

**ENSEMBLE ODEL FOR TARGET SENTIMENT
ANALYSIS BASED ON NAÏVE BAYES AND SUPPORT
VECTOR MACHINE FOR PRODUCT REVIEW
CLASSIFICATION**

RHODA VIVIANE ACHIENG OGUTU

**MASTER OF SCIENCE
(Information Technology)**

**JOMO KENYATTA UNIVERSITY
OF
AGRICULTURE AND TECHNOLOGY**

2023

**Ensemble Model for Target Sentiment Analysis Based On
Naïve Bayes and Support Vector Machine for Product Review
Classification**

Rhoda Viviane Achieng Ogutu

**A Thesis Submitted in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Information Technology of the
Jomo Kenyatta University of Agriculture and Technology**

2023

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University

SignatureDate.....

Rhoda Viviane Achieng Ogutu

This thesis has been submitted for examination with our approval as University Supervisors.

SignatureDate.....

Dr. Richard Rimiru, PhD
JKUAT, Kenya

SignatureDate.....

Dr. Calvins Otieno, PhD
Maseno University, Kenya

DEDICATION

I dedicate this work to my entire family, especially my two sons, *Nathaniel* and *William*.

ACKNOWLEDGEMENT

This thesis work has been the fruit of my intense perseverance, hard work and dedication to my research work and experiments. The work would have not been complete without the various inputs from my loved ones, supervisors and mentors.

I would first want to thank the Almighty God for inspirations, provisions and favor. Secondly, I thank my ‘cheer leader’ mum, Nya Chula, who encouraged me along the way and supported me financially. Lastly, I also have to thank my co-authors and supervisors, Dr. Richard Rimiru and Dr. Calvins Otieno whose input and expertise resulted to an exceptional piece of work. I am highly grateful.

TABLE OF CONTENTS

DECLARATION.....	ii
DEDICATION.....	iii
ACKNOWLEDGEMENT.....	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES.....	ix
LIST OF FIGURES.....	xi
LIST OF APPENDICES.....	xvi
ABBREVIATIONS AND ACRONYMS.....	xvii
ABSTRACT.....	xviii
CHAPTER ONE.....	1
INTRODUCTION.....	1
1.1 Research Background.....	1
1.2 Problem Statement.....	4
1.3 Objectives.....	5
1.3.1 General Objective.....	5
1.3.2 Specific Objectives.....	5
1.4 Research Questions.....	6
1.5 Justification.....	6
1.6 Scope of the Study.....	7
1.7 Thesis Organization.....	8
CHAPTER TWO.....	10
LITERATURE REVIEW.....	10
2.1 General Introduction.....	10

2.2 Text Mining.....	15
2.2.1 Opinionated Information	16
2.2.2 Social Networking	17
2.2.3 Microblogging.	18
2.3 Sentiment Analysis.....	19
2.3.1 Sentiment Analysis Process in R-Studio	20
2.3.2 Approaches Used for Informal or Mixed Linguistic Languages’ Sentiment Analysis.....	22
2.4 Automation of Sentiment Analysis	22
2.4.1 Machine Learning.....	25
2.4.2 Supervised Machine Learning.	25
2.4.3 Unsupervised Machine Learning.....	31
2.4.4 Bag-of-Word and n-gram Sentiment Modelling.....	32
2.5 Sentiment Analysis Classifiers.....	33
2.5.1 Naïve Bayes Classifier.....	34
2.5.2 Support Vector Machine Classifier.	41
2.5.3 Ensemble Learning.	54
2.6 Previous Works and Approaches in Machine Learning to Sentiment detection in Social Media Monitoring - Sentiment Analysis.	62
2.7 Related Work to Unstructured Text Sentiment Analysis on Social Media.....	65
CHAPTER THREE	68
RESEARCH METHODOLOGY	68
3.1 Introduction	68
3.2 Sentiment Analysis Process in R-Studio.....	69
3.2.1 Data Sources and Collection.....	69

3.2.2 Setting Working Directory, Install and Load Libraries (Packages) In R	70
3.2.3 Authorize Access for Twitter API	71
3.2.4 Tweet Harvesting and Data Mining	71
3.2.5 Creating a Data Frame	72
3.2.6 Pre-processing and Data Preparation	73
3.2.7 Creating a Bag-of-Word Corpus	77
3.2.8 Build a Document Term Matrix	80
3.2.9 Analysis for the Tweets	84
CHAPTER FOUR	93
EXPERIMENTS AND RESULTS	93
4.1 Introduction	93
4.2 Models Creation in R	93
4.2.1 Naïve Bayes Classifier Model Creation in R	94
4.2.2 Support Vector Machine Classifier Model Creation in R	113
4.2.3 The Ensemble Model Creation in R	129
4.3 Experiment Results and Discussions	147
CHAPTER FIVE	155
SUMMARY, CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK	155
5.1 Introduction	155
5.2 Summary	155
5.3 Achievements	156
5.4 Conclusion	158
5.5 Recommendations	159
5.6 Limitations of the Research Study	159

5.7 Knowledge Contribution to the field of study.....	160
5.8 Future Work/ Further Gaps in Related Research	161
REFERENCES	162
APPENDICES	185

LIST OF TABLES

Table 2.1: Computations for conditional probabilities for both positive and negative class	41
Table 3.1: Document Term Matrix (DTM) for the sentences	79
Table 4.1: Summary of the data frame <i>classifiedwords_df</i>	99
Table 4.2: Confusion Matrix Table for Binary Classification and the Array Representation Used in the Study	104
Table 4.3: Performance Metrics for Model Evaluation.....	105
Table 4.4: Naïve Bayes Confusion Matrix for 2000 Product Review Analysis	106
Table 4.5: Summary of Confusion Matrix for 2000 Product Review Analysis.....	108
Table 4.6: Kappa Statistic Interpretation	109
Table 4.7: SVM Confusion Matrix for 2000 Product Review Analysis	126
Table 4.8: Comparative Results for Classifiers Accuracy	130
Table 4.9: Comparative Results for Classifiers Error Rates (<i>err</i>).....	131
Table 4.10: <i>nbmodel</i> Confusion Matrix for the Ensemble Model Computation.....	136
Table 4.11: The Stacked Ensemble Model Confusion Matrix Table.....	147
Table 4.12: Comparative Results for Classifiers Performance in terms of Accuracy – with Caret Package.....	148
Table 4.13: Comparative Results for Classifiers Performance in terms of Error Rate – with Caret Package.....	149

Table 4.14: Comparative Results for Classifiers Performance in terms of Precision (p)	150
Table 4.15: Comparative Results for Classifiers Performance in terms of Recall (r)	150
Table 4.16: Comparative Results for Classifiers Performance in terms of F-Measure (FM)	151
Table 4.17: Comparative Results for Feature Vectors	151
Table 5.1: Publication Achievements	158

LIST OF FIGURES

Figure 2.1: Machine Learning.....	24
Figure 2.2: Steps in Text Classification	28
Figure 2.3: A 2-Dimensional Classification	30
Figure 2.4: The Naïve Bayes algorithm, using Laplace Smoothing	39
Figure 2.5: Miniature training and test documents simplified from OnePlus mobile product reviews	40
Figure 2.6: Linearly Separable Data	43
Figure 2.7: Multiple Separation Boundaries (Hyperplanes)	44
Figure 2.8: Optimal Hyperplane in a Supervised Linearly Separable Dataset	45
Figure 2.9: Soft margin classifier of a supervised linearly separable dataset	46
Figure 2.10: Non-Linearly Separable Data	47
Figure 2.11: Geometric Margin of a point (r) and a decision boundary width ρ	49
Figure 2.12: Stages of Stacking Algorithm Classification Process	55
Figure 3.1: The Processes of Supervised Machine Learning.....	68
Figure 3.2: A representation of the mined tweets	70
Figure 3.3: The 16-feature variables raw dataset	72
Figure 3.4: The raw data feature variables in a <i>csv</i> file format data frame.....	73
Figure 3.5: The stripped text data in a <i>csv</i> file format	74
Figure 3.6: Partially clean text data	77

Figure 3.7: Word Frequencies.....	83
Figure 3.9: A flowchart of a typical text analysis that uses tidy data principles with <i>tidytext</i> package	84
Figure 3.10: The <i>AFINN Lexicon</i>	86
Figure 3.11: The <i>bing Lexicon</i>	86
Figure 3.12: The <i>nrc Lexicon</i>	87
Figure 3.13: A Sample of Sentiment Emotion Types with Their Scores.....	88
Figure 3.14: Comparison of Number of words by Sentiment Lexicon.....	89
Figure 3.15: Words tokenized to n-grams for Analysis	90
Figure 3.16: Head return of matching rows and columns from <i>dtm_tidy</i> and <i>bing lexicon</i>	91
Figure 3.17: Tail return of matching rows and columns from <i>dtm_tidy</i> and <i>bing lexicon</i>	92
Figure 4.1: The <i>classified_sentiments_df.csv</i> data frame	96
Figure 4.2: Randomized Observations Corresponding to the Sample Index	97
Figure 4.3: Data frame with character data types	98
Figure 4.4: A Factorized data frame with two Factor levels ' <i>positive</i> ', ' <i>negative</i> ' ..	98
Figure 4.5: Bar plot of Sentiment grouping for 2000 Product Reviews from the <i>classifiedwords_df</i> data frame.....	99
Figure 4.6: A Portion of the Naïve Bayes Classifier output for 2000 Product Reviews	101

Figure 4.7: Summary of the trained model for 2000 Product Reviews	102
Figure 4.8: A Sample of Naïve Bayes Prediction Outcome for the Test Document for 2000 Product Reviews.....	103
Figure 4.9: pred_nb model Probability Measures for the first 20 Individual Words	103
Figure 4.11: A Sample of Training 2000 Product Reviews with Laplace Smoothing	111
Figure 4.12: Confusion Matrix Results with Laplace Smoothing	112
Figure 4.13: Given a particular data point (y and x) a classification <i>a</i> or <i>b</i> is made	113
Figure 4.14: One-Hot Encoding for 2000 Product Reviews from the <i>classifiedwords_df</i> data frame.....	116
Figure 4.15: A Sample Output for the Data Frame <i>classifiedwords_df2</i>	117
Figure 4.16: A One-Hot Encoding Output.....	117
Figure 4.17: Creation of <i>classifiedwords_df3</i> data frame.....	119
Figure 4.18: <i>dataset_svm</i> data frame	119
Figure 4.19: Sample Observations for Encoded <i>dataset_svm</i> data frame.....	120
Figure 4.20: Sample Observations for Encoded <i>dataset_svm1</i> data frame.....	120
Figure 4.21: <i>dataset_svm1</i> data frame Structure.....	121
Figure 4.22: Factorised <i>dataset_svm1</i> data frame Structure	121
Figure 4.23: Partitioning of the Data Frame <i>dataset_svm1</i> for Analysis with Support Vector Machine	122

Figure 4.24: Training SVM model at a value of 0.1 for the Cost Parameter.....	123
Figure 4.25: Training SVM model at a value of 1 for the Cost Parameter.....	123
Figure 4.26: Training SVM model at a value of 10 for the Cost Parameter.....	123
Figure 4.27: A ten-fold Cross Validation for the ‘best’ SVM Model.....	124
Figure 4.28: The ‘best’ Model obtained.....	124
Figure 4.29: A trained SVM Model with the Largest Cost value.....	125
Figure 4.30: A sample of the Predicted SVM Model	126
Figure 4.31: Ensemble Model Tuning Algorithm (Kuhn, 2019).....	130
Figure 4.32: Stacking Algorithm	132
Figure 4.33: Defining the Training control and a training output for the first base classifier <i>nbmodel</i>	134
Figure 4.34: Summary of the trained <i>nbmodel</i>	135
Figure 4.35: A sample of the Predicted <i>nbmodel</i> base Model	135
Figure 4.36: A 2000 Product Review Analysis Sample of the <i>nbmodel</i> Performance for the Ensemble Model Computation	136
Figure 4.37: Renaming the Levels of the Response Variable for the Train Set	137
Figure 4.38: Renaming the Levels of the Response Variable for the Test Set	138
Figure 4.39: Defining the Training control for the SVM Model	138
Figure 4.40: The training output for the Second base classifier SVM Model	139
Figure 4.41: A Training plot for the SVM Model.....	139

