rest (sleep) for a fixed time period in the web scraping process. This can avoid the occurance of Denial-to-service attack to the server. Json library is used to save the output data extarcted online.

Web scraping using request module is run in the cloud computer to continuously extract data. This is because cloud computing allows a script to be run continuously without stop even though exit from the server. A local computer is unable to do this without turning off. On the other hand, web scraping using selenium module is not run in cloud computer as it requires the server to be connected and turned on continuously. This also requires the local computer to be turned on continuously and makes no different between running in cloud or local. The cloud computing software used in this project is Digital Ocean.

Due to the limitation of time, web scraping is only done for two websites, particularly “https://www.fool.com” and “https://www.bloomberg.com/markets”. Various websites should be scrapped to have a deeper understanding about web scraping. Different problems and challenges will be faced when scraping different websites. Some websites may have no or less restriction and some may highly restricted web scraping activities.

3.1.1.2 Really simple syndication (RSS)

O'Shea and Levene (2011) stated that RSS provides a method to syndicate and aggregate online content, particularly the frequently updated works such as news, blog entries and HTML. The collection and syndication of updated data can be done automatically.

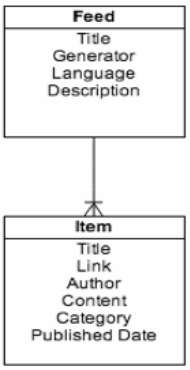

Figure 3.1.2: The RSS feed structure (Hurtado, 2015).

A RSS feed is composed by items (as shown in *Figure 3.1.2*). Items and feed have specific attributes which decribe each entity respectively. Though, all attributes or complete information are not necessary provided (Hurtado, 2015). This is the challenge that may be faced by RSS user if there is a lack of important information. Hurtado (2015) stated that the most common issues arose are incomplete content, non-categorised article, low resolution image, missing main information and missing author. The problem faced in this project is the lack of necessary and required attributes such as title, summary or published (date), particularly in items.

In this project, feedparser, time and json libraries are imported into Python for the aquisition of online frequently updated data using RSS feeds. Feedparser (Universal Feed Parder) is a module in Python for downloading and parsing syndicated fees such as RSS and Atom. Time module is used to measure the resting time for the script in the data acquisition using RSS. Since high frequency data collection may result in the same set of data being collected. Json library is for the storing of data extrated for future use.

The Python script to extract data using RSS is run in the cloud computer. This is due to the continuous running despite disconnected feature of cloud computing.

With this feature, the script can be executed repeatedly to continuously retrieve data. Continuous executing a script can only be done on a local computer if the computer is switched on. The use of cloud computing can increase the amount of data collected without bringing any disadvantages particularly to the local computer. The cloud computing software used is also Digital Ocean.

3.1.2 Numerical data

For numerical data, the highest, lowest, open and closing index prices for The Standard & Poor's 500 (S&P 500) index prices were downloaded from The Wall Street Journal. Besides, stock prices of the top 30 companies with the highest component weight listed in S&P 500 as at October 2018 are extracted. The stock prices of the 30 companies are used to develope model consisting numerical data, and model consisting the augmented textual and mumerical data. The index prices and stock prices will also be collected for September 2017 and October 2017 from Bloomberg.

3.2 Processing Textual Data

The articles are being imported into Python in dictionary data type. The textual data is reduced to include only the title and textual content of the article itself while ignoring other parts such as the authors, descriptions and keywords. Before starting to process the data, the articles with no content are removed. As a result, 4 article data (rows) have been removed from the 300 data, remaining 296 data. Textual data including both title and content are then being processed.

The processing of textual data are mainly being done using the Natural Language Toolkit (NLTK) library in Python. Firstly, the paragaph or sentences of the text data are tokenise using the word_tokenize imported from NLTK library. Next, the tokens of words are being cleaned by removing all the stopwords, punctuations and numbers, leaving only alphabets. The list of stopwords can be obtained from nltk.corpus. The words are then being stemmed using Porter Stemmer imported from nltk.stem. The frequency of occurance of words are multiplied by a factor of 2 if it presents in the title words. Besides, the list of words are reduced based on list of

sentiment words obtained online. The list of sentiment words is made up of positive and negative words obtained from http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html under "Opinion Lexicon". This is to ensure that all the words are sentimental and significant in predicting the market trend. The number of distinct words (columns) reduced from 11168 to 1856.

The occurance frequency of each word in each article are counted afterwards. Both the frequencies and the words appearing in each article are being fitted into the dataframe of pandas library for futher analysis. Besides, some of the articles are being removed from the dataset due to the lack of significant information, for instance the published date, as the market trend of the day the article is published cannot be determined. Words with the total frequency of appearance in all articles less than or equal to 3 are also removed. The number of distinct words further reduced from 1856 to 822. The matrix for textual data now has the size of (296, 822).

3.3 Processing Numerical Data

Among all the numerical data obtained, only the closing prices of the top 30 stocks with highest weight in S&P 500 and the index itself will be used in this project. The historical closing prices of 30 stock and S&P 500 index are imported as a pandas dataframe respectively.

Between the two dataframes created, only the data in S&P index price dataframe needs to be further processed. The target, market trend is determined based on the percentage change of index price as compared to the day before. All the three models are having the same target. Only percentage change of more than 0.05% on index price will be considered as increase or decrease, others it is considered unchanged. Positive percentage change of more than 0.05% indicates upward trend while negative percentage change of more tan 0.05 indicates downward trend. Price percentage change of value less than or equal to 0.05% represents neutral as the price doesn't fluctaute much. The percentage change of index price as compared to the day before is calculated as below:

$$\text{Percentage Change of index price} = \frac{\text{Today's index price} - \text{Yesterday's index price}}{\text{Yesterday's index price}} \times 100\%$$

3.4 Data Preparation for Modelling

This section explains how to prepare data for the use of model development. For model using only text data, the dataframe consisting frequency of words (obtained from *3.2 Processing of textual data)* and the dataframe consisting target (obtained from *3.3 Processing numerical data)* are joined together. The dataframes are joined while removing the rows that do not have matched index. This is because some of the article published during the market closed and some market opening day may not have article published. Each input data (article) must have a target (market trend) in order to be used to build model. 37 data are removed, remaining 259 data for modeling.

For model using only numerical data, the dataframe consisting the 30 stock prices and the dataframe consisting target are joined together. Both of the dataframe are obtained from *3.3 Processing numerical data*. No data is removed from the joining process since the index price of S&P 500 is derived partly from the prices of the 30 stocks in S&P 500 index.

For model using the augmented data of text and numerical, the dataframes prepared for building textual data model and numerical data model are joined. No data is removed from this joining process as the features without target or the target without features are removed during the data preparation for text data modeling. Since there will be duplicate columns of target, one of it is removed.

3.5 Model Development with Machine Learning

Scikit-learn library is necessary for all the process in this stage and hence will be imported. The algorithms used to develop the models are also obtained from scikit-learn library such as logistic regression, support vector machine, naive bayes, neural network and decision tree.

For textual data model, the features or input data (X) is the part of its dataframe (prepared in *3.4 Data preparation for modelling*) that consists of textual information which are the word counts of each word in each article repectively. Whereas, the prediciton or target that the model makes (y) will be the directional movement of index prices (which is also the market trend) such as up, neutral or down for the day the article is published. The y-value refers the target column of its dataframe.

For numerical data model, the features (X) is the part of the dataframe (prepared in *3.4 Data preparation for modelling*) with 30 stock prices of companies listed in S&P 500 while the prediction (y) will be target in its dataframe.

For augmented data model, the input data (X) is the part of the dataframe (prepared in *3.4 Data preparation for modelling*) consisting the frequency of words and prices of the 30 stocks. Whereas, the prediction (y) is the target in its dataframe.

The prediction for text data model and augmented data model is exactly the same. However, the target from the all the three models is also calculated in the same way as mentioned in *3.3 Processing of numerical data* and originated from the same dataframe which obatined from the *3.3 Processing of numerical data*.

The following steps are gone through to develop each of the models. Before conducting machine learning to develop the model, the data is split into training and testing set. 70% of the respective data used to developing model will be used as training set to train model and the remaining 30% will be used to test the performance of model. There are some basic steps that are necessary to develop all types of machine learning algorithms. Firstly, both the data (X and y) are loaded and splitted into training set and testing set. Next, a classifier is created depending on the model to be used. For example, the classfier for logistic regression is created for model employing logistic regression. The model is then trained to make prediction. After the training is done, prediciton is performed and the model score is determined. These

processes are the same for all three types of models developed. Hence, they are performed three times in order to build the three models.

Since the problem of this study is a classification problem, classification type algorithm are employed. A few models are developed based on different type of classification algorithms such as logistic regression, support vector machines (SVM), naive bayes (NB), neural network and decision tree.

3.5.1 Logistic regression

Logistic regression is a linear classification algorithm rather than regression. Logistic function is used to model the probability of possible outcomes which is the probability of success for a trial (scikit-learn, n.d.).
The logistic function is given by:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

where x is the $x - value$ of the sigmoid's midpoint, and $(x - x_0)$ is the horizontal translation of the logistic function; L is the curve's maximum value or horizontal asymptote; and k is the steepness of the curve (Cruzan, n.d.).