Figure 4.42: A sample of the Predicted SVM base Model	140
Figure 4.43: A 2000 Product Review Analysis Sample of the SVM Model Performance for the Ensemble Model Computation.....	140
Figure 4.44: Defining the Training control for the Ensemble Model	141
Figure 4.45: The Trained Ensemble Output for 2000 Reviews	142
Figure 4.46: The Ensemble Summary Prediction Performance Results for 2000 Reviews	142
Figure 4.47: The Resampling Results in terms of Accuracy	143
Figure 4.48: Comparison of the Base Models.....	143
Figure 4.49: The Correlation between the Base Classifiers Results	144
Figure 4.50: Defining the Stacked Ensemble Training control and the Training output	145
Figure 4.51: Summary of the Resulting Stacked Ensemble Model	146
Figure 4.52: The Confusion Matrix Result for the Stacked Ensemble Model.....	147
Figure 4.53: Resulting Stacked Ensemble Model Performance	148

LIST OF APPENDICES

Appendix I: Data Mining and Sentiment Analysis Experiment Sheet	185
--	-----

ABBREVIATIONS AND ACRONYMS

SMS	Short Message Service
App	Computer Application
SVM	Support Vector Machine
NB	Naïve Bayes
CPU	Central Processing Unit
API	Application Programming Interface
CSV	Comma-Separated Values
RT	Retweets
HTML	Hyper Text Markup Language
TXT	Text Files
PDF	Portable Document Format
SA	Sentiment Analysis
DTM	Document Term Matrix

ABSTRACT

Sentiment analysis has demonstrated that automation and computational recognition of sentiments is possible and evolving, due to factors such as, emergence of new technological trends and the continued dynamic state the human language. Sentiment analysis is therefore an Information extraction task that aims at obtaining private sentiments that can either be expressed as ‘positive’ or ‘negative’, toward a specific object or subject. However, social media platforms are marred with unstructured texts that make extraction and parsing of relevant information a problem for most systems and models. This can pose as a challenge to companies, individuals or organizations seeking to make specific strategic decisions based on the available data. To overcome such inefficiencies, on the first phase of experimentation of this study, the research implemented the use of two classifier models, Naïve Bayes and Support Vector Machine, based on feature selection and extraction of sentiments from Twitter product reviews. This was with the aim of evaluating performance of the classifiers on a ‘positive’ and ‘negative’ sentiment classification of product reviews. The classifiers are commonly used as benchmarks against which state-of-the-art (SOTA) approaches and techniques can be compared. The second phase of the experiments involved implementation of an ensemble model of the two classifiers, where the two supervised classifiers together with the ensemble model were compared and evaluated, based on the models’ accuracy measures, precision and robustness. Comparison of the two classifiers were concluded, and a competitive performance between Naïve Bayes and SVM was recorded. In addition, initial experiments in terms of accuracy and error rate measurement with Naive Bayes and SVM classifiers, indicated Naive Bayes classifier to be better in performance with progressive increase in the total number of documents to be analyzed. SVM classifier on the other hand, equally demonstrated good performance on the final phase of Ensemble model experimentations. The results indicated that both SVM and Naïve Bayes were good classifiers for text classification, where, relatively good performances were achieved on accuracy, with the best performing classifiers attaining 99.40%. As there existed significant amount of errors for the classifiers, SVM and the Stacked Ensemble model performed at 0.60% on error rate. While based on the general performance outcome of the classifiers, Naïve Bayes could not be arguably regarded as an unstable classifier. In the final phase, the Stacked Ensemble model additionally demonstrated a good ability to cope with errors, resulting to the development of a robust model.

CHAPTER ONE

INTRODUCTION

1.1 Research Background

Vosoughi, Zhou and Roy (2016) argue that the rise and popularity of the need to mine unstructured data, and the use of social media platforms or microblogging, especially Twitter, has made Sentiment analysis of tweets an important area of research. At the same time, it has given web users a venue for expressing and sharing their thoughts, opinions or sentiments on all kind of topics and events. As expressed by Alayba, Palade, England and Iqbal (2017) and Ahmad, Aftab, Muhammad and Ahmad (2017), Twitter has millions of users worldwide that constantly tweet, making it a gold mine for communities, organizations as well as individuals to monitor their reputation, monitor how people feel over time about their brands – by extracting and analyzing the tweet sentiments posted by the public about them, their market or competitors. These monitoring processes can be referred to as Social Listening (Ducange, Fazzolari, Petrocchi, & Vecchio, 2019). In addition, popularity of online shopping has also increased. As such, reviews of almost any product or business exist and are monitored by the respective stakeholders. One way this is being accomplished is through Sentiment Analysis (Ahmad et al., 2017; Sarker, 2021).

According to Yang, Zhang, Yu, Yu and Zeghlache (2014) these online social interactions can be used to reveal individuals' or groups of individuals' behavior, or a community dynamics, to better understand the public's perception, by mining the digital traces left by users while interacting with cyber-physical space, such as microblogs, for profitable reasons. This has led to the emergence of research on Social and Community Intelligence that present opportunities to compile these digital footprints into a comprehensive picture of individuals' daily life facets, transform the understanding of our needs, organizations and societies; while also enable innovative improvement on products, public safety, resource management and environmental monitoring.

Such information submitted to the online services are a form of data sources that can be used in Sentiment analysis. Models in Sentiment analysis over twitter data and other microblogs faces several new challenges due to the short length and irregular structure of textual data – also commonly known as high dimensions. These challenges may include informal or colloquial content, misspellings errors, use of various languages, voluminous data, among others (Chandni, Chandra, Gupta, & Pahade, 2015; Shirbhate & Deshmukh, 2016; Awachate & Kshirsagar, 2016). As such, a problem of focusing on the most relevant information from the voluminously complex data, during Sentiment Analysis, may arise. In addition, relevant feature extraction is significant for Sentiment classification as the opinionated texts may have high dimensions, which can affect classifier performance (Tripathy, Anand, & Rath, 2017; Zheng, Wang, & Gao, 2018).

Studies in the field of Sentiment Analysis on microblogging, such as research works done by Tripathy et al. (2017) and Sarker (2021) indicate that in today’s digital world, standard Machine Learning techniques outperform human produced techniques which may include manually choosing good indicator words for positive and negative sentiments in textual data. Additionally, combination of models, for instance, where a research study in Tripathy et al. (2017) studied microblogging reviews and built Unigram. Feature-based models for classifying reviews, with models built for two classification tasks, where the binary task (positive and negative) found out that a model of combining unigrams with selected features, outperformed other combinations of models and was the best performing system for the positive versus negative task.

Yadav, Kudale, Gupta, Rao and Shitole (2020) extracted features inform of unigrams and bigrams while at the same time experimented with various classifiers such as Naïve Bayes, Decision Tree, Random Forest, XGBoost and SVM. The researchers concluded that using sentiment features rather than conventional text classification models gave high accuracy. This conclusion was as a result of the various inherent draw backs of the individual classifiers. For instance, while experimenting with Naïve Bayes and Decision Tree, it was found that it was easier to confirm feature presence, as opposed to identifying the

frequency of features, because Naïve-Bayes is essentially built to work better on integer features rather than floats.

From the above researches, it was hypothetical that experimentation of Naïve Bayes and SVM could lead to good results since according to Jurafsky and Martin (2017) and as demonstrated in Chapter 2.5 in this study, subject category classification is the task for which Naïve Bayes algorithm was invented. Meaning that the algorithm's goal is to take a single observation, extract some useful features, and classify the observation into one of a set of discrete classes.

While Naïve Bayes and SVM are commonly used as benchmarks against which state-of-the-art (SOTA) approaches are compared, other techniques of classifying text such as the rule-based classifiers, prove to be fragile, as situations or data change overtime. At the same time, humans aren't necessarily good at coming up with rules for all tasks (Jurafsky & Martin, 2017). In addition, Decision Tree classifiers are relatively expensive as complexity and time taken to train the model is more (Dhiraj, 2019). However, classifiers like Naïve Bayes and SVM are not computationally costly, but neural networks and attention models have shown that they are computationally costly (Wankhade, Rao, & Kulkarni, 2022).

SVM is a non-probabilistic binary linear classification algorithm that has the ability to linearly separate classes by a large margin and has been proven to be highly effective in traditional text categorization and opinion mining, performing better than other machine learning techniques (Osisanwo, et al., 2017; Balahur & Perea-Ortega, 2015). Sentiment Classification using Naïve Bayes and SVM classifiers has been commonly used as benchmarks against which state-of-the-art or newly proposed approaches can be compared (Wankhade et al., 2022).

With this background in mind, the aim of this research was to experiment and evaluate the performance of Naïve Bayes and Support Vector Machine (SVM) classifiers, with an intension of integrating the two classifiers, and creating an ensemble model for accurate

classification purposes. This was aimed at achieving and evaluating a better predictive model, as the research problem originates as a combination of a Machine Learning and Information Retrieval challenge. By combining these fields, a problem of extraction of relevant Features in Sentiment analysis existed.

1.2 Problem Statement

Opinionated texts in Sentiment Analysis are dynamic, shortened, have spelling errors and generally unstructured, hence, focusing on the most relevant information (features) becomes daunting. In addition, Sentiment analysis involves processing of voluminous numbers of data points, which results to high computational cost (Wankhade et al., 2022). These factors lead to classification inefficiencies.

Alayba et al. (2017) in their research study established that their model of combining unigrams with selected features was the best performing system for positive versus negative task in Sentiment Analysis. However, as much as Naïve Bayes classifier was regarded as a good classifier for text classification (Jurafsky & Martin, 2017) while both Naïve Bayes and SVM were frequently used as base classifiers or benchmarks against which state-of-the-art (SOTA) approaches were evaluated, most studies in Machine Learning such as the research work done by Tripathy et al. (2017) and Kusumawati, D'arofah, & Pramana (2019) demonstrate that SVM performs better than most Machine Learning algorithms. Tripathy et al. (2017) and Wankhade et al. (2022) also pointed out that Hybrid Machine Learning approaches to sentiment classification of reviews produces better accuracies in comparison to the baseline results, and are widely utilized technique in sentiment categorization.

As such, these outcomes intuitively guided our research to therefore study and experiment on the impact of Naïve Bayes and SVM as Machine learning sentiment classifiers on classification performance in terms of accuracy, precision and robustness; while at the same time experiment on combining unigrams with selected features, and finally experiment on creating an ensemble of the two classifiers.

Our study therefore evaluated the performance of Naïve Bayes and Support Vector Machine classifiers as single models, and experimented on integrating the two classifiers so as to create an ensemble model. The assumption was that the experiments would lead us to achieving a better predictive performance. This was based on the hypothesis that ensemble learning helps improve machine learning results by combining several models' benefits (Araque, Corcuera-Platas, Sánchez-Rada, & Iglesias, 2017). SVM classifier has been widely used for Sentiment analysis and is also suitable for high-dimensional input space (Tellez, et al., 2017), while Naïve Bayes is a simple and intuitive probabilistic classifier (Devika, Sunitha, & Ganesh, 2016). A proposed combination of the two classifiers was expected to improve the classification process for product reviews (Araque et al., 2017).

1.3 Objectives

1.3.1 General Objective

To develop an ensemble model for Target Sentiment classification of product reviews using Naïve Bayes and Support Vector classifiers.

1.3.2 Specific Objectives

- 1 To identify a basic workflow for conducting sentiment analysis for product reviews from Twitter data.
- 2 To assess and validate the performance of Naïve Bayes and Support Vector Machine as supervised Machine Learning algorithms in sentiment classification.
- 3 To experiment with the proposed algorithms singularly and finally based on the resulting outcome, determine a suitable approach for creating an ensemble model of the two classifiers.
- 4 To develop and validate the performance of the proposed ensemble model, based on Naïve Bayes and Support Vector Machine, to be used in sentiment classification of products reviews.

1.4 Research Questions

- 1 What is the basic workflow for conducting sentiment analysis for product reviews from Twitter data?
- 2 What is the performance of Naïve Bayes and Support Vector Machine as supervised Machine learning algorithms in sentiment classification?
- 3 Based on the results of the experiments of the two proposed algorithms, what is the suitable approach for creating an ensemble model of the two classifiers?
- 4 What is the performance of the proposed ensemble model, based on Naïve Bayes and Support Vector Machine, as used in Sentiment classification of products reviews?

1.5 Justification

Automated softwares have various major impacts on Sentiment Analysis; improved processing speed, is one impact that many users can attest to. However, questions that relate to conclusiveness, accuracy or the general performance of systems may arise. For instance, challenges for automation in Sentiment analysis may include analysis of colloquial or unstructured data, and selection of the most relevant features to drive a learning process (Dashtipour, et al., 2016; Bagheri, Saraee, & de Jong, 2013). The unstructured data or sentiments may include non-standardized languages such as slang. A sentiment's contextual meaning in one informal language setting, may be completely different in another, while automated analysis may also be at a greater risk of assigning an incorrect sentiment or wrongly classifying a sentiment. This is because it may not take into account semantic orientation or context of a sentiment, or the consideration of prior posts or sentiments of an author (Vosoughi et al., 2016). Identifying context manually in sentences or texts has proven to be easier for human readers, while also allowing readers to correctly classify or assign sentiments as they read. However, it has at the same time proven to be complex in automation of large scale sentiment analysis (Dashtipour, et al., 2016).

Social media is dynamic in nature (Kapoor, et al., 2018) as it is characterized by constantly changing languages that seem to appear almost on a daily basis (Dashtipour, et al., 2016), this makes it almost impossible for automation to keep up (Hira & Gillies, 2015). Yet, data from analysis such as Social Listening provides a strong base for product development, market research, customer insights and consumer profiling (Yang et al., 2014). These factors encourages more research in the field of Sentiment analysis.

1.6 Scope of the Study

Computational analysis for this research was carried out using Twitter and R-Studio statistical software. Twitter is an online news and Social networking platform that allows users to post and interact by means of messages, popularly referred to as '*tweets*'. It is also a widely used online 'word of mouth' free microblogging service, which is known to be a useful social networking platform for trending analysis in real time, given the amount of data and its popularity in different countries (Vilares, Alonso, & Gómez-Rodríguez, 2017). Access to twitter is through its website interface, SMS or mobile device Apps. Users are also enabled to upload video contents. These factors have made twitter an important tool for studying the behavior, attitude of people and generally, it has been used continually as a social listening tool. Ultimately, it is important to classify these topics or information into general categories with high accuracy for better information retrieval, as in the case of Sentiment Analysis (Soong, Jalil, Ayyasamy, & Akbar, 2019).

One of the greatest appeal about Twitter is in its accessibility. It is easy to use for both sharing information and collecting it. Most of the trending topics on Twitter are global, for instance, '#smartphone' for smartphone features and accessories' reviews, and '#electronics' for various electronic products reviews. These characteristics make twitter a good place to collect real time and latest data to analyze and do any kind of research for real life situations (Mehta, Pandya, & Kotecha, 2021). This research was proposed to use datasets that were generated by mining tweets related to product reviews. Authors of these tweets were expected to write about the products they use, their experiences, and most importantly share their reviews about their ecommerce activities.

Additionally, for computational text analysis and data mining, R-Studio statistical software (R Core Team, 2022) was also used as a computational environment to carry out the sentiment analysis. R-Studio is a free, open-source, cross-platform programming environment, and according to Welbers, Atteveldt and Benoit (2017) in contrast to most programming languages, R-Studio was specifically designed for statistical analysis, which makes it highly suitable for Data science applications such as Sentiment Analysis.

1.7 Thesis Organization

The research thesis is basically divided into five chapters. Chapter 1 is an introductory chapter in which we have discussions about Sentiment analysis background. This is then followed by a description of the problem statement together with the research study objectives and questions. The chapter is then finally concluded by the discussion on the research justification and scope of the study.

Chapter 2 begins with a general introduction to Sentiment Analysis and Machine Learning. A detailed description is then given on the need and rise of Sentiment Analysis over the periods, due to the emergence of technologies such as Social Networking. This is then followed by a discussion and analysis of various text and document modeling approaches that have been used in the field of Sentiment Analysis, Text Mining and n-gram document modeling. An analysis of Naïve Bayes and SVM Machine learning algorithms and their applications in text categorization is then discussed.

Chapter 3 presents the research study's methodology. Scientific experiments were used in this research study. The chapter therefore starts by describing the sentiment analysis process in R-Studio environment, data mining procedures and data preparation. Finally, it describes how the Lexicon based sentiment classification process was carried out.

Chapter 4 is dedicated to giving a detailed explanation on the experiments, results and discussions. This starts with an outline of the data preparation techniques and finally models creation. The classifiers are first modelled singularly before an ensemble model is

created. Finally, Chapter 5 covers all the work done in summary, achievements, conclusions, recommendations, limitations of the research study, knowledge contribution and future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 General Introduction

In today's digital economies, individuals and companies are exchanging business ideas, thoughts through online web forums, blogs and social media platforms. A major chunk of these information shared online are about reviews and opinions on various aspects of human interactions. It is evident that online users give such reviews and sentimental opinions on products, brands or services provided. According to Bose, Dey, Roy and Sarddar (2020) analyzing reviews on products not only improve product quality, but also influences purchase decisions of the consumers. Thus, product review analysis is a widely accepted technological trend in Sentiment analysis.

As indicated by Soong et al. (2019) and Gao, Feng, Song and Wu (2019) in the field of Sentiment Analysis, an increased attention is now focused on analysis of social media content especially, Twitter. This may facilitate the understanding of social aspects and measure confidence level of products, policies, or the perceived image of a company. These social media contents are often in form of informal messages that are short and textual in nature. As such, these messages bring in new challenges to Sentiment Analysis. They are limited in length, tend to have many misspellings errors, shortened form of words, over capitalization, an over use of numbers or the use of non-standard expressions such as 'gr8' instead of 'great' (Lo, Cambria, Chiong, & Cornforth, 2017). They also have special markers such as hashtags and other characters (Kham, 2019; Gollapalli & Ng, 2021; Awachate & Kshirsagar, 2016). These challenges can be referred to as high dimensions in social media textual data (Gaikwad, Chaugule, & Patil, 2014). According to Tellez et al. (2017) along with these challenges, a practical sentiment classifier should be able to handle efficiently, large workloads.

For this reason, in the field of Artificial Intelligence, Machine Learning approaches have been widely applied for the automation of Sentiment Analysis to provide computers with the ability to handle large workloads, and also the ability to learn without being explicitly programmed, while at the same time improve efficiency for classifiers. With emphasis on text sentiment analysis, researches are mostly narrowed down to carrying out studies on feature selection or extraction, and analysis of classifiers used in models. This has been evidenced in the reports of works done by various researchers as reviewed in the following literature.

According to Yadollahi, Shahraki and Zaiane (2017) Sentiment analysis is an Information Extraction task that aims to obtain a writer's feelings expressed in positive or negative comments by analyzing a large number of documents. It is therefore the computational technique for extracting, classifying, understanding and determining opinions expressed in various contents. Sentiment analysis attempts to identify a sentiment held towards an object and helps in the automation of extraction or classification of sentiment from unstructured text. Further, in their research, the researchers describe Sentiment Analysis as an aim to determine the state of mind of a speaker or writer with respect to some topic or even the overall tonality of a document.

As Mohammad (2017) and Ebrahimi, Yazdavar and Sheth (2017) clearly puts it in their research, there are challenges in sentiment analysis such as subjectivity classification, word sentiment classification, detecting sentiment, document sentiment classification and opinion extraction, among other challenges. These challenges can be resolved through various computational approaches for sentiment analysis such as Linguistic Approaches and Machine Learning approaches. Linguistic approach relies on disambiguation, using background information such as a set of rules and vocabularies. Thus, such a system normally contains lexicons, which consist of words and their polarity values such as 'positive/negative' or 'bad/good' (Wankhade et al., 2022; Gao, Feng, Song, & Wu, 2019). There are also a set of rules that help produce more accurate results as an integral part of such a system. The Machine Learning approaches however, are used for automatic sentiment classification and are approved by many researchers as the efficient way to

analyze sentiment laden terms in a document. The researchers also recommend that improving the quality of these systems is an area for future work.

With this regard, feature selection and extraction has been exhibited as an important area geared towards the improvement of quality and efficiency in Sentiment Analysis by many researchers (Tripathy et al., 2017 & Pratiwi, 2018). This partly forms the basis of this study. Das, Mishra and Rout (2019) together with Hasan and Abdulazeez (2021) suggested that employing various feature reduction and extraction techniques decreases the running time of learning while increasing success rate of algorithms.

Zheng et al. (2018) also acknowledges that feature selection is significant for Sentiment Analysis, as opinionated texts may have high dimensions, which have proven to adversely affect the performance of classifiers. These high dimensions may include mixed sentiment in a piece of text, sarcastic tones present in texts, presence of slang or shorthand writing among others (Ebrahimi et al., 2017).

Das et al. (2019) presented a Firefly algorithm with a very high evolutionary framework that delineated outstanding performance for their prediction model. The feature reduction and selection method tried to answer common questions that users have when looking for new techniques to select distinctive features to result in improvement of classification performance. Consequently, it is clear that to reach an optimal performance level, and improve efficiency of classifiers during analysis, it is advisable to include important features in the prediction and extraction of Sentiment information. These important features can be referred to as 'relevant' features (Badillo, et al., 2020).

Deniz and Kiziloz (2017) researched and proposed a model, using n-gram features, stemming and feature selection to overcome some Turkish newspaper article classification challenges in sentiment analysis. The researchers also acknowledged, according to their findings, that feature selection in Sentiment analysis could improve classifier performance. The proposed method used different preprocessing techniques, namely; n-

gram, stemming, and punctuation removal. The researchers then examined the effects of each preprocessing method separately, and presented their results.

Asif, Ishtiaq, Ahmad, Aljuaid and Shah (2020) focused their research on sentimental analysis of social media multilingual textual data to discover the intensity of extremist sentiments. The study classified the incorporated textual views into four categories that included high extreme, low extreme, moderate, and neutral; based on their level of extremism. On the initial part of their experiments, a multilingual lexicon with the intensity weights was created. This lexicon was validated from domain experts and it attained 88% accuracy for validation. Classification was performed by using different supervised and unsupervised algorithms and concluded that supervised algorithms performed better than unsupervised algorithms. In supervised algorithms, Linear Support Vector Classifier resulted in the highest accuracy of 82% as feature selection methods were also incorporated. The unsupervised algorithms – KNN, classified with an accuracy of 26%.

Vosoughi et al. (2016) used contextual information to better predict the sentiment of tweets by studying the rich metadata of tweets, which included location, timestamp and author information. This was achieved by the use of a distant supervised Twitter sentiment classifier, Bayesian approach and a standard n-gram model, which achieved better accuracy performance as compared to other state of the art distant supervised Twitter sentiment classifiers. The Bayesian approach was used to incorporate the relationship between the contextual factors and tweet sentiments into standard n-gram based Twitter sentiment classification. Probability of the contextual feature sentiments was calculated as being positive or negative based on historical tweets. However, the researchers did not consider other contextual features that could be predictive of sentiment on Twitter such as topic type features, for example hash tags.