3.5.2 Support vector machine (SVM)

In this study, we are focusing on the classification usage of SVM which known as support vector classification (SVC). The decision function of SVC is given by :

$$sgn(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

There are two main types of SVC that can perform multiclass classfication which are linear SVC and SVC (scikit-learn, n.d.). Linear SVC performs linear classification while SVC performs non-linear or kernel classification. Linear SVC is also known as linear SVM and SVC is also called kernel SVM.

3.5.3 Naive bayes

The prediction rule for Naive Bayes method as follow:

$$\hat{y} = arg \max_{y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

There are three types of naive bayes classifier which are bernoulli, multinomial and gaussian. Bernoulli naive bayes is for binary data, multinomial naive bayes is for count data and gaussian naive bayes is for continuous data. The difference of the three naive bayes classifier are the assumptions they make on the distribution of $P(x_i|y)$ (scikit-learn, n.d.).

3.5.4 Neural network

In this study, we will employed the multi-layer preceptrons (MLP) neural network for classification. MLP can learn linear or non-linear function for classification and regression. It is not the same as logistic regression because it allows the presence of one or more non-linear layers which known as hidden layers in between the input and output layer (scikit-learn, n.d.).

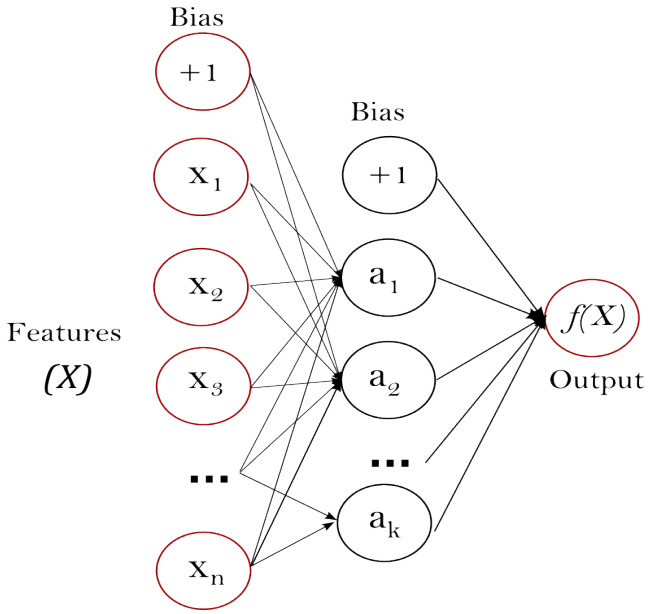

Figure 3.5.1: One hidden layer MLP (scikit-learn, n.d.).

3.5.5 Decision tree

Decision tree is a hierarchy of the possible outcomes of a series of choices leading to a decision (Lucidchart, n.d.). It is a tree made up of nodes, branches and leaves. Each node represents a test on attribute or feature, each branch represents an outcome of the

test and each leaf represents an class label (Saloni, n.d.). Leaf (decision) is the end of the branch that doesn't split anymore (Prashant, 2017). Decision tree can be used for regression and classification.

A tree starts with a single node that branches out into possible outcomes and each of the outcomes becomes nodes that braches into other possibilities (Lucidchart, n.d.). The splitting of data into subsets at node is based the attribute value test. This process is repeated until all the subsets at nodes has the same value (outcome) for the target variable or when further splitting has no value to the predictions (Saloni, n.d.).

3.6 Evaluation of Models

The evaluation of models can be done based on their performance and scoring on various parameters such as accuracy and precision by using Scikit-learn library. There are some packages that can be imported from sklearn.metrics to evaluate model performance such as classification_report, precision_recall_curve and roc_curve. The model score of different machine learning algorithms may have different meaning depending on their nature. However, all the models developed in this project have classification performance and hence their model scores represent (mean) accuracy. Accuracy, which is the fraction of correct predictions over n samples, is computed as follow:

$Let\ \hat{y}_i\ be\ the\ predicted\ value\ of\ the\ i^{th}\ sample,$
$y_i be\ the\ true\ value\ of\ the\ i^{th}\ sample, and$
$n\ be\ the\ number\ of\ samples.$

$$Accuracy(y, \hat{y}) = \frac{1}{n}\sum_{\square=0}^{n-1} 1(\hat{y}_i = y_i)$$

where $1(x)$ refers to indicator function (scikit-learn, n.d.).

The prediciting power of the models in this project is evaluated based on the model score for training dataset, model score for testing dataset, and the difference between the score for training set and score for testing set. The difference between the scores is used to determine whether the model developed is overfitted to the training dataset. High difference in scores with higher score on training set indicates

overfitting. Model score for testing set is used to evalaute the accuracy of the model in predicting financial market trends.

CHAPTER 4: RESULTS AND DISCUSSIONS

This chapter is divided into two main parts which are Section 4.1 Data exploration and Section 4.2 Model performance. Total frequency of each word (Section 4.1.1), average frequency of each word (Section 4.1.2) and standard deviation of each word (Section 4.1.3) are analysed in Section 4.1. On the other hand, Section 4.2.1, 4.2.2 and 4.2.3 present and discuss the model scores of text data model, numerical data model and augmented data model respectively. The comparison between the three models is discussed in Section 4.2.4.

4.1 Data Exploration

Exploration on the data provided is conducted in order to have a better understanding on the dataset and discover any significant pattern that may present. The data exploration is done after the processing of data and before the matching of textual and numerical data during machine learning. Therefore, we still have 296 sets of data. Our data consists of 296 articles and 1856 words. The data collected through web scraping and RSS is not explored, analysed and included in the development of models due to the limitation of time.

4.1.1 Total frequency of each word

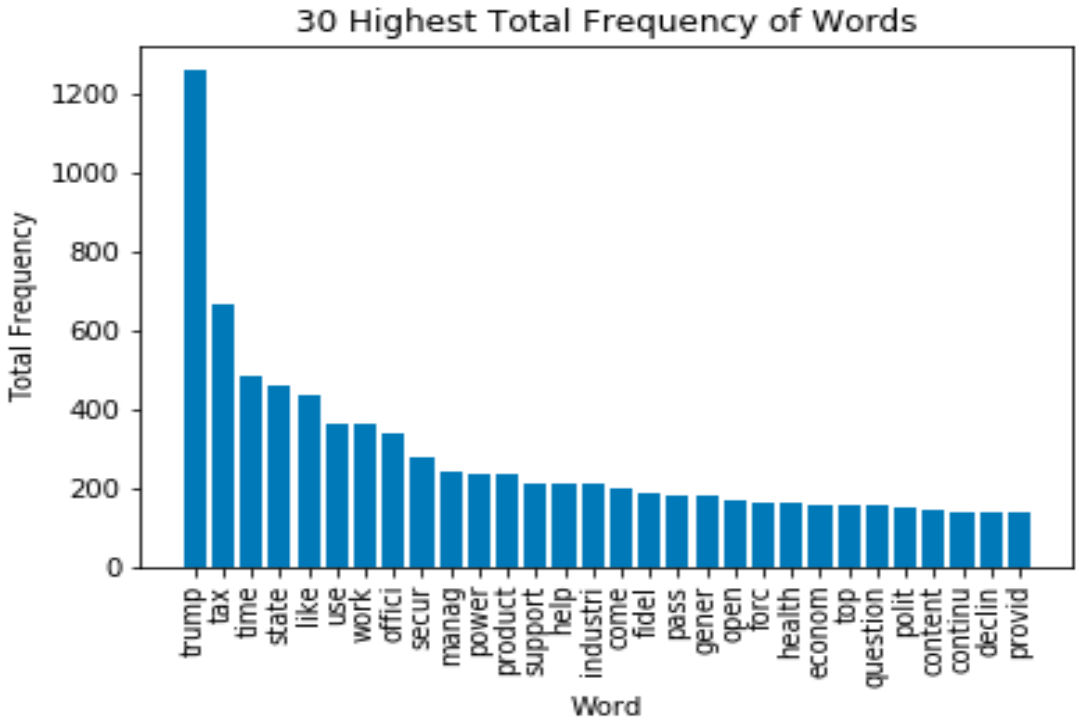

Figure 4.1.1: The 30 highest total frequency of words.

Total frequency of a particular word is the sum of the frequencies of that word in all articles. The collection of the total frequency for each word is then ranked from the highest to the lowest. Words with the highest 30th total frequency are plotted in descending order.

From the plot, we noticed that “trump” has the highest total frequency which is 1258 among all the words, followed by “tax” with total frequency of 664 and “time” with total frequency of 486. The total frequency drops sharply from the highest (“tax”) to the second highest (“time”) with approximately half of the value. The less steep decrease of total frequency from the “tax” to “time” (second highest to third highest) is around 26% which is a lot more lesser as compared to the previous drop. There are only slight drops on total frequency after the word ”time”. The total frequency starting from “time” decreases steadily with only slightly significant drops from “like” to “use” and from “offici” to “secur”. The word “trump” which most likely represents Donald Trump has the highest frequency of occurrence in all articles since the United States’ election of president is just ended. Hence, a lot authors

discussed and related the events in financial market to Donald Trump who was the new elected president.