Alayba et al. (2017) carried out a research that detailed a process of collecting tweets, filtering, pre-processing and carrying out text normalization. Their initial experiments were conducted by utilizing Deep Neural Networks and several other Machine Learning

algorithms while, a combination of “Unigram” and “Bigram” techniques were used for text feature selection. The best performing classifiers for the study were Machine Learning algorithms.

Apart from feature selection and extraction, Sabri and Saad (2016) acknowledges that Feature Engineering is also a very important task in the domain of Sentiment Analysis and generally in text categorization, and converting original documents to feature vectors is critical. The study proposes a supervised learning approach for sentiment analysis in Arabic language, while empirical study was done to evaluate various Feature Selection Methods and showcase that selection of the right feature set, determines the overall performance of classifiers.

Shirbhate and Deshmukh (2016) in their research, focused on acquiring a considerably reduced amount of features, without affecting the performance of the classifier. The classifier was able to determine positive, negative and neutral sentiments from tweets. The classifier was based on the multinomial Naïve Bayes classifier that uses Unigram and POS tags as features. These researchers also found that different types of features and classification algorithms could be combined in order to increase performance, and so, they combined Naïve Bayes classifier with Mutual Information Feature selection algorithm (MI). They however did not explore other combinations such as combinations of classifier algorithms. By using MI feature selection algorithm along with Naïve Bayes classifiers, the system recorded an improved performance in terms of accuracy.

Duwairi and Qarqaz (2014) in their paper, researched on Sentiment Analysis in Arabic reviews from a Machine Learning perspective, and employed three classifiers, Naïve Bayes, SVM and K-Nearest Neighbor in a parallel experimentation to detect polarity of reviews. These classifiers were run on the dataset of tweets and comments that were collected from Twitter and Facebook. The tweets and comments addressed general topics such as education, sports and political news. Precision and recall were used as performance metrics and evaluation for each classifier.

Most of these researches involved single models for Sentiment Analysis and classification. However, this study was mainly intended to create and use Ensemble Machine Learning approach, that is, a combination of Support Vector Machine (SVM) and Naïve Bayes for Classification of Sentiments. As Zhou (2021) describes in his book, Ensemble methods, also referred to as multiple classifier systems, try to construct a set of learners or algorithms and combine them for generalization. Their outputs could also be combine into a single prediction. The generalization of ensembles is often assumed by various researchers to be much stronger than the use of single learners or algorithm. Among other various advantages, ensembles are able to boost weak learners and are unlikely to over fit and improve performance (Rokach L. , 2005). According to Džeroski, Panov, and Ženko (2009), the purpose of learning is typically to achieve better predictive performance as ensembles can be typically more accurate than single learners. Conclusively, majority of state-of-the-art sentiment analysis makes use of accuracy, F1 score, and precision as performance evaluation parameters (Wankhade et al., 2022). One of our proposed aim was therefore to evaluate the performance of our proposed classifiers for Sentiment Classification in terms of accuracy, precision and robustness.

2.2 Text Mining

According to Vijayarani and Janani (2016) Text Mining is a domain in Data Mining that is used to extract interesting information, knowledge or pattern from both unstructured and semi-structured data in databases. This kind of analysis involves processes such as document gathering, pre-processing, text transformation, attribute selection, pattern selection and finally interpretation and evaluation of results.

Data mining however, is the process of extracting hidden predictive information from databases and transforming it into meaningful and understandable formats for future use (Gupta & Chandra, 2020).

Text mining is used to analyze large quantities of Natural Language text – these include text in both Standardized languages and Informal languages, or used in Computational

Linguistic. It is assumed universally by researchers that writers or speakers have some affective value (sentiment) to entities (Hovy, 2015). Human beings use their understanding of emotional intent of words to infer as to whether a text is ‘positive’ or ‘negative’, or maybe characterized by other more nuanced emotions such as ‘surprise’ or ‘disgust’ (Silge & Robinson, 2020). Text mining helps detect lexical patterns that are used for the ultimate extraction of unseen data or useful information (Vijayarani & Janani, 2016). This unseen data is useful to organizations and individuals in various Social listening applications, for instance, business analytics or trend predictions. Moreover, the Text Mining tools can be used to approach the emotional content of text programmatically (Silge & Robinson, 2020).

2.2.1 Opinionated Information

Textual information can be categorized into Factual and Opinionated information (Pawar, Jawale, & Kyatanavar, 2016; Carrillo-de-Albornoz, Rodriguez Vidal, & Plaza, 2018). Factual information (facts) are objective expressions that describe entities, events and their properties, while, Opinionated information (opinions) are commonly subjective expressions that describe individual’s sentiment, opinions or feelings toward entities, events and their properties. These kind of information are mostly available on user-generated content platforms such as Social networks, internet forums, discussion groups and blogs (Chaturvedi, Cambria, Welsch, & Herrera, 2018).

Sentiments and opinions can be analyzed in terms of the topic of the sentiment/ opinion, the holder of the sentiment/ opinion, the claim and sentiment within the a document (Li, Peng, Sun, Chai, & Wang, 2017). The holder believes a claim about a topic, and associates a sentiment, for example ‘good’ or ‘bad’, with his/her belief. Generally, human beings love to express their sentiments or opinions and emotions over entities or objects in both their physical and online interactions, for example, “this lipstick shade is better than that one” or “I hate this food!” As a result, it is evident that there is a great deal of importance to opinions especially in decision making. In e-commerce, we sometimes need to review other people’s opinion in order to make purchase decisions (Chaturvedi et al., 2018).

Large volumes of data are therefore collected from the World Wide Web, while ‘user-generated’ content forums are also provided to collect such opinions and sentiments, which can also be referred to as an online word of mouth, and are later presented to the world. To help navigate through these large volumes of data and address challenges that are posed during analysis and mining of the said data, adoption of processes such as Sentiment Analysis and Opinion Mining have risen to provide automatic and semi-automatic methods for generalization and interpretation of Sentiments/Opinions in form of texts (Liu B. , 2012).

2.2.2 Social Networking

Social Networking is a social infrastructure technology that allows microblogging. It is the grouping of individuals into specific groups such as cliques. These groups can be influenced on various grounds based on friendship, political, religious, business or peer preferences; all together sharing information of their interest online. Social networks enable users to share information, knowledge and support each other in an informational and emotional way (Forouzandeh, Sheikahmadi, Rezaei Aghdam, & Xu, 2018)

Social Media however, are the forms of electronic communication such as Websites for social networking and microblogging, through which users create online communities to share information, ideas, personal messages, and other contents such as videos. Examples can include websites like Facebook, Twitter, and LinkedIn (Shah, 2017).

Social Networking and Social Media are platforms that are characterized by ease of access and are mostly free of charge to users. For this reason, as well as the increased dependence on technology globally, individuals and organizations indulge in these forms of interactions to make new friends, stay in touch with loved ones, give feedback or opinions on product or service, and have fun while doing business (Vosoughi et al., 2016; Shah, 2017).

Consequently, a lot of opinionated information are generated from such interactions that are collectively referred to as User-generated content. These are textual information that can be inform of known standard languages e.g. English or Informal and mixed linguistic languages (Kim & Johnson, 2016).

2.2.3 Microblogging.

According to Gao and Li (2017) Microblogging is a form of online collaboration that has become increasingly popular with time. It is a web service that allows subscribers to relay short or ‘micro’ textual messages, links, videos and images to other users of the service. These contents can be sent from either a computer or mobile device.

‘Blog’ is a short word for ‘web blog’ which provides a platform where users write about their feelings and activities in form of sentiments or opinions, so other people can read them. Blogs are therefore generally website that are periodically updated by authors and writers about and around some common theme or topic of interest. Commercial microblogs exist to promote service and/or products, websites and collaboration within an organization or externally with consumers. Microblogging tools include Twitter, Facebook, and Instagram among others (Shirbhate & Deshmukh 2016).

The major appeal of microblogging is for both its immediacy and profitability in terms of ease of use and accessibility. Blog posts are typically brief and short, and can be relayed with various computing devices including phones (Gao & Li, 2017). According to Hornak (2009) with microblogging, one can get quick tidbits of information to many or hundreds of potential user or followers in seconds. Some of the benefits that can be achieved by individuals or companies include building brand awareness, growing a business network, giving the client base important announcements, and finally, giving and getting feedback both internally and externally from stakeholders.

2.3 Sentiment Analysis

Sentiment Analysis is also a text analysis process that can be referred to as an analysis involving the use of Machine Learning, Natural Language Processing and Computational Linguistics; to identify and extract structured information from the unstructured, voluminous and highly dimensional text documents, to drive profitability and foster better decision making. It is also referred to as Opinion mining (Wankhade et al., 2022; Bhadane, Dalal, & Doshi, 2015; Ahmad et al., 2017). It is therefore the application of Natural language processing, Computational linguistics, Text analytics to identify and extract subjective information from source materials (Hovy, 2015) such as internet text like documents, Product reviews, Tweets and other social media materials (Tejwani, 2014).

The term Sentiment Analysis can be interchangeably used with Opinion Mining, and involves the process of computational treatment of opinions, sentiments, and subjectivity in texts (Pang & Lee, 2008; Zhang & Liu, 2017). Its basis is to ascertain the attitude of a writer or speaker, in reference to certain topic or specific targeted object or entity, or the overall contextual polarity of a document (Hovy, 2015). This ‘attitude’ that changes overtime, reflects the speaker’s or writer’s appraisal, judgment, opinion, or evaluation; his/her affective state, which is the emotional state of the writer at the time of writing, and intended emotional communication, which refers to the emotional effect of the writer, the writer wishes to have on the reader (Hovy, 2015).

Sentiments on the other hand are emotions, judgments, opinions or ideas, and can also be thoughts or views based on emotion instead of reason. It is a kind of subjective impression, and can be termed as the expression of sensitive feeling in art and literature (Yadollahi et al., 2017).

Human beings are subjective creatures by nature (Tejwani, 2014). Our decisions are based on or influenced by stimuli, which includes personal feelings, tastes or opinions. According to Rana and Cheah (2016) together with Li et al. (2017) as much as sentiments

reflect the holder's emotions or desires, people however, express their sentiments in complex ways. These can be in terms of explicit or implicit expressions. For example, "I think that attacking Somali would put the Kenyan Government in a difficult position", is an implicit sentiment, while, "The Kenyan attack on Somali is wrong", is an explicit sentiment.

Explicit sentiments are aspects explicitly mentioned as nouns or noun phrases in a sentence, while, implicit sentiments are those aspects not explicitly mentioned in a sentiment or sentence but are implied (Rana & Cheah 2016).

2.3.1 Sentiment Analysis Process in R-Studio

With the increasing importance of computational text analysis and Data mining (Boumans & Trilling, 2016; Grimmer & Stewart, 2013), many researchers face the challenge of learning how to use advanced software that enables Text analysis. Currently, one of the most popular environments for computational methods and the emerging field of Data Science is the R statistical software, R-Studio (R Core Team, 2017).

R-Studio is a free, open-source, cross-platform programming environment, and according to Welbers, Atteveldt and Benoit (2017) in contrast to most programming languages, R-Studio was specifically designed for statistical analysis, which makes it highly suitable for Data science applications such as Sentiment Analysis. R-Studio has tools that are adequate for Sentiment Analysis. The tools now available for carrying out text analysis in R environment make it easy to perform powerful, cutting-edge text analytics using only a few simple commands.

Welbers et al. (2017) also points out that one of the main reasons for R's explosive growth has been its densely populated collection of extension software libraries, known in R terminology as packages, supplied and maintained by R's extensive user community. Each package extends the functionality of the base R language and core packages, and in addition to functions, data must include documentation and examples, often in the form

of vignettes demonstrating the use of the package. The best-known package repository, the Comprehensive R Archive Network (CRAN), currently has over 10,000 packages that have been published, and which have gone through an extensive screening for procedural conformity and cross-platform compatibility before being accepted by the archive. R software therefore features a wide range of inter-compatible packages, maintained and continuously updated by scholars, practitioners, and projects such as R-Studio and rOpenSci. Furthermore, these packages may be installed easily and safely from within the R environment using a single command. R thus provides a solid bridge for developers and users of new analysis tools to meet, making it a very suitable programming environment for scientific collaboration.