4.1.2 Average frequency of each word

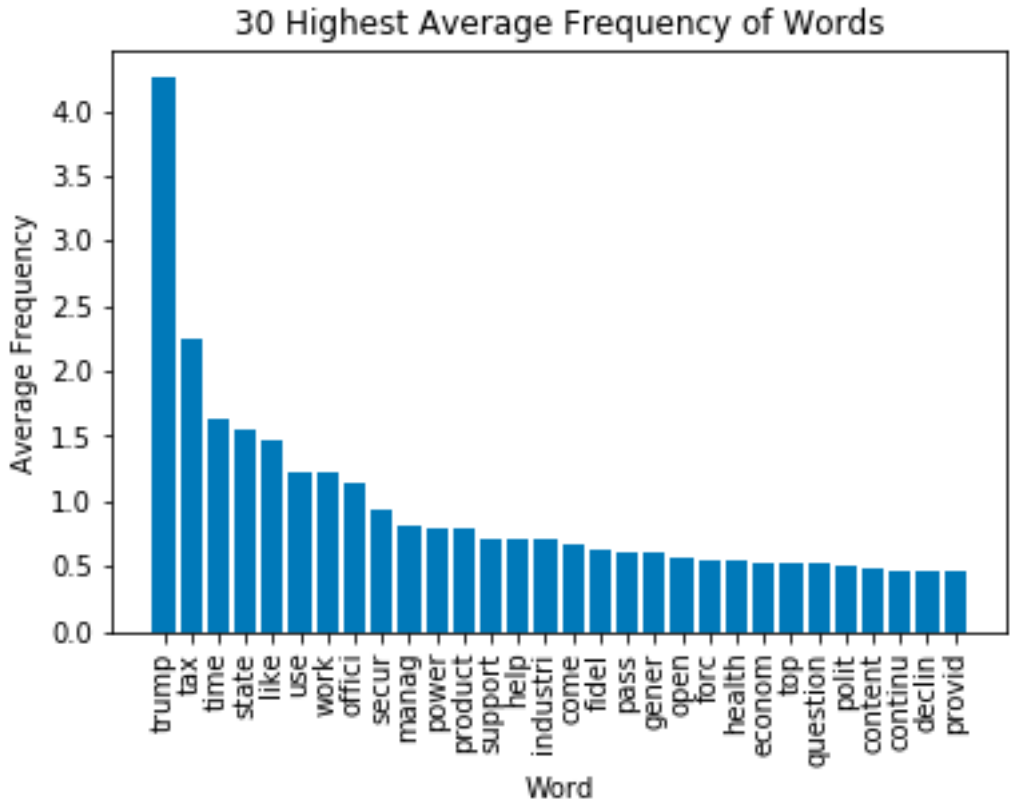

Figure 4.1.2: The 30 highest average frequency of words.

Average frequency of each word is calculated by taking the total frequency of that word to divide the total number of articles which is 296. The words is arranged in descending order based on their average frequency and the 30 highest average frequency of words is plotted.

The word with the highest average frequency of 4.25 is “trump”, followed by “tax” with average frequency of 2.24 and “time” with average frequency of 1.64. The plot has the same pattern as the plot in *Figure 4.1.1*. Sharp drop from highest to second highest value and less steep fall from second highest to third highest while the drops afterwards are steady with a slightly obvious drop fom “like” to “use” and from”offici” to ”secur”. In addition, the set of words with 30 highest average frequency is exactly the same as the set of words with 30 highest total frequency. This

is because the computation of average frequency using the total frequency divides the total number of articles which is a constant. Hence, the pattern of the plot remains unchanged. In average, “trump” has the highest frequency in each article due to the same reason as explained in Section 4.1.2.

4.1.3 Standard deviation of each word

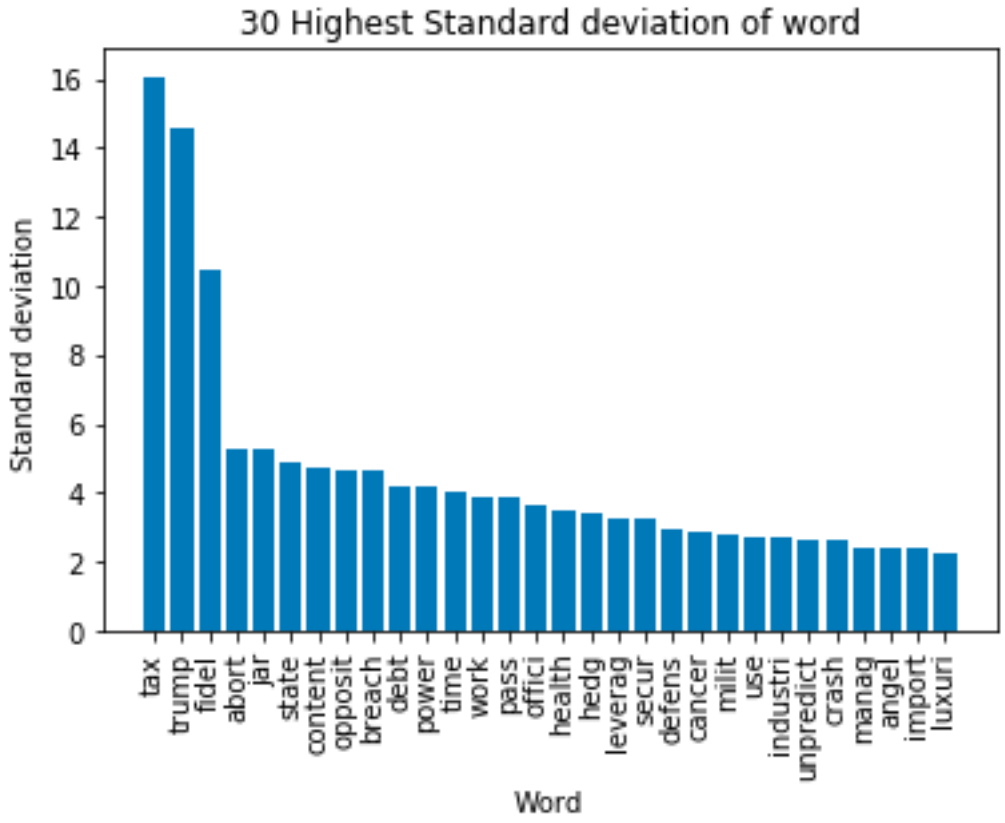

Figure 4.1.3: The 30 highest standard deviation on the frequency of words.

Standard deviation on the frequency of a word is computed based on all the frequency in all articles for a particular word. Besides, standard deviation represents the volatility of the frequency of a word. The words are ranked according to their standard deviation values and the top 30 standard deviation’s words are plotted.

From this plot, “tax” becomes the word with highest standard devaition on frequency (16.06), followed by “trump” with second highest standard deviation of 14.55 and “fidel” with third highest standard devation of 10.46. This plot has a slightly different pattern as compared to the previous two plots. The drop in the standard deviation of the highest (“tax”) to the second highest (“trump”) is less steep

as compared to the drop from “trump” (second highest) to “fidel” (third highest) and the drop from “fidel” to ”abort”(forth highest). All the falls begin from “abort” is relatively steady and less significant. The set of words with 30 highest standard deviation is different from the set of words with 30 highest total frequency or with 30 highest average frequency.

4.2 Model Performance

The score for training set and score for testing set for all the algorithms in *Table 4.2.1, 4.2.2 and 4.2.3* is represented by the accuracy of testing data and training data respectively, in predicting market trend. For classification algorithms, it is are often to represent its scores using accuracy. Our main focus is on the difference between training and testing scores that is used to determine the occurance of overfitting. Besides, the score for testing set which is used to determine the accuracy of model in predicting the testing data is also important in evaluating the model performance. Score for training set is used to determine the accuracy of model in predicting the training data. Training data is the data used to train the model. Hence, it determines how well the model trained by training data is able to predict the training data. It appears to be the model fitting score and is for complementing purpose in evaluting the model performance.

4.2.1 Text data model

Algorithm	Score for Training Set	Score for Testing Set	Difference Between Training and Testing Scores
Logistic regression	0.988950276	0.435897436	0.55305284
Linear support vector machine	0.994475138	0.423076923	0.571398215
Kernel support vector machine	0.994475138	0.5	0.494475138
Bernoulli naive bayes	0.878453039	0.41025641	0.468196628
Multinomial naive bayes	0.82320442	0.525641026	0.297563394
Gaussian naive bayes	0.939226519	0.538461538	0.400764981
Neural network	0.994475138	0.346153846	0.648321292
Decision tree	0.994475138	0.371794872	0.622680266

Table 4.2.1: The score for training set, score for testing set and difference between scores of text data models trained using eight different algorithms respectively.

From *Table 4.2.1*, the significantly large difference between score for training dataset and score for testing dataset for model trained using neural network and model trained using decision tree indicates a considerably strong degree of overfitting. Models employing logistic regression and linear support vector machine (linear SVM) respectively also exhibit overfitting error due to the high difference between scores. Moreover, kernel support vector (kernel SVM), bernoulli naive bayes (NB) and gaussian naive bayes developed less overfit models with lower difference of training data and sting data scores. The lowest difference between scores of multinomial naive bayes model indicates that it has only slight overfit error as compared to other models.