Welbers et al. (2017) further explains that Text analysis in particular has become well established in R environment. There is a vast collection of dedicated text processing and text analysis packages, from low-level string operations (Gagolewski, 2017) to advanced text modelling techniques such as fitting Latent Dirichlet Allocation models (Blei, Ng, & Jordan, 2003; Roberts, et al., 2014).

According to Welbers et al. (2017) one of the main advantages of performing text analysis in R-Studio is interoperability. This means that it is often possible, and relatively easy, to switch between different packages or to combine them. This helps to maximize flexibility and choice among users, for example, the *tif* (Text Interchange Formats) package (rOpenSci Text Workshop, 2017) describes and validates standards for common text data formats, while R-Studio also has functions that allow reading texts from various types of file formats such as, txt, csv and pdf into a raw text corpus. As a result, learning the basics for text analysis in R provides access to a wide range of advanced text analysis features.

2.3.2 Approaches Used for Informal or Mixed Linguistic Languages' Sentiment Analysis.

According to Soong et al. (2019) and Lo et al. (2017) based on Sentiment Analysis taxonomy, there are two approaches that are fundamentally used for Informal or mixed linguistic language Sentiment analysis, these are Subjectivity and Polarity detection.

2.3.2.1 Subjectivity Detection

Subjectivity detection involves procedures that help in understanding if contents contain personal views and opinions in private state. Subjective expressions may arise from cultures or various experiences of an individual or community, and hence can be localized and made specific to individuals or society as a whole. According to Sarker (2021), most studies on Sentiment analysis focus on highly subjective texts, such as product reviews, movie reviews or more generally, textual data reviews.

2.3.2.2 Polarity Detection

Polarity detection is about studying subjective expressions in terms of different polarities, intensities or rankings. These categories can also be referred to as classes, such as 'positive', 'negative' and 'neutral' or 'joy', 'sadness', 'anger' and 'fear' (Rashid, Anwer, Iqbal, & Sher, 2013; Namugera, Wesonga, & Jehopio, 2019).

2.4 Automation of Sentiment Analysis

Social Networking and microblogging have revolutionized social interactions on the online platform. Individuals and organizations interact online to carryout business transactions, share information, connect to friends or loved ones, while also indulge in gossip. This has been done with no particular format or standard, as users' sentiments are coupled with rampant errors in spelling, poor punctuations and are written quickly just as thought. These sentiments can also be written in both standard and Informal languages.

For this reason, Social Networking has become very popular (Shah, 2017; Gao & Li 2017).

With the large amount of opinionated information generated from these interactions, it is therefore critical to make fact based decisions, analyze user behavior and make real time assessments that depend on how fast the individuals or organizations can make insightful predictions from these voluminous and disseminated information, so as to drive profitable actions (Shirbhate & Deshmukh 2016; Shah, 2017).

Consequently, there is a need to automate the process of Sentiment Analysis since too much work is involved in manual processing. Various approaches in both Machine Learning and Natural Language Processing are adopted; however, Machine Learning and Artificial Intelligence related approaches are most popular (Jotheeswaran & Koteeswaran, 2015; Neethu & Rajasree, 2013; Asif et al., 2020).

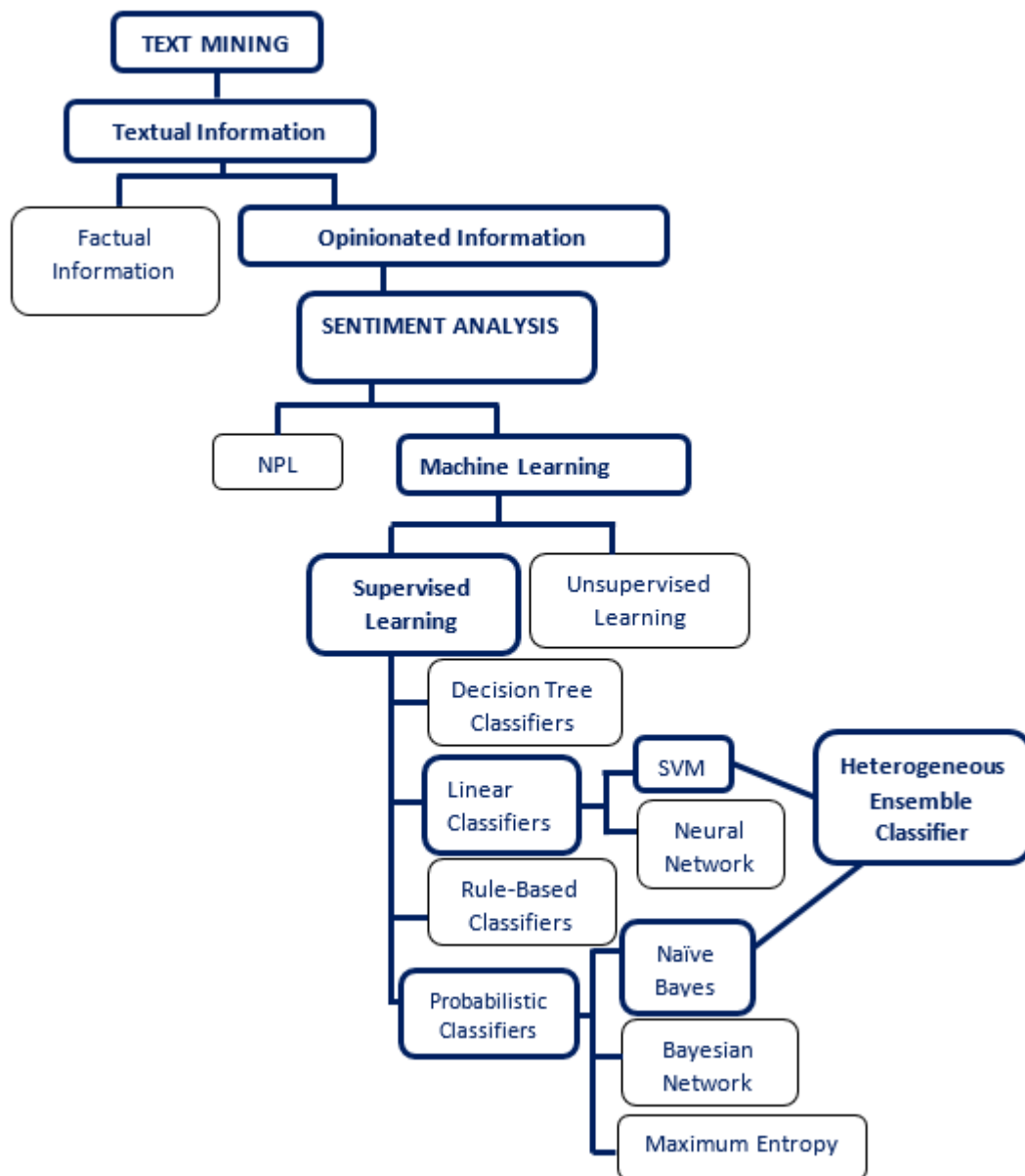


Figure 2.1: Machine Learning

2.4.1 Machine Learning.

According to Chatsiou and Mikhaylov (2020) Machine Learning is a field concerned with the question of how to construct computer programs that automatically improve with experience. This means that Machine Learning explores the study and construction of algorithms that can learn from, and make predictions on data. Precisely put, a computer program is said to learn from experience ' E ' with respect to some class of tasks ' T ' and performance measure ' P ', if its performance at tasks in ' T ', as measured by ' P ', improves with experience ' E '.

Chatsiou and Mikhaylov (2020) and Cho, Vasarhelyi, Sun and Zhang (2020) state that the purpose of Machine Learning is to learn from training data so as to make predictions on new or unseen data, this is what is referred to as 'Generalization'. These learning algorithms operate by building specific models through learning or training, while the learned model can be called a hypothesis or a learner. There exist different learning settings, among which the most common ones are supervised learning and unsupervised learning (Chatsiou & Mikhaylov, 2020).

As demonstrated in Figure 2.1, Machine Learning has sub categories of Supervised Learning, Unsupervised Learning and Semi-Supervised Learning.

2.4.2 Supervised Machine Learning.

Supervised Learning is a Machine Learning approach where all data is labelled and the algorithms learn to predict the output from the input data or the training dataset. Specific output values are supplied, and the algorithm iteratively makes predictions on the training data, while learning is improved with the experience learned. In this approach, learning stops when the algorithm achieves an acceptable level of performance (Hastie, Tibshirani, & Friedman, 2009).

According to Korjus, Hebart and Vicente (2016) and Alpaydin (2014) Supervised Machine Learning require splitting of the dataset into a training set and a test set. A

training set is the actual dataset or a sample of the dataset used to train or fit a model. The model learns from the training set. A test set on the other hand, is the sample of the dataset used to provide an unbiased evaluation of a final model trained on the training dataset. It provides the ultimate standard used to evaluate the model. Finally, the goal of Supervised Machine Learning for a classification problem is to find a model that accurately classifies and generalizes. To test the generality of a learned model, that model is typically applied to the test dataset and the prediction outcome then informs a researcher about the performance of the model. This therefore informs the understanding that the learning process in a simple Machine Learning model is divided into two steps; training and testing (Nasteski, 2017).

Further, Korjus et al. (2016) states that finding an optimally performing model as per the researcher's objective requires a set of assumption and a trade-off in model complexity. Too simple parameters lead to under-fitting, meaning that the model is not able to account for the complexity of the data; while too complex parameters lead to over-fitting, meaning that the model is too complex and fits to noise in the data.

The trade-off in model complexity refers to the bias-variance trade-off. Bias is the difference between the average prediction of a model and the correct value of the parameter being predicted. A model with high bias refers to a model that does not accurately classify the training data and oversimplifies the model. This leads to high errors on training and test data. Variance on the other hand, is the variability of the model's prediction for a given data point. A model with high variance focuses on the training data and does not generalize on the data which it has not seen before – the test set. Consequently, such models perform very well on training data, but dismally on test data, leading to errors (Singh, 2018).

Maniruzzaman, Rahman, Ahammed and Abedin (2020) and Korjus et al. (2016) explains that to test the different research assumptions and optimize the bias-variance trade-off, a researcher may have to partition the dataset into training and test set, a common practice for model training in the field of machine learning.

2.4.2.1 Supervised Learning Classifiers in Text Classification

The recent explosion of information and the availability of the increasing number of electronic documents from a variety of sources, together with the emergence of the Information age era, has particularly had an impact on businesses. These involve impact on corporate strategies and business decision making processes. Businesses need intelligence inform of information to help understand the market trends, and as a tool for their day-to-day operations. They therefore need to obtain accurate and relevant information about their products, competitor's activity or consumer feedback in their sectors. For this reason, text-mining studies have gained more importance. The main purpose of text mining is to enable users extract information from textual resources and deal with the operations such as retrieval, summarization and classification (Korde & Mahender, 2012).