Multinomial NB model had the lowest difference of scores among all trained models, including bernoulli NB model and gaussian NB. This indicates marginal overfitting in multinomial NB model. The model has high score for testing data, which is only slightly lower than gaussian NB. Hence, it is the best model among the NB classifier and also among all models trained. Multinomial NB is better than the other NB classifier in this model because it is for count data while bernoulli NB expects binary features and gaussian NB is for continuous data. Only multinomial NB

for count data matches the model feature (frequency of word) data type. Hence, it is the most suitable and best classifier out of the three NB classifier.

Comparing the two SVM alogorithms, the scores for training set of both SVM are the same while the score for testing set of kernel (non-linear) SVM is higher than that of linear SVM. This is because kernel SVM is used to perform non-linear or kernel classification whereas linear SVM is for linear classification. Since the data used to train the model is not linear, kernel SVM algorithm is more suitable and accurate than linear SVM in predicting the target.

Same training data scores are obtained by four models each employing different algorithm, for instance the linear SVM, kernel SVM, neural network and decision tree. However, the testing data scores are different, with the highest among the four is kernel SVM followed by linear SVM, decision tree and lastly neural network. The increasing overfitting degree sequence among the four is the decreasing sequence in score for testing data. The higher the score for testing data, the lower the degree of overfitting, when the score for traing data is constant. Hence, kernel SVM trained the best model among the four algorithms. Bernoulli NB also has very low difference between scores for training and testing data than the others.

Although the model developed using gaussian NB has the highest score for testing data, its score for training data is a lot more higher than the testing data score. This high difference indicates strong overfitting of the model. In fact, multinomial NB built the best models among the alogorithms used as it has the lowest difference of scores, reasonable score for training set and high score for testing set. On the other hand, the worst algorithm used to develop model is neural network as it has the highest overfitting (difference between scores) and lowest score for testing data.

All models built based on text data has a significant amount of difference between training and testing scores (ranges from 0.47 to 0.65) besides multinomial NB model (0.30). All models except multinomial NB model have strong indication of overfitting. The minimum and maximum difference between scores in this model are widely dispersed with a difference of about 0.35. Low testing data scores (less than or

around 0.5) are also obtained by all the models trained. This shows that the prediction power of the models developed is more worse or the same as random guessing (accuracy of 0.5). Text data extracted from news articles is unable to predict the directional movement of index prices well.

4.2.2 Numerical data model

Algorithm	Score for Training Set	Score for Testing Set	Difference Between Training and Testing Scores
Logistic regression	0.964285714	0.384615385	0.57967033
Linear support vector machine	0.75	0.461538462	0.288461538
Kernel support vector machine	1	0.461538462	0.538461538
Bernoulli naive bayes	0.75	0.461538462	0.288461538
Multinomial naive bayes	0.75	0.461538462	0.288461538
Gaussian naive bayes	0.535714286	0.153846154	0.381868132
Neural network	0.75	0.461538462	0.288461538
Decision tree	1	0.461538462	0.538461538

Table 4.2.2: The score for training set, score for testing set and difference between scores of numerical data models trained using eight different algorithms respectively.

From *Table 4.2.2*, models developed using logistic regression, kernel SVM and decision tree repectively exhibit strong overfitting due to high difference between training and testing data scores as compared to other alogorithms. Whilst kernel SVM and decision tree are having the same difference between scores thus contain approximately same amount of overfitting error. Overfitting is indicated in the gaussian NB model by the moderate difference between scores for training and testing data. The difference of scores for the remaining algorithms model are the same. Linear SVM, bernoulli NB, multinomial NB and neural network have the lowest difference between scores and hence are the least overfitted in all the models using numerical data. Most of the models obtain the same set of scores with others. Only logistic regression and gaussian NB have their own set of score for training data, score for testing data and difference in scores.

Among the NB algorithms, bernoulli NB and multinomial NB have the same score for training set, scorefor testing set and difference of scores. They have lower difference between scores, higher training data score and higher testing data score than that of gaussian NB. Therefore, gaussian NB model is more overfitted as compared to bernoulli NB model and multinomial NB model. Bernoulli NB and multinomial NB can build better model with higher accuracy than gaussian NB.

For SVM, both the testing dataset scores for linear and kernel SVM are the same. Kernel SVM has higher score for training data compared to linear SVM, leading to larger difference between score and stronger indication of overfitting. Hence, linear SVM model is a better model than kernel SVM to predict financial market sentiment. This implies that linear classification algorithm is more accurate and suitable in fitting and predicting data. The numerical data (features) of this model and the relationship with its target is in linear form.

In addition, the score for testing data of six algorithm models which are linear SVM, kernel SVM, bernoulli NB, multinomial NB, neural network and decision tree are the same. Among the six algorithms, linear SVM, bernoulli NB, multinomial NB and neural network have the same low degree of overfitting (lowest difference in scores) due to equal score for training data. Hence, the predicting performance and power for them are excatly the same in this numerical data model. They will make the best and most accurate predictions as compared to the other alogorithm trained models. Kernel SVM and decision tree also have same (medium) predicting power due to same score for training data and thus same (moderate) overfitting. Furthermore, gaussian NB has below avearage score for training data and score for testing data.

To conclude, there are a few best algorithms employed for the numerical data model which are equally good. These are linear SVM, bernoulli NB, multinomial NB and neural network as they are having the lowest degress of overfitting (lowest difference between training data and testing scores), reasonable score for training set and the highest score for testing set. Although models using kernel SVM and decision tree respectively also have the highest testing data scores, they are not considered as the best model due to stong indication of overfitting (resulted from high difference of

scores). This is because these models might get lower score for testing set when tested on other datasets since they are overfitted to the original dataset and obtain high (the maximum) score for training set. On the other hand, the worst algorithm build model is logistic regression due to its substantial amount of difference between training (fitting) and testing (performance) scores despite It has very strong indication of overfitting even though it obtains moderate score for testing data. Its values are a lot lower than the others but its difference between score is still considered normal.

All the models trained using numerical data has an obvious and near amount of difference between scores (ranges from 0.29 to 0.58). They exhibit significant overfitting. The difference between the lowest and highest difference of scores is around 0.29. Less than 0.5 of scores for testing set are obtained by all the models. All models developed based on numerical data performed more worse than random guessing (accuracy of 0.5). Model using numerical data is not useful in predicting the market sentiments.

4.2.3 Augmented data model

Algorithm	Score for Training Set	Score for Testing Set	Difference Between Training and Testing Scores
Logistic regression	1	0.923076923	0.076923077
Linear support vector machine	0.895027624	0.730769231	0.164258394
Kernel support vector machine	1	0.512820513	0.487179487
Bernoulli naive bayes	0.834254144	0.525641026	0.308613118
Multinomial naive bayes	0.817679558	0.512820513	0.304859045
Gaussian naive bayes	0.994475138	0.602564103	0.391911036
Neural network	0.867403315	0.807692308	0.059711007
Decision tree	1	0.987179487	0.012820513

Table 4.2.3: The score for training set, score for testing set and difference between scores of augmented data models trained using eight different algorithms respectively.

From *Table 4.2.3*, model employing kernel SVM shows the strongest overfitting behaviour due to its highest difference between training and testing scores. The three NB classifier, bernoulli NB, multinomial NB and gaussian NB obtain high difference

between scores implying overfitting. Model trained using linear SVM indicates gradual overfitting as it has an unsignificant difference of scores. Logistic regression, neural network and decision tree models can be concluded as no overfitting. This is because they have realtively small difference between training data and testing data score (less than 0.08).

Among the NB classifier, gaussian NB trained model has the highest difference between training set and testing set scores, followed by bernoulli NB trained model and the lowest is multinomial NB trained model. The amount of difference beatween scores of gaussian NB is significantly larger than bernoulli NB and multinomial NB. Gaussian NB shows relatively strong indication of overfitting while other two NB classifier exihibits about the same degree of overfitting. This is because gaussian NB is for continuous data whereas bernoulli NB for binary data and multinomial for count data. The dataset for this model is not in continuous form hence obtaining poor performance in model employing gaussian NB. Bernoulli NB has slightly higher difference of scores and thus overfitting than multinomial NB since the features are not only made up of two values. The augmented data consists of text (word frequency) data and numerical (index prices) data. Multinomial NB is more suitable to be employed.

Model employing linear SVM performs better than the one employing kernel SVM. Linear SVM has significantly lower difference between training and testing scores and hence indicates marginal overfitting as compared to kernel SVM. The testing dataset score of linear SVM is also a lot more higher than that of kernel SVM. This results shows that linear algorithm (linear SVM) is more suitable and accurate than non-linear algorithm (kernel SVM) to predict the target. However, this is illogical since the data is not linearand should not have any linear relationship. Illogical result obatined is probably due to the bias and insufficient data. This might be solved by using larger and balance dataset in training the model.

Kernel SVM is having the same score for testing set as multinomial NB. However, they do not have the same model fitting score, and difference between training and testing score. Since multinomial NB has lower indication of overfitting

due to smaller difference of scores, it develops a better model as compared to kernel SVM. The high difference of scores and unreasonable score for training set (1) of kernel SVM model indicates a very stong overfitting. In fact, it has highest difference of scores and strongest indication of overfitting among all models trained.

The best algorithm to train the augmented model is decision tree. Model employing decision tree has the lowest and marginal difference between training and testing scores, showing almost no overfitting. It also obtains the highest performace score with a reasonable amount of model fitting score. Decision tree model has the best predicting power to determine the market trend. On the other hand, the worst algorithm for this model is kernel SVM. Kernel SVM trained model has the highest and significant amount of difference between training and testing scores, indicating high degree of overfitting. It also obtains the lowest score for testing and an unreasonable score for training data of 1.

All algorithms developed models with moderate or marginal overfitting. Most of the models have average amount of difference between training and testing scores (0.30 to 0.41) while some obtain a relatively minimal difference of scores (less than 0.08). The dispersion between the lowest and largest difference of scores are high with the difference of (0.474). In general, the models built predict better than random guessing (scores for testing set more than 0.5). Only three out of the eight algorithms trained model with approximately the same predicting power as random guessing (accuracy of 0.5).

4.2.4 Comparison between models

Among the three type of models built, which are text data model, numerical data model and augmented data model, in general, augmented data model makes the best predictions, followed by text data model and the last is numerical data model. Model based on augmented data has scores for testing set of more than or equal to 0.5. The scores for testing set of model based on text data are less than or equal to 0.5. Numerical data model obtain scores for testing set of less than 0.5 (less than 0.46 in actual fact). Prediction better than random guessing (accuracy of 0.5) is only made by augmented data model. Whilst, the other two models are more worse or just the same

as random guessing. Hence, it is showed that models consisting of only text data and only numerical data are unable to predict the financial market sentiment. It is only possible when model is built based on the augmented data of both the text and numerical data.

Moreover, the dispersion in the scores for testing set of augmented data model is the highest (difference between maximum and minimum of 0.47) which ranges from 0.51 to 0.99. Numerical data model has the medium dispersion of 0.31 (ranges from 0.15 to 0.46) due to the presence of an outlier (relatively low score for testing set of gaussian NB). The scores for testing set of numerical data model are mostly the same (0.46) except two values. One of them is from the abovementioned gaussian NB (0.15) and the other belongs to logistic regression (0.38). The scores for testing set of gaussian NB in numerical data model is also the lowest among all the scores for testing set (including those from augmented and text data model). Text data model has the lowest dispersion in scores for testing set (0.19) which ranges from 0.35 to 0.54.

Besides, in overall, the differences between training set and testing set scores are generally lowest in augmented data model, followed by numerical data model and text data model. Model using only text data shows strongest indication of overfitting. Overfitting is lesser in numerical data model and the minimal in model using augmented data. Some of the models developed using augmented data even do not show indication of overfitting as the differences are too marginal.

Furthermore, the dispersion in the differences between training and testing scores is the highest in augmented data model (0.47) and lowest in numerical data model (0.29). The difference between maximum and minimum of model based on text data is 0.35 which falls between that of augmented data model and numerical data model. The dispersion in difference of scores is the highest in model based on augmented data despite having the lowest difference of scores among all models. This is because its lowest difference of scores is very low (0.01) as compared to all the difference of scores including those from text data model and numerical data model.

In addition, the dispersion in score for training set of numerical data model is the highest (0.46) which ranges from 0.54 to 1, due to the very low score for training set of gaussian NB (outlier). The scores for training set of model based on numerical data are mostly from 0.75 to 1 except the one from gaussian NB (0.54). Text data model and augmented data model have almost the same dispersion in score for training set (0.17 and 0.18 respectively). The training set scores of the respective model do not differ much from one another.

CHAPTER 5: CONCLUSION

This chapter is divided into three parts as follows: Section 5.1 discusses on the conclusion obtained in this report. Section 5.2 presents the limitation of this project. Section 5.3 gives suggestions on possible inprovements and future works.

5.1 Conclusion

The main purpose of this project is to develop model with augemented text and numerical data and compare it with model using only textual data and model using only numerical data. The text data is the news articles from The Wall Street Journal and the frequency of words in each article is used in developing models. There are two types of numerical data in this project, which are S&P 500 index prices and the individual stock prices. The S&P 500 index prices is used as our target for machine learning since index price can capture the financial market sentiment better as compared to individual stock prices. This is because it involved a lot of individual stocks in different industries. Index prices is used to find the financial market sentiment for that particular day and financial market sentiment is the thing that we want to predict and analyse. Wheras the individual stock prices is used in developing model consisting only numerical data and model consisting the augmented data to predict the movement of index price and the overall financial market trend. Movement of stock prices or market trend is based on the financial market sentiment. Therefore, we can achieve our goal to analyse financial market sentiment by using news article and/or prices of individual stocks. At the same time, we manage to predict the market trend.

It is proved that model developed based on the augmented data predicts the directional movements of index prices better than the model based on only text data and model based on only numerical data. Model considering both the effects in text and numerical data can better analyse and predict the financial market sentiment. In addition, it is shown that only model using the augmented data has better predicting power than random guessing. Model using text data and model using numerical data have about the same or more worst predicting power than random guessing. It is

proposed that useful, valuable and accurate prediction can only be done using the augmented text and numerical data. From all of the foregoing, augmented model has great potential in predicting the financial market sentiment.

5.2 Limitation

The limitation of time is the main limitation of this project. Due to the restricted time, the data collected using web scraping and RSS is not explored and analysed. The data is not used to train models as the data is not processed. Short period and marginal amount of articles has been collected. Besides, constricted data is obtained as most of the websites with news articles need to be paid and require subscription.

The algorithms employed in developing models are not studied and investigated thoroughly. As a result, models developed might be inaccurate and bias. The features, particularly word frequency are not selected in a perfect way. Further understanding of the data is required to select better features and employ more suitable algorithm in training models.

Moreover, model built using word frequency is unable to predict the market sentiment as the sentiment and meaning of a sentence can be different from the meaning of the individual word. Hence, the models developed in this project have limitation in predicting the market trend.

5.3 Future Work

In future, more data over a longer period can be extracted and used to develop the models in order to obtain a more comprehensive results and predictions. More work is necessary to improve the model through some refinement techniques. Tuning of the algorithms employed to train the models can be further studied and investigated. Further understanding and exploration on the features is needed to employ a more appropriate algorithm in developing models. Variety of data from various sources such as social media (Twitter) can be used in predicting the financial market sentiment. In addition, phrases or pattern of words should be taken into consideration in predicting the sentiment of that particular data. This is because a single word is

unable to represent a sentiment as the meaning might be inverted if “not” presents in front of that particular word.

REFERENCES

Abdulhamid, U. N., 2019. An Overview: Internet of Things, 5G Communication System and Cloud Computing. [Online]. Available at: https://www.academia.edu/38484250/Literature_Revie1.pdf [Accessed 23 March 2019].

Akita, R., Yoshihara, A., Matsubara, T. and Uehara, K., 2016. Deep learning for stock prediction using numerical and textual information. *IEEE/ACIS 15th International Conference on Computer and Information Science*. Available at: https://www.researchgate.net/publication/306925671_Deep_learning_for_stock_prediction_using_numerical_and_textual_information [Accessed 9 July 2018].

Bollen, J., Mao, H. and Zeng, X.J., 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1), pp. 1-8.

Brick Factory, 2007. American Newspapers and the Internet: Threat or Opportunity?. [Online]. Available at: https://blog.thebrickfactory.com/2007/07/american-newspapers-and-the-internet-threat-or-opportunity/ [Accessed 27 March 2019].

Bross, J., Quasthoff, M., Berger, P., Hennig, P. and Meinel, C., 2010. Mapping the Blogosphere with RSS-Feeds. *24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 453-460. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5474737 [Accessed 25 March 2019].

Cavallo, A., 2013. Online and official price indexes: Measuring Argentina's inflation. *Journal of Monetary Economics*, 60(2), pp. 152-165.

Cruzan, J., n.d. Logistic functions. [Online]. Available at:

http://xaktly.com/LogisticFunctions.html [Accessed 13 August 2018].

Das, S. R., 2014. Text and Context: Language Analytics in Finance. *Foundations and Trends in Finance,* 8(3), pp. 145–260.

Draxl, V., 2018. Web Scraping Data Extraction from websites. [Online]. Available at: https://www.academia.edu/35901535/BACHELOR_PAPER_Web_Scraping_Data_Extraction_from_websites [Accessed 20 March 2019].

Fama, E.F., 1965. The behavior of stock-market prices. *The Journal of Business*, 38(1), pp. 34–105.

Fama, E.F., 1970. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25, pp. 383-417.

Fortuny, E.J., Smedt, T.D., Martens, D. and Daelemans, W., 2014. Evaluating and understanding text-based stock price prediction models. *Information Processing & Management*, [E-journal] 50, pp. 426-441.

Gidófalvi, G., 2001. *Using News Articles to Predict Stock Price Movements.* [Online]. Available at: http://cseweb.ucsd.edu/~elkan/254spring01/gidofalvirep.pdf [Accessed 25 March 2019].