2.4.2.2 Text Classification Process

According to Undhad & Bhalodiya (2017) and Korde and Mahender (2012) the aim of research on text classification is to improve the quality of text representation and develop high quality performing classifiers. The following steps are therefore included in Text classification process;

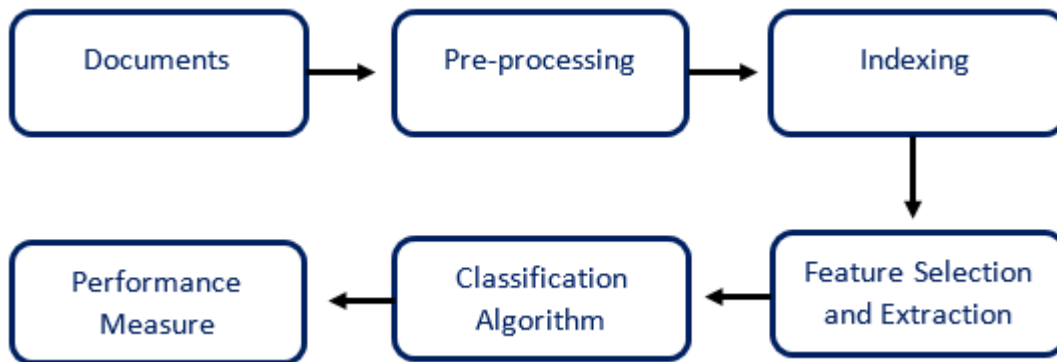


Figure 2.2: Steps in Text Classification

Documents: This is the first step in text classification process, which entails the collection of data from different data formats such as pdf, doc, and html.

Pre-processing: The data collected from various source is often incomplete, inconsistent and likely to contain errors. It is therefore taken through various other procedures such as stop word removal, to present the data in the text documents in clear word formats.

Indexing: This is a document representation, and is one of the pre-processing technique, which is used to reduce the complexity of the documents and make them easier to handle. The documents are transformed from full text documents to document vectors.

Feature selection and extraction: The aim of Feature selection and extraction is to select and extract subset of features from the main documents. Feature selection is performed by keeping the words with the highest score according to predetermined measure of the importance of the word.

Classification: Documents are classified into predetermined categories. This is widely done by using Machine Learning approaches such as Naïve Bayes, Decision tree, k-NN, SVM, Neural networks among others.

Performance Measure: This is the last stage of Text classification where the evaluation of text classifiers is typically carried out experimentally. The experimental evaluation of

classifiers usually tries to evaluate the effectiveness of a classifier, that is, its capability of taking the right classification decision. Measures may include accuracy, precision and robustness

Techniques for text classification can be classified in two main commonly used approaches; Linear approaches and Probabilistic approaches (Fragos, Belsis, & Skourlas, 2014).

2.4.2.2.1 Linear classifiers

The goal of classification in machine learning is to group items that have similar feature values into classes or groups. A linear classifier achieves this by making a classification decision based on the value of the linear combination of the features (Osisanwo, et al., 2017). If the input feature vector to the classifier is a real vector \vec{x} , then the output score can be found in the equation below;

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right) \quad 2.1$$

Linear models for classification such as Support Vector Machine (SVM) separate input vectors into classes using linear (hyperplane) decision boundaries (Osisanwo, et al., 2017). Osisanwo, et al. (2017) further elaborates that for a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane, such that, all points on one side of the hyper plane can be classified as ‘yes’, while the others are classified as ‘no’, as shown in Figure 2.3. He further states that a linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when the real vector (\vec{x}) is sparse. At the same time the researcher points out that linear classifiers often work very well when the number of dimensions in \vec{x} is large, especially in document classification, where each element in \vec{x} is typically the number of counts of a word in a document. The rate of convergence among dataset variables however, depends on the margin. Roughly speaking,

the margin quantifies how linearly separable a dataset is, and hence how easy it is to solve a given classification problem (Setiono & Loew, 2000 ; Osisanwo, et al., 2017).

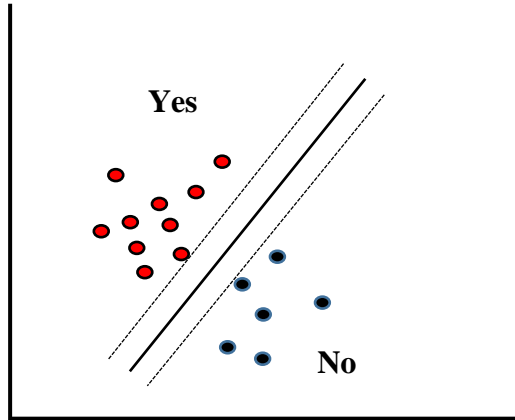


Figure 2.3: A 2-Dimensional Classification

2.4.2.2.2 Probabilistic Classifiers

The study of probabilistic classification is the study of approximating a joint distribution with a product distribution. Bayes rule is used to estimate the conditional probability of a class label, and then assumptions are made on the model, to decompose this probability into a product of conditional probabilities (Trivedi & Dey, 2013).

Trivedi and Dey (2013) further elaborates that a classifier is a function that assigns a class label to an example. From the probability perspective, according to Bayes Rule, the probability of an example $F = (f_1, f_2, \dots, f_n)$ being class C is

$$p(c|F) = \frac{p(F|C)p(C)}{p(F)} \quad 2.2$$

F is classified as the class $C = +$ if and only if;

$$B_c(F) = \frac{p(C = +|F)}{p(C = -|F)} \geq 1, \quad 2.3$$

Where $B_C(F)$ is called a Bayesian classifier.

Assume that all attributes are independent given the value of the class variable as in the below equation;

$$p(E|c) = p(f_1, f_2, \dots, f_n|c) = \prod_{i=1}^n p(f_i|c) \quad 2.4$$

The resulting classifier is then;

$$B_{nb}(F) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^n \frac{p(f_i|C=+)}{p(f_i|C=-)} \quad 2.5$$

The function $B_{nb}(F)$ can be referred to as naïve Bayesian classifier or Naïve Bayes. Naïve Bayes classifier being a simplest form of Bayesian network, is one of the most common model used in text classification since the original class attribute are assumed to be independent given the class label (Trivedi & Dey, 2013).

2.4.3 Unsupervised Machine Learning

Unsupervised Learning is a Machine Learning approach where algorithms are trained to use data that is not labelled. This means that no training data is provided and the algorithm is made to learn by itself while classifying the data without any prior experience (Sethi, 2020). The goal for unsupervised learning is to model the underlying structure in order to learn more about the data, provide insights that were previously unknown and to identify hidden patterns (Brownlee, 2019). As such, there aren't necessarily defined outcomes from unsupervised learning algorithms. Rather, this type of learning determines what is different or interesting from the given dataset. Unsupervised Learning algorithms can therefore be practically applicable in fraud detection and identification of human errors during data entry among other applicable areas (Sethi, 2020).

The goal for this study was to optimize performance criteria using experience (Chatsiou & Mikhaylov, 2020). For this reason, the Supervised Learning algorithms applied allowed

the researchers to collect data, label the data and produce a data output from the previous experiences.

According to Jurafsky and Martin (2017), one of the oldest tasks in Text classification is assigning a library subject category or topic label to a text. The researchers study reveal that subject category classification is the task for which Naïve Bayes algorithm was invented, while SVM has been applied successfully in many opinion mining tasks (Balahur & Perea-Ortega, 2015).

2.4.4 Bag-of-Word and n-gram Sentiment Modelling

Bag-of-word and n-gram Sentiment modelling were used to define a generic way of how the text reviews were structured and features extracted. N-gram is a sequence of n words from a given sequence of text or speech such as, letters or syllables. The n-grams are typically collected from a text or speech corpus. N-grams are commonly used in natural language modelling, and assumes that only the previous $n-1$ word in a sentence have any effect on the probabilities for the next word (Nguyen, Nguyen, Duong, & Snasel, 2016; Schmidt & Heckendorf, 2017).

Unigram text model is similar to bag-of-word text model. In unigram model, one word is used as a feature, bi-gram model; two words are used, while tri-gram models use three words as features. For example, the sentiment ‘This is a great looking phone!’ when modeled using unigrams would have ‘This’, ‘is’, ‘a’, ‘great’, ‘looking’, ‘phone’ as the sentiment features. Bigrams would model the sentiments as ‘This is’, ‘a great’, ‘looking phone’ as the sentiment features. Trigrams would model the sentiments as ‘This is a’, ‘great looking phone’ as sentiment features (Nguyen et al., 2016). Trigram and bigram models sentiments with dependence or relationship between words which make up the sentiments, which can be used to restrict the context within which words are used. The n-gram model was implemented by the n-gram tokenizer in R-Studio, available in R-Studio environment. This was achieved by choosing the minimum and maximum n-gram size.

For fixed-length n-grams, the minimum and maximum n-gram number value was equal, while in a variable-length, the minimum n-gram size value would have been less than the maximum n-gram size value. That is, If N = the number of words in a given sentence K , the number of n-grams for sentence K would be $N_{\text{grams } K} = N - (n - 1)$. The most frequent and the most informative words were used as unigram feature vectors, as this described the characteristics of the features that were used for classification in the model as seen in Figure 3.7 and Figure 3.8 (Nguyen et al., 2016).

Bag-of-words modelling on the other hand, according to Jurafsky and Martin (2017) was used as the simplest representation of text where we had unordered set of words, with their exact position ignored. Zhao and Mao (2017) together with Jurafsky and Martin (2017) indicated that Bag-of-word features were effective at capturing the general topic of the discourse in which the target words had occurred. They also stated that ‘Stop’ words such as “the”, “in” and “a”, together with ‘unknown’ words which intuitively seemed statistically irrelevant, were often filtered out in various studies, but were nevertheless sometimes very helpful in text categorization tasks in other experimental environments. Unknown words are words that may appear on the test set and not on the training set, or vice versa, during training (Pawar et al., 2016; Carrillo-de-Albornoz et al., 2018 & Jurafsky & Martin, 2017).

2.5 Sentiment Analysis Classifiers

This research was regarded as a qualitative research. According to Kumar (2011) a qualitative research usually involves studying perceptions, beliefs or feelings and a researcher does not make any attempt to establish uniformity in them across respondents.

The use of algorithms is usually dependent on the type of problem to be studied (Osisanwo, et al., 2017). In addition, a researcher can look at a number of dimensions to give a sense of what will be a reasonable algorithm or classifier. Some of these could include answering questions such as; does the researcher expect the problem to be linearly separable? What is the relevance of dimensionality of the feature space? Is over fitting

expected to be a problem? And also what are the system's requirement in terms of speed, performance, memory usage? In addition, selection of a classification model may also be made on the basis of factors such as resources available, accuracy requirement, and training time available among other factors (Gupte, Joshi, Gadgul, Kadam, & Gupte, 2014). Wolpert and Macready (1997) also poses a fundamental question in there research that may be of guidance to a researcher; “how should we assess the performance of algorithms on problems so that we may programmatically compare those algorithms?”

The question of identifying a reasonable set of algorithm was therefore addressed by following the Occam's Razor principle: ‘Use the least complicated algorithm that can address your needs and only go for something more complicated if strictly necessary’. However, Osisanwo, et al. (2017) further suggest that, it is essential for a researcher to first decide upon a metric to measure performance, then, compare at least a handful of different algorithms in order to train and select a best performing model. The commonly used metric is classification accuracy, which is defined as the proportion of correctly classified instances.

2.5.1 Naïve Bayes Classifier.

Most language processing tasks are Classification tasks. According to Khan, et al. (2016) Naïve Bayes algorithm is extensively used for text classification and also widely used in solving classification problems such as text categorization and is based on Bayes theorem. The study focused on the problem of Text categorization, the task of classifying an entire text by assigning it a label drawn from some set of labels.

Sentiment Analysis is an important yet, common text categorization task that involves sentiment extraction. At the same time, the ‘positive’ or ‘negative’ orientation that a writer expresses toward an object or target is also an important aspect in Sentiment Analysis. A review of items such as a book, services or a movie on an online platform, reveals the authors’ sentiment towards the items (Wankhade et al., 2022; Bhadane, Dalal, & Doshi, 2015; Ahmad et al., 2017).