Hessisches Statistisches Landesamt, 2018. *Web scraping from company websites and machine learning for the purposes of gaining new digital data*. [Online]. Available at: https://statistik.hessen.de/sites/statistik.hessen.de/files/Webscraping_english.pdf [Accessed 15 March 2019].

Hoekstra, R., Bosch, O. T. and Harteveld, F., 2012. Automated data collection from web sources for official statistics: First experiences. *Statistical Journal of the IAOS,* 28(3), pp. 99-111.

Hurtado, J. F., 2015. Automated System for Improving RSS Feeds Data Quality. arXiv, [Online]. Available at: https://arxiv.org/pdf/1504.01433.pdf [Accessed 24 March 2019].

Joshi, K., Bharathi, H. N. and Jyothi, R., 2016. Stock Trend Prediction Using News Sentiment Analysis. *arXiv*, [Online]. Available at: https://arxiv.org/pdf/1607.01958.pdf [Accessed 15 August 2018].

Kratzke, N., 2018. A Brief History of Cloud Application Architectures. *Applied Sciences*, 8 (1368). Available at: https://www.researchgate.net/publication/327014262_A_Brief_History_of_Cloud_Application_Architectures [Accessed 20 March 2019].

Krotov, V. and Silva, L., 2018. Legality and Ethics of Web Scraping. *Twenty-fourth Americas Conference on Information Systems.* New Orleans, United States.

Li, X., Yan, J., Deng, Z., Ji, L., Fan, W., Zhang, B. and Chen, Z., 2007. A Novel Clustering-based RSS Aggregator. *Proceedings of the 16th International Conference on World Wide Web*. Banff, Alberta, Canada.

Liu, B., Hu, M. Q. and Cheng, J. S., 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. *Proceedings of the 14th International World Wide Web conference*. Chiba, Japan.

Lucidchart. n.d. *What is a Decision Tree Diagram*. [Online]. Available at: https://www.lucidchart.com/pages/decision-tree [Accessed 21 March 2019].

Malkiel, B.G. and McCue, K., 1985. *A random walk down Wall Street*. New York: Norton.

Mittermayer, M.-A. and Knolmayer, G.F., 2006. NewsCATS: A news categorization

and trading system. *Proceedings of IEEE International Conference on Data Mining*. pp. 1002 - 1007.

Mottosinho, F. J. A. P., 2010. Mining Product Opinions and Reviews on the Web. [Online]. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.7357&rep=rep1&type=pdf

O'Shea, M. and Levene, M., 2011. Mining and visualising information from RSS feeds: a case study. *International Journal of Web Information Systems*, [Online]. Available at: https://www.researchgate.net/publication/220135799_Mining_and_visualising_information_from_RSS_feeds_A_case_study/download [Accessed 23 March 2019].

Pagolu, V. S., Reddy, K. N., Panda, G. and Majhi, B., 2016. Sentiment analysis of Twitter data for predicting stock market movements. *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. Available at: https://arxiv.org/pdf/1610.09225.pdf [Accessed 15 August 2018].

Polidoro, F., Riccardo, G., Conte, R. L. and Rossetti, F., 2015. Web scraping techniques to collect data on consumer electronics and airfares for Italian HICP compilation. *Statistical Journal of the IAOS 31(2),* pp. 165-176.

Prasad, M. R., Naik, R. L. and Bapuji, V., 2013. Cloud Computing: Research Issues and Implications. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 2(2), pp. 134-140, [Online]. Available at: https://www.researchgate.net/publication/292046750_Cloud_Computing_Research_Issues_and_Implications [Accessed 23 March 2019].

Prashant, G., 2017. *Decision Trees in Machine Learning*. [Online]. Available at: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 [Accessed 20 March 2019].

Priyanshu, S. and Rizwan, K., 2018. A Review Paper on Cloud Computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 8(6), pp. 17-20, [Online]. Available at: https://www.researchgate.net/publication/326073288_A_Review_Paper_on_Cloud_Computing [Accessed 23 March 2019].

Qian, B and Rasheed, K., 2006. Stock market prediction with multiple classifiers. *Applied Intelligence,* 26, pp. 25-33.

Qian, L., Luo, Z., Du, Y. and Guo, L., 2009. Cloud Computing: An Overview. *IEEE International Conference on Cloud Computing*, pp. 626-631. Springer, Berlin, Heidelberg.

Ranco, G., Aleksovski, D., Caldarelli, G., Grčar, M., and Mozetič, I., 2015. The Effects of Twitter Sentiment on Stock Price Returns. *PLoS ONE*, 10(9), [Online] Available at: https://arxiv.org/pdf/1506.02431.pdf [Accessed 15 August 2018].

Salerno, J. J. and Boulware, D.M., 2006. Method And Apparatus For Improved Web Scraping. [Online]. Available at: PATENT

Saloni, G., n.d. *Decision Tree*. Avaialble at: https://www.geeksforgeeks.org/decision-tree/ [Accessed 21 March 2019].

Santosh, K. and Goudar, R. H., 2012. Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey. *International Journal of Future Computer and Communication*, 1(4), pp. 356-360, [Online]. Available at: http://www.ijfcc.org/papers/95-F0048.pdf [Accessed 23 March 2019].

Saurkar, A. V., Pathare, K. G. and Gode, S.A., 2018. An Overview On Web Scraping

Techniques And Tools. *International Journal on Future Revolution in Computer Science & Communication Engineering,* 4(4), pp. 363-367. Available at: http://www.ijfrcsce.org/download/browse/Volume_4/April_18_Volume_4_Issue_4/1524638955_25-04-2018.pdf [Accessed 15 March 2019].

Schumaker, R. P. and Chen, H., 2009. Textual analysis of stock market prediction using breaking financial news. *ACM Transactions on Information Systems*, 27(2), pp. 1–19.

scikit-learn, n.d. *1.17. Neural network models (supervised).* [Online]. Available at: http://scikit-learn.org/stable/modules/neural_networks_supervised.html [Accessed 13 August 2018].

scikit-learn, n.d. *1.4. Support Vector Machines*. [Online]. Available at: http://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation [Accessed 13 August 2018].

scikit-learn, n.d. *3.3. Model evaluation: quantifying the quality of predictions.* [Online]. Available at: http://scikit-learn.org/stable/modules/model_evaluation.html [Accessed 14 August 2018].

scikit-learn, n.d. *sklearn.naive_bayes.GaussianNB*. [Online]. Available at: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html [Accessed 13 August 2018].

Sidana, M, 2017. Types of classification algorithms in Machine Learning. [Online]. Available at: https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14 [Accessed 25 March 2019].

Wong, F. M. F., Liu, Z. and Chiang, M., 2015. Stock Market Prediction from WSJ:

Text Mining via Sparse Matrix Factorization. *Proceedings - IEEE International Conference on Data Mining*, pp. 430-439.

APPENDICES

Appendix 1: Python script for processing of text and numerical data, and developing of models.

```
import json
from nltk import word_tokenize
from nltk.corpus import stopwords
import pandas as pd
from nltk.stem import PorterStemmer
import datetime

factor = 2
PctChgTh = 0.05
MinTFreqTh = 3


### textual data processing
eng_stop=stopwords.words('english')
porter = PorterStemmer()

j = json.load(open('text.json','r'))

keys=j.keys()

error={k:j[k] for k in j  if 'ERROR' in j[k]['title']}
for k in error:
   del j[k]

engstop_stem=[porter.stem(t) for t in eng_stop]

def func(article):
   tokens=word_tokenize(article.lower())
   stem_tokens=[porter.stem(t) for t in tokens]
   red_tokens=[]

   for w in stem_tokens:
      if w not in engstop_stem and w.isalpha() :
         red_tokens.append(w)

   wordcnt_d=dict([w,red_tokens.count(w)] for w in set(red_tokens))
   return wordcnt_d


master_d=dict([k,func(j[k]['article'])]for k in j)
title_d=dict([k,func(j[k]['title'])] for k in j)
```

```
for k in j:
    for w in master_d[k]:
        if w in title_d[k]:
            master_d[k][w]*=factor

articlewords=[list(master_d[k]) for k in master_d]

all_words = list(set([item for sublist in articlewords for item in sublist]))

pos_words = open('positive-words.txt','r', encoding='latin').read().split('\n')
neg_words = open('negative-words.txt','r', encoding='latin').read().split('\n')

def processw (words):
    sentimental_words=[]
    stem_words=[porter.stem(w) for w in words]
    for w in stem_words:
        if w not in eng_stop and w.isalpha() :
            sentimental_words.append(w)
    return sentimental_words

pos_words=processw(pos_words)
neg_words=processw(neg_words)

words=[w for w in all_words if w in pos_words or w in neg_words]

def cleanwrd(wordlist,wordcnt_d):
    words_by_doc={}
    for w in wordlist:
        words_by_doc[w]=[]
        for k in j:
            if w in wordcnt_d[k]:
                words_by_doc[w].append(wordcnt_d[k][w])
            else :
                words_by_doc[w].append(0)
    return words_by_doc

words_by_doc=cleanwrd(words,master_d)

valid_date={k:j[k] for k in j  if not j[k]['publish_date'] == ''}
for k in valid_date:
    date=j[k]['publish_date'].split('T',1)[0]
    j[k]['publish_date']=datetime.datetime.strptime(date,'%Y-%m-
%d').strftime('%m/%d/%y')

tmatrix=pd.DataFrame(data=words_by_doc,index=list(j[k]['publish_date'] for k in j))

tmatrix.loc['TOTAL']= tmatrix.sum()

#consist of words with total freq > MinTFreqTh
```

```
tsig_matrix = tmatrix.T.loc[tmatrix.T['TOTAL'] > MinTFreqTh].T

totalsigcnt=dict(tsig_matrix.loc['TOTAL',])
desc_sigtotal_col=sorted(totalsigcnt,key=totalsigcnt.get,reverse=True)
tsig_matrix=tsig_matrix.reindex(columns=desc_sigtotal_col)
tsig_matrix.drop('TOTAL',inplace=True)

tsig_matrix.to_csv('t'+'matrix display.csv')


#target
indexprice=pd.read_csv('HistoricalPrices.csv')
indexprice.set_index(['Date'],inplace=True)
indexprice.sort_index(inplace=True)

indexprice['PctChg']=indexprice['Close'].pct_change()  #in decimal, not %
indexprice.dropna(subset=['PctChg'],inplace=True) #no effect
indexprice['PctChg']=indexprice['PctChg']*100

indexprice.loc[:,'Target'] = 0
indexprice.loc[indexprice['PctChg'] > PctChgTh, 'Target'] = 1
indexprice.loc[indexprice['PctChg'] < -PctChgTh, 'Target'] = -1



### textual data: final matrix
tfinalmatrix=tsig_matrix.join(indexprice['Target'])
tfinalmatrix.dropna(subset=['Target'],inplace=True) # 259 data
tfinalmatrix.to_csv('textual '+'matrix display.csv')



### textual data: model building
score = {}
scoreindex = [ 'Logisitc', 'LinearSVC', 'SVC',
            'GaussianNB', 'BinomialNB', 'MultinomialNB',
             'MLPC_lbfgs', 'DecisionTree']
score['t_fit'] = []
score['t_pref'] = []

from sklearn.model_selection import train_test_split as split
X=tfinalmatrix.drop(columns=['Target'])
y=tfinalmatrix.loc[:,'Target']
X_train, X_test, y_train, y_test = split(X,y, train_size=0.7,
test_size=0.3,random_state=50) #random_state=seed

from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression().fit(X_train, y_train)
print( logistic.score( X_train, y_train) )
```

```
print( logistic.score( X_test, y_test) )
prediction = logistic.predict(X_test)
score['t_fit'].append(logistic.score( X_train, y_train))
score['t_pref'].append(logistic.score( X_test, y_test))

from sklearn.svm import LinearSVC
lsvm = LinearSVC().fit(X_train,y_train)
prediction = lsvm.predict(X_test)
print( lsvm.score( X_train, y_train) )
print( lsvm.score( X_test, y_test) )
score['t_fit'].append(lsvm.score( X_train, y_train))
score['t_pref'].append(lsvm.score( X_test, y_test))

from sklearn.svm import SVC
svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X_train,y_train)
prediction = svm.predict(X_test)
print( svm.score( X_train, y_train) )
print( svm.score( X_test, y_test) )
score['t_fit'].append(svm.score( X_train, y_train))
score['t_pref'].append(svm.score( X_test, y_test))

#BernoulliNB - for binary data
#MultinomialNB - for count data
#GaussianNB - for continous data
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(X_train, y_train)
prediction = gnb.predict(X_test)
print( gnb.score( X_train, y_train) )
print( gnb.score(X_test, y_test) )
score['t_fit'].append(gnb.score( X_train, y_train))
score['t_pref'].append(gnb.score( X_test, y_test))

from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB().fit(X_train, y_train)
prediction = mnb.predict(X_test)
print( mnb.score( X_train, y_train) )
print( mnb.score(X_test, y_test) )
score['t_fit'].append(mnb.score( X_train, y_train))
score['t_pref'].append(mnb.score( X_test, y_test))

from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB().fit(X_train, y_train)
prediction = bnb.predict(X_test)
print( bnb.score( X_train, y_train) )
print( bnb.score(X_test, y_test) )
score['t_fit'].append(bnb.score( X_train, y_train))
score['t_pref'].append(bnb.score( X_test, y_test))

from sklearn.neural_network import MLPClassifier
```

```
mlp = MLPClassifier(solver='lbfgs', random_state=0)
mlp.fit(X_train,y_train)
print( mlp.score(X_train, y_train) )
print( mlp.score(X_test, y_test) )
score['t_fit'].append(mlp.score( X_train, y_train))
score['t_pref'].append(mlp.score( X_test, y_test))

from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state=0)
dtree.fit(X_train, y_train)
print( dtree.score(X_train, y_train) )
print( dtree.score(X_test, y_test) )
score['t_fit'].append(dtree.score( X_train, y_train))
score['t_pref'].append(dtree.score( X_test, y_test))



### numerical data processing
ind_prices=pd.ExcelFile("30 stock prices.xlsx")
print(ind_prices.sheet_names)
stocklist=list(ind_prices.sheet_names)
stocklist.remove('Sheet1')

nmatrix = ind_prices.parse('AAPL US Equity',skiprows=5)
nmatrix.set_index(['Dates'],inplace=True)
nmatrix=pd.DataFrame({'AAPL US Equity': nmatrix['PX_OFFICIAL_CLOSE']})

for k in stocklist:
    add = ind_prices.parse(k,skiprows=5)
    add.set_index(['Dates'],inplace=True)
    nmatrix[k]=add['PX_OFFICIAL_CLOSE']

nmatrix.drop(nmatrix.index[0],inplace=True)

newdatesindex=[]
for d in nmatrix.index.values:
    d=str(d)
    date = d.split('T',1)[0]
    date=datetime.datetime.strptime(date,'%Y-%m-%d').strftime('%m/%d/%y')
    newdatesindex.append(date)
nmatrix = nmatrix.reindex(newdatesindex)



### numerical data: final matrix
nfinalmatrix=nmatrix.join(indexprice['Target'])
nfinalmatrix.dropna(subset=['Target'],inplace=True)
nfinalmatrix.to_csv('matrix display.csv')
```

```
### numerical data: model building
score['n_fit'] = []
score['n_pref'] = []

X=nfinalmatrix.drop(columns=['Target'])
y=nfinalmatrix.loc[:,'Target']
X_train, X_test, y_train, y_test = split(X,y, train_size=0.7,
test_size=0.3,random_state=50)

logistic = LogisticRegression().fit(X_train, y_train)
prediction = logistic.predict(X_test)
print( logistic.score( X_train, y_train) )
print( logistic.score( X_test, y_test) )
score['n_fit'].append(logistic.score( X_train, y_train))
score['n_pref'].append(logistic.score( X_test, y_test))

lsvm = LinearSVC().fit(X_train,y_train)
prediction = lsvm.predict(X_test)
print( lsvm.score( X_train, y_train) )
print( lsvm.score( X_test, y_test) )
score['n_fit'].append(lsvm.score( X_train, y_train))
score['n_pref'].append(lsvm.score( X_test, y_test))

svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X_train,y_train)
prediction = svm.predict(X_test)
print( svm.score( X_train, y_train) )
print( svm.score( X_test, y_test) )
score['n_fit'].append(svm.score( X_train, y_train))
score['n_pref'].append(svm.score( X_test, y_test))

gnb = GaussianNB().fit(X_train, y_train)
prediction = gnb.predict(X_test)
print( gnb.score( X_train, y_train) )
print( gnb.score(X_test, y_test) )
score['n_fit'].append(gnb.score( X_train, y_train))
score['n_pref'].append(gnb.score( X_test, y_test))

mnb = MultinomialNB().fit(X_train, y_train)
prediction = mnb.predict(X_test)
print( mnb.score( X_train, y_train) )
print( mnb.score(X_test, y_test) )
score['n_fit'].append(mnb.score( X_train, y_train))
score['n_pref'].append(mnb.score( X_test, y_test))

bnb = BernoulliNB().fit(X_train, y_train)
prediction = bnb.predict(X_test)
print( bnb.score( X_train, y_train) )
```

```
print( bnb.score(X_test, y_test) )
score['n_fit'].append(bnb.score( X_train, y_train))
score['n_pref'].append(bnb.score( X_test, y_test))

mlp = MLPClassifier(solver='lbfgs', random_state=0)
mlp.fit(X_train,y_train)
print( mlp.score(X_train, y_train) )
print( mlp.score(X_test, y_test) )
score['n_fit'].append(mlp.score( X_train, y_train))
score['n_pref'].append(mlp.score( X_test, y_test))

dtree = DecisionTreeClassifier(random_state=0)
dtree.fit(X_train, y_train)
print( dtree.score(X_train, y_train) )
print( dtree.score(X_test, y_test) )
score['n_fit'].append(dtree.score( X_train, y_train))
score['n_pref'].append(dtree.score( X_test, y_test))



### augmented data: final matrix ###
afinalmatrix=tfinalmatrix.join(nfinalmatrix,lsuffix='_extra')
afinalmatrix.drop(columns=['Target_extra'],inplace=True)
afinalmatrix.dropna(inplace=True)   #no effect



### augmented data: model building
score['a_fit'] = []
score['a_pref'] = []

X=afinalmatrix.drop(columns=['Target'])
y=afinalmatrix.loc[:,'Target']
X_train, X_test, y_train, y_test = split(X,y, train_size=0.7,
test_size=0.3,random_state=50)

logistic = LogisticRegression().fit(X_train, y_train)
prediction = logistic.predict(X_test)
print( logistic.score( X_train, y_train) )
print( logistic.score( X_test, y_test) )
score['a_fit'].append(logistic.score( X_train, y_train))
score['a_pref'].append(logistic.score( X_test, y_test))

lsvm = LinearSVC().fit(X_train,y_train)
prediction = lsvm.predict(X_test)
print( lsvm.score( X_train, y_train) )
print( lsvm.score( X_test, y_test) )
score['a_fit'].append(lsvm.score( X_train, y_train))
```

```
score['a_pref'].append(lsvm.score( X_test, y_test))

svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X_train,y_train)
prediction = svm.predict(X_test)
print( svm.score( X_train, y_train) )
print( svm.score( X_test, y_test) )
score['a_fit'].append(svm.score( X_train, y_train))
score['a_pref'].append(svm.score( X_test, y_test))

gnb = GaussianNB().fit(X_train, y_train)
prediction = gnb.predict(X_test)
print( gnb.score( X_train, y_train) )
print( gnb.score(X_test, y_test) )
score['a_fit'].append(gnb.score( X_train, y_train))
score['a_pref'].append(gnb.score( X_test, y_test))

mnb = MultinomialNB().fit(X_train, y_train)
prediction = mnb.predict(X_test)
print( mnb.score( X_train, y_train) )
print( mnb.score(X_test, y_test) )
score['a_fit'].append(mnb.score( X_train, y_train))
score['a_pref'].append(mnb.score( X_test, y_test))

bnb = BernoulliNB().fit(X_train, y_train)
prediction = bnb.predict(X_test)
print( bnb.score( X_train, y_train) )
print( bnb.score(X_test, y_test) )
score['a_fit'].append(bnb.score( X_train, y_train))
score['a_pref'].append(bnb.score( X_test, y_test))

mlp = MLPClassifier(solver='lbfgs', random_state=0)
mlp.fit(X_train,y_train)
print( mlp.score(X_train, y_train) )
print( mlp.score(X_test, y_test) )
score['a_fit'].append(mlp.score( X_train, y_train))
score['a_pref'].append(mlp.score( X_test, y_test))

dtree = DecisionTreeClassifier(random_state=0)
dtree.fit(X_train, y_train)
print( dtree.score(X_train, y_train) )
print( dtree.score(X_test, y_test) )
score['a_fit'].append(dtree.score( X_train, y_train))
score['a_pref'].append(dtree.score( X_test, y_test))



scorematrix = pd.DataFrame(score, scoreindex)
scorematrix.to_csv(str(factor)+"factor_"+str(MinTFreqTh)+"mtfreq_"+"model
score.csv")
```