According to Jurafsky and Martin (2017), one of the oldest tasks in text classification is assigning a library subject category or topic label to a text. For instance, deciding whether a research paper concerns epidemiology or instead, perhaps, embryology, is an important component of information retrieval. The researchers study, also reveal that subject category classification is the task for which Naïve Bayes algorithm was invented for in the year 1961. The researchers further state that the goal of classification is to take a single observation, extract some useful features, and thereby classify the observation into one of a set of discrete classes. There are various methods of classifying texts such as the application of rule-based classifiers, which, according to the researchers prove to be fragile, as situations or data change overtime while for some tasks humans aren't necessarily good at coming up with the rules. For this reason, most problems of classification in language processing are done by Supervised Machine Learning. However, Decision Tree classifier training in Supervised Learning on the other hand, are relatively expensive as complexity and time taken to train the model is more (Dhiraj, 2019).

This study therefore implemented Naïve Bayes algorithm for sentiment detection, while experiments such as 'Bag-of-words' and n-gram were carried out for subjective analysis of the datasets. Naïve Bayes has also been shown by Osisanwo, et al. (2017) and Viegas, Gonçalves, Martins and Rocha (2015) to be superior in terms of CPU and memory utilization. In addition, the algorithm is a probabilistic classifier and is used to indicate the probability of observation being in the class, by calculating the conditional probability of each attribute occurring in the predicted classes. The algorithm also assumes attribute independence, meaning, each individual feature is assumed to be an indication of assigned class, independent of each.

2.5.1.1 Multinomial Naïve Bayes Classifier

For the purpose of implementing Naïve Bayes classifier in this study, our focus was the Multinomial Naïve Bayes. Multinomial Naïve Bayes classifiers is mostly used in text

classification and ultimately makes two simplifying independence assumptions (Jurafsky & Martin, 2017; Khan, et al., 2016).

1. Bag-of-words assumption: where it is assumed that the position of words does not matter. A word such as the word ‘great’ has the same effect on classification whether it occurs as the 1st or 20th or even the last word in a document. Thus, it is also assumed that features f_1, f_2, \dots, f_n only encode word identity and not position.
2. Conditional independence assumption: where it is assumed that the probabilities of features $P(f_i|c)$ are independent given the class ‘C’, and hence can be ‘naively’ multiplied as follows:

$$P(f_1, f_2, \dots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \dots \cdot P(f_n|c) \quad 2.6$$

Jurafsky and Martin (2017) further explains that Multinomial Naïve Bayes classifier is called so because it is a Bayesian classifier that makes a simplifying ‘naive’ assumption about how the features interact. The intuition of Multinomial Naïve Bayes algorithm is that text documents are represented as if they were a bag-of-words. This means that unordered set of words with their positions on the document ignored are analyzed, while keeping only their frequency in the document. For example, instead of representing the word order in all the phrases like, ‘I love this movie’ and ‘I would recommend it’, simply, what is noted, is the number of times each word occurs in the entire document.

According to Abbas, et al. (2019) Given a Hypothesis (H) and evidence (E), Bayes' Theorem states that the relationship between the probability of the hypothesis before getting the evidence, $P(H)$, and the probability of the hypothesis after getting the evidence, $P(H|E)$, is :

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Bayes Rule

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad 2.7$$

To apply the Naïve Bayes classifier to text, we considered word positions, by simply introducing an index through every word position in the document as follows;

Positions \leftarrow all word positions in the test document

$$C_{NB} = \operatorname{argmax} P(c) \cdot \prod_{i \in \text{positions}} P(w_i|c) \quad 2.8$$

Naïve Bayes calculations were done in log space (Abbas, et al., 2019; Jurafsky & Martin, 2017). This was to avoid underflow and increased speed, thus Equation (2.9) was generally instead expressed as the final equation;

$$C_{NB} = \operatorname{argmax} \log P(c) + \sum_{i \in \text{positions}} \log P(w_i|c) \quad 2.9$$

2.5.1.2 Training the Multinomial Naïve Bayes Classifier

Further, Abbas, et al. (2019) together with Jurafsky and Martin (2017) explains that to learn the probabilities of $P(c)$ and $P(f_i|c)$, that is, the prior probability of a given class ‘c’ and the probability likelihood of a given feature f_i given a class ‘c’; we considered the maximum likelihood estimate by using frequencies in the data. For $P(c)$, we identified what percentage of documents in the training set were in each class ‘c’. Let N_c be the number of documents in our training data with class ‘c’, and N_{doc} be the total number of documents. Thus;

$$P(c) = \frac{N_c}{N_{doc}} \quad 2.10$$

To learn the probability $P(f_i|c)$, we assumed a feature is the existence of a word in the document’s bag-of-words, and so we had $P(w_i|c)$, which was computed as a fraction times the word w_i appeared among all words in all the documents of class ‘c’. We first summed up all documents with category ‘c’ into one big “category c” text. We then used

the frequency of w_i in this summed up document to establish (by counting) a maximum likelihood estimate of the probability (Abbas, et al., 2019; Jurafsky & Martin, 2017).

$$P(w_i|c) = \frac{\text{count}(w_i|c)}{\sum_{w \in V} \text{count}(w, c)} \quad 2.11$$

A vocabulary ‘V’ of the document which consisted of all the list of words in all classes, and not just the words in one class ‘c’, was created (Abbas, et al., 2019; Jurafsky & Martin, 2017).

2.5.1.3 Laplace Smoothing (add-1)

According to Jurafsky and Martin (2017), Abbas, et al. (2019) together with Liu and Martin (2011) Naïve Bayes Classifier has a problem with maximum likelihood training. For instance, the problem of ‘unknown word’ particularly in cases where if a feature (or word) does not occur in any document in the training set, all documents in the test set that contain this same feature will be zero for all classes ‘c’, causing Multinomial Naïve Bayes to lose all discriminative power. In addition, rarely occurring features may also be problematic if smoothing is not performed. For example, a rare feature that may occur in some classes in the training set but does not occur in the test set will dominate probability estimates since it will force $P(w_i|c)$ to be zero, regardless of the values of the remaining word features. For instance, when trying to estimate the likelihood of the word “great” given a class ‘positive’, but there may be no training documents that contain the word “great” and are classified as ‘positive’. The word “great” may have occurred sarcastically in the class ‘negative’. In such a scenario the probability for this word feature will be zero as shown in Equation 2.12.

$$P(\text{great}|\text{positive}) = \frac{\text{count}(\text{great}|\text{positive})}{\sum_{w \in V} \text{count}(w \cdot \text{positive})} = 0 \quad 2.12$$

Since Naïve Bayes naively multiplies all the feature likelihoods together, zero probability in the likelihood word for any class will cause the probability of the class to be zero, despite the evidence (Jurafsky & Martin, 2017).

```

function TRAIN NAIVE BAYES(D, C) returns log  $P(c)$  and log  $P(w|c)$ 

for each class  $c \in C$            # Calculate  $P(c)$  terms
   $N_{doc}$  = number of documents in D
   $N_c$  = number of documents from D in class  $c$ 
   $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$ 
   $V \leftarrow$  vocabulary of D
   $bigdoc[c] \leftarrow$  append( $d$ ) for  $d \in D$  with class  $c$ 
  for each word  $w$  in V           # Calculate  $P(w|c)$  terms
     $count(w,c) \leftarrow$  # of occurrences of  $w$  in  $bigdoc[c]$ 
     $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \in V} (count(w',c) + 1)}$ 
  return  $logprior, loglikelihood, V$ 

function TEST NAIVE BAYES( $testdoc, logprior, loglikelihood, C, V$ ) returns best  $c$ 

for each class  $c \in C$ 
   $sum[c] \leftarrow logprior[c]$ 
  for each position  $i$  in  $testdoc$ 
     $word \leftarrow testdoc[i]$ 
    if  $word \in V$ 
       $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$ 
  return  $argmax_c sum[c]$ 

```

Figure 2.4: The Naïve Bayes algorithm, using Laplace Smoothing

Source: (Jurafsky & Martin, 2017)

Therefore Jurafsky and Martin (2017), Abbas, et al. (2019) together with Liu and Martin (2011) asserts that the solution to these limitations is parameter smoothing, and for the purpose of this study, we applied Laplace smoothing (add-1 smoothing) to prevent cases where missing, unknown or rarely occurring features inappropriately dominated the probability estimates in Multinomial Naïve Bayes. This approach is commonly used in Naïve Bayes text categorization:

$$P(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|} \quad 2.13$$

A demonstrative example:

Using a demonstrative example to demonstrate our implementation, we trained and tested Naïve Bayes with the Laplace smoothing and used a Sentiment analysis domain with two classes; *positive* (+) and *negative* (-) (Jurafsky & Martin, 2017; Abbas, et al., 2019; Liu &

Martin, 2011). We also used the following miniature training and test document in Figure 2.5 as simplified from OnePlus mobile product reviews on Twitter.

	Doc	Document	Class
Training	1	oneplus mobile 8pro rumored good pop selfie camera	+
	2	oneplus mobile 7t achieve hybrid armor case	+
	3	mobile note9 equipped everything need fulfill orders seamlessly	+
	4	issue with fortnite mobile	-
	5	samsung s9 sprint network worst mobile experience endure mon	-
Test	6	predictable with no fun	?

Figure 2.5: Miniature training and test documents simplified from OnePlus mobile product reviews

From the example in Figure 2.5, we had a total number of 5 documents in the training set, with 3 documents in the class ‘positive’ (+), while 2 documents were in the class ‘negative’ (-). Our goal was then to determine the class of the test document (doc₆).

We therefore first computed the priors as follows;

$$P(+)=\frac{3}{5}$$

$$P(-)=\frac{2}{5}$$

We then computed the conditional probabilities or likelihoods, for the words in the test set, ‘predictable’, ‘with’, ‘no’, ‘fun’. However, from miniature in Figure 2.5, the word ‘predictable’ did not occur in the training set therefore, as mentioned previously, we used Laplace Smoothing (add 1 smoothing) to find its likelihood. The total summation of words in the positive (+) class, for the training set was 23, with a vocabulary of 31 words. While

we had a total summation of words in the negative (-) class to be 13. We then established the count or appearance of the words in each of the given specific class (+/-).

Table 2.1: Computations for conditional probabilities for both positive and negative class

$P(\text{"predictable"} +) = \frac{0 + 1}{23 + 31} = \frac{1}{54}$	$P(\text{"predictable"} -) = \frac{0 + 1}{13 + 31} = \frac{1}{44}$
$P(\text{"with"} +) = \frac{0 + 1}{23 + 31} = \frac{1}{54}$	$P(\text{"with"} -) = \frac{0 + 1}{13 + 31} = \frac{1}{44}$
$P(\text{"no"} +) = \frac{0 + 1}{23 + 31} = \frac{1}{54}$	$P(\text{"no"} -) = \frac{0 + 1}{13 + 31} = \frac{1}{44}$
$P(\text{"fun"} +) = \frac{0 + 1}{23 + 31} = \frac{1}{54}$	$P(\text{"fun"} -) = \frac{0 + 1}{13 + 31} = \frac{1}{44}$

With the use of the Naïve Bayes final Equation (2.9), we thus carried out computation for the chosen class as follows;

$$\begin{aligned}
 P(+)\ P(\text{doc6} | +) &= \frac{3}{5} \times \left(\frac{1}{54}\right)^4 = \frac{1}{14171760} \\
 &= 0.00000007
 \end{aligned}$$

$$\begin{aligned}
 P(-)\ P(\text{doc6} | -) &= \frac{2}{5} \times \frac{2}{44} \times \left(\frac{1}{44}\right)^3 = \frac{1}{4685120} \\
 &= 0.000000213
 \end{aligned}$$

With this, the test document could be classified as a ‘*negative*’ document because the highest probability between the two computations was **0.000000213**. The model therefore predicted the class *negative* for the test document (doc₆).

2.5.2 Support Vector Machine Classifier.

Support Vector Machine (SVM) is a supervised machine learning algorithm which is also widely used for classification problems, even though it can also be employed for both classification and regression purposes. It is a non-probabilistic binary linear classifier that has the ability to linearly separate classes by a large margin (Al Amrani, Lazaar, & El Kadiri, 2018). According to Balahur and Perea-Ortega (2015), SVM has been proven to be highly effective in traditional text categorization and has been applied successfully in many opinion mining tasks, performing better than other machine learning techniques.