Appendix 2: Python script for data collection using RSS (running in cloud computer).

```
import feedparser
import time
import json

data_wd = 'DATA_'
i=0

try:
   while True:
      i=i+1
      with open('c_rss_list.txt','r') as rss_f:
         rss_l = rss_f.readlines()

      rss_list = [ rss.partition('#')[0].strip() for rss in rss_l ]
      rss_list = list(filter(None, rss_list)) #remove empty string

      print(i,'rss_list')

      rdata_d={}
      for rss in rss_list:
         feed=feedparser.parse(rss)
         if not feed['items'] == []:
            rdata_d[rss]=feed['items']

      print(i,'rdata_d',len(rdata_d))

      data_d={}
      for k in rdata_d.keys():
         print(k)
         for d in range (len(rdata_d[k])):
            try:
               link = rdata_d[k][d]['link']

data_d[link]={'title':rdata_d[k][d]['title'],'summary':rdata_d[k][d]['summary'],'publishe
d':rdata_d[k][d]['published']}
               print(d,link)
            except:
               print(d,"-- Sth wrong")

      print(i,'data_d')
      time.sleep(60)

      with open(data_wd + 'rss.json' ,'a') as f:
         json.dump(data_d, f)
         f.write('\n')
```

```
        print(i,"pass")
        time.sleep(600)

except KeyboardInterrupt:
    print ('Terminated')
except:
    print ('Sth wrong')
```

Appendix 3: Python script for data collection using web scraping with request (running in cloud computer).

```
import requests
from bs4 import BeautifulSoup
import json
import time

#only for one website

webpg_wd = 'WEBPAGE_'
data_wd = 'DATA_'

url = 'https://www.fool.com'
i=0

try:
    while True:
        i=i+1
        r = requests.get(url)
        r.text

        soup = BeautifulSoup(r.text,'html.parser')
        with open(webpg_wd + 'Fool.txt', 'w',encoding='utf-8') as f:
            f.write(soup.prettify())

        samples = soup.find(class_="hp-trending-articles-list")
        samples = samples.find_all('a', href=True)

        print(i,"ori_soup", len(samples))

        data_d = {}
        for l in samples:
            link = 'https://www.fool.com' + l.attrs['href']
            data_d[link] = {}
            title = l.string.strip()
            data_d[link]['title'] = title

            lsoup = BeautifulSoup(requests.get(link).text,'html.parser')
            filename = webpg_wd + "Fool-" + title.replace(' ','_') + '.html'
```

```
            with open(filename, 'w',encoding='utf-8') as f:
                f.write(soup.prettify())

            content = lsoup.find(class_="article-content")
            txt = content.get_text(strip=True,separator=' ')

            article = txt.replace('\xa0', ' ')
            data_d[link]['content'] = article

            print(link)
            time.sleep(180)

        print(i,'data_d')

        with open(data_wd + 'Fool(soup).json' , 'a') as f:
            json.dump(data_d, f, sort_keys=True)
            f.write('\n')

        print(i,"pass")
        time.sleep(1800)   #1800

except KeyboardInterrupt:
    print ('Terminated')
except:
    print ('Sth wong')
```

Appendix 4: Python script for data collection using web scraping with selenium (running in local computer).

```
from selenium import webdriver
from bs4 import BeautifulSoup

import json
import time


webpg_wd = '/Users/home/Documents/FYP/WEBPAGE/'
data_wd = '/Users/home/Documents/FYP/DATA/'

driver = webdriver.Safari()

url = "https://www.bloomberg.com/markets"
c_names = ["single-story-module__headline-link",
           "single-story-module__related-story-link",
           "story-list-story__info__headline-link"]

output = open(data_wd+'OUTPUT_sel.txt','w')
i=0
```

```
try:
    while True:
        i=i+1
        driver.get(url)
        with open(webpg_wd + 'Bloomberg-Markets.html', 'w') as f:
            f.write(driver.page_source)
        soup = BeautifulSoup(driver.page_source, 'lxml')

        output.write(str(i))
        output.write('ori_soup 3\n')
        print(i,'ori_soup 3')

        data_d = {}
        try:
            for c in c_names:
                tags = soup.findAll('a', {'href':True, 'class':c})

                output.write(str(i))
                output.write(str(c))
                output.write(str(len(tags)))
                output.write('\n')
                print(i,c, len(tags))

                for t in tags:

                    link = "https://www.bloomberg.com" + t['href']
                    data_d[link] = {}
                    title = t.get_text(strip=True,separator=' ')
                    data_d[link]['title'] = title

                    driver.get(link)
                    filename = webpg_wd + 'Bloom-' + title.replace(' ','_') + '.html'
                    with open(filename, 'w') as f:
                        f.write(driver.page_source)

                    lsoup = BeautifulSoup(driver.page_source, 'lxml')
                    content = lsoup.find_all({'p'})
                    txt = [s.get_text(strip=True,separator=' ') for s in content]
                    fulltxt = ' '.join(txt)

                    data_d[link]['content'] = fulltxt

                    output.write(link)
                    output.write('\n')
                    print(link)
                    time.sleep(60) #300

            output.write(str(i))
            output.write('data_d\n')
```

```
            print(i,'data_d')

            with open(data_wd + 'Bloom(sel).json', 'a') as f:
                json.dump(data_d, f, sort_keys=True)
                f.write('\n')

            output.write(str(i))
            output.write("pass\n")
            print(i,"pass")
            time.sleep(300)    #1800

        except KeyboardInterrupt:
            with open(data_wd + 'Bloom(sel).json', 'a') as f:
                json.dump(data_d, f, sort_keys=True)
                f.write('\n')
            output.write(str(i))
            output.write("fail\n")
            print(i,"fail")
            time.sleep(300)    #1800
        except:
            with open(data_wd + 'Bloom(sel).json', 'a') as f:
                json.dump(data_d, f, sort_keys=True)
                f.write('\n')
            output.write(str(i))
            output.write("fail\n")
            print(i,"fail")
            time.sleep(300)    #1800

except KeyboardInterrupt:
    output.write ('Terminated')
    print ('Terminated')
except:
    output.write ('Sth wong')
    print('Sth wong')

output.close()
```