Machine learning algorithms involve, in most cases, minimising an error measure of some kind. This measure is often called an objective function or loss function. The objective of a linear programming problem would typically be, to maximize or to minimize some numerical value. It indicates how much each variable contributes to the value to be optimized in a problem (Anderson, 2005; Graves & Jaitly, 2014; Kim & Chung, 2019). The objective function therefore takes the following general form:

Maximize *or* minimize;

$$f = \sum_{i=1}^n c_i X_i \quad 2.14$$

Where: c_i = is the objective function coefficient corresponding to the i^{th} variable, and;

X_i = is the i^{th} decision variable.

The summation notation for the objective function can be expanded out as follows:

$$f = \sum_{i=1}^n c_i X_i = c_1 X_1 + c_2 X_2 + c_3 X_3 + \dots + c_n X_n \quad 2.15$$

The rationale behind SVMs for the purpose of this study, was best explained by considering a set of data points that belonged to one of the two classes, ‘*positive*’ and ‘*negative*’, as illustrated in Figure 2.6.

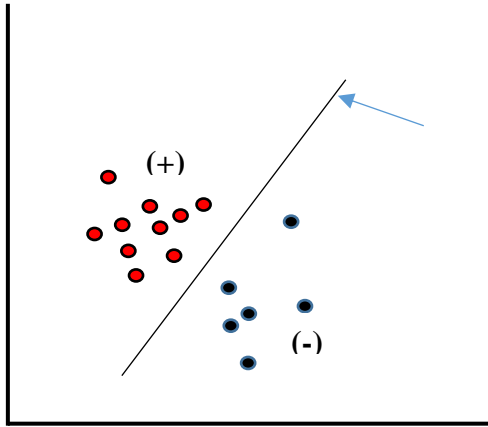


Figure 2.6: Linearly Separable Data

According to Al Amrani et al. (2018), SVM is based on the idea of finding a hyper plane that best divides a dataset into two classes, an optimal separating hyper plane that separates itself as far as possible from data points from each category as shown in Figure 2.6. These datasets could be referred to as, linearly separable datasets or, linear datasets.

From the Figure 2.6, it was clear that there was a non-zero distance between the two closest points, that is, between the categories, therefore, there were infinite numbers of possible separation lines. This raised the question as to whether it was possible to choose an optimal hyperplane. For SVM, the best or optimal hyperplane was the one that maximized the margin, that is, the distance between the separation boundary (the hyperplane) and the points that were closest to it (Al Amrani et al., 2018). This is illustrated in Figure 2.7.

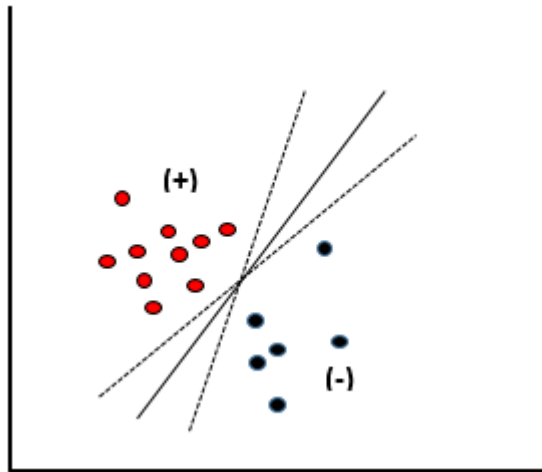


Figure 2.7: Multiple Separation Boundaries (Hyperplanes)

From the above criterion in Figure 2.7, the position of the hyperplane depends only on the data points that are closest to it. This implies that unlike other classification methods, the classifier does not depend on any other data points in dataset. The vectors that define the hyperplane, or the data points that are closer to the hyperplane and influence the position and orientation of the hyperplane are called Support Vectors. The distance between the hyperplane and the points closest to it, is referred to as the Euclidean Distance (Ishtiaque Ahmed & Nasrin, 2022). This is depicted in Figure 2.8. A direct implication of this was that the fewer the support vectors, the better the generalizability of the boundary. A vector is a quantity consisting of a magnitude and direction. Geometrically, a vector in a 2-dimensional plane (x and y graph) is a line from the origin to its coordinates. For instance, there would be coordinates (2, 3) where we can sketch a line from the origin to (2, 3) which is 2 on the x-axis, and 3 on the y-axis. Additionally, to calculate for magnitude, we would have to find the length between the origin (0, 0) and (2, 3) (Ishtiaque Ahmed & Nasrin, 2022).

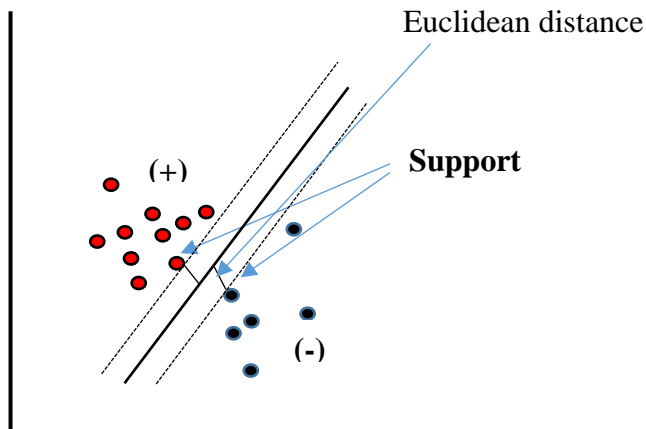


Figure 2.8: Optimal Hyperplane in a Supervised Linearly Separable Dataset

2.5.2.1 Linearly Separable Dataset - Soft Margin Classification

As depicted in Figure 2.6 linearly separable datasets are datasets in which the hyperplane can be easily drawn to separate two classes of data in a 2-dimensional space, using a straight line (Al Amrani et al., 2018; Balahur & Perea-Ortega, 2015). In practice, most datasets or real text data sets are seldom linearly separable and demonstrate very high dimensional problems, as common in text classification (Osisanwo, et al., 2017; Misra, 2019). Therefore, when dealing with such real data sets, a simple approach to deal with deviations from separating linear datasets was to allow a small number of Support Vectors to be misclassified. The number of possible misclassifications were governed by a free parameter ‘ C ’, which was referred to as the *Cost*. The ‘ C ’ could essentially be referred to as the penalty associated with making an error, or more arguably, the tolerance for errors was more or less accentuated with the Cost parameter. The higher the value of ‘ C ’ during an algorithm training, the less likely it was that a misclassification would occur. This approach is referred to as the *Soft Margin Classification*, as illustrated in the Figure 2.9. The ‘ C ’ parameter was therefore used to tune the classifier model. However, when ‘ C ’ was too high, the model would tend to over fit while, when ‘ C ’ was too low, the model would on the other hand tend to under fit. With the ‘ C ’ parameter, we were also able to develop a robust model with SVM, as our model was robust against the outliers, and controlled with the parameter ‘ C ’ (Mishra, 2020).

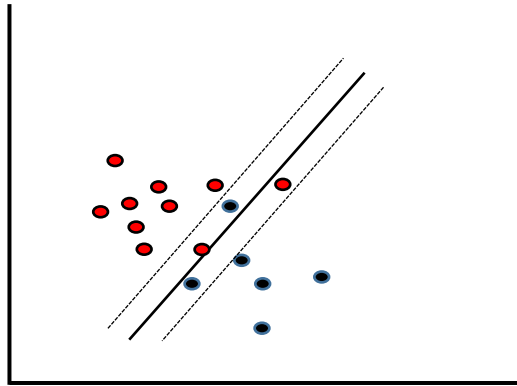


Figure 2.9: Soft margin classifier of a supervised linearly separable dataset

Real data sets are much more complicated and complex, and in most cases, cannot be dealt with using Soft margin classifiers. For example, Figure 2.10 presents a situation that requires the use of multiple and non-linear hyperplanes, while it is also clear that there is no single linear decision boundary (hyperplane) to separate the datasets. At the same time, the vectors are clearly segregated, and it appears as though it should be easy to separate them (Mishra, 2020).

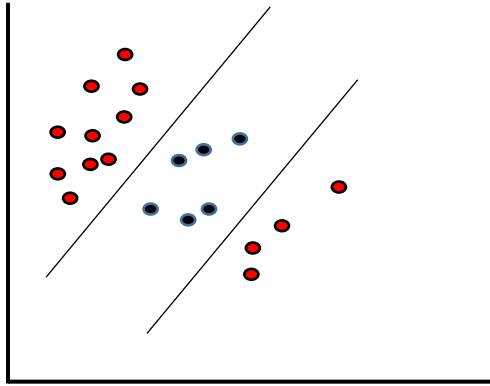


Figure 2.10: Non-Linearly Separable Data

The solution for such a case was to add a third dimension. We continually had two dimensions; x and y . We then created a new dimension ' f ', and ruled that it be calculated in a certain convenient way for the study as follows;

$$f = x^2 + y^2 \tag{2.16}$$

This now gave us another higher dimension, and could now introduce us to kernels (Bortolussi, Gallo, Křetínský, & Nenzi, 2022).

2.5.2.2 Non Linearly Separable Dataset - The Kernel Trick

As discussed earlier in linearly separable datasets, SVM algorithms works by finding an optimal hyperplane, which is a boundary that maximizes the margin. This process result in a straight line hyperplane, as also illustrated in Figure 2.8, which may not work in most cases (Zhang, 2018). This is because the notion of distance can be generalized in terms of considering the key properties that any measure of distance must satisfy, which are; Non-negativity, meaning a distance cannot be negative; Symmetry, which is the distance between a point A and a point B is the same as the distance between point B and point A; Identity, meaning that the distance between a point and itself is zero; finally, Triangle inequality, that is the sum of distances between point A and B and points B and C must

be less than or equal to the distance between A and C. Equality holds only if all three points lie along the same line (Ciaccia, Patella, & Zezula, 1997).

Mathematical objects that displays the above properties are related to distance and as such, they can be referred to as metrics, while, the mathematical space in which such metrics live is called a Metric Space (Sharma, 2019; Ciaccia et al., 1997). Metrics are defined using special mathematical functions designed to satisfy the above properties. These functions are known as Kernels. A kernel trick essentially maps a classification problem to a metric space (feature space), where the problem can be rendered separable by a simple hyperplane in the new higher (complex) dimensional space, with which each of the dimensions being a combination of the original problem variables. The function of kernel is to take data as input and transform it into the required form. SVM algorithms use different types of kernel functions such as linear, non-linear and Radial Basis Function – RBF (Bortolussi et al., 2022).

The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces. In practice, one does not mess around with the transformations, but uses the different kernels in trials to identify the one kernel that completes the job. The prediction resulting from using different kernels is then tested against a subset of the data. For the purpose of this study, we therefore implemented and experimented with the Linear Kernel, since our intension was to have a linearly separable dataset (Zhang, 2018).

2.5.2.3 The Classification Concept with Support Vector Machine Classifier

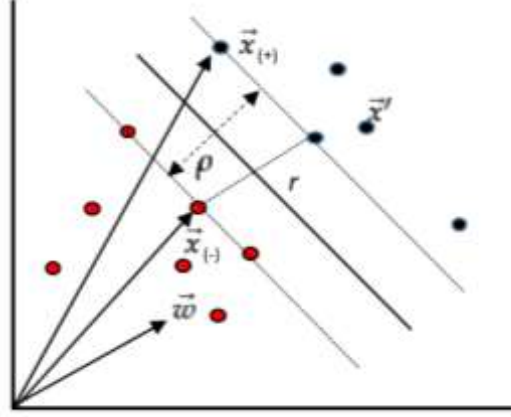


Figure 2.11: Geometric Margin of a point (r) and a decision boundary width ρ

To formalize the SVM with algebra, a decision hyperplane as shown in Figure 2.11, can be defined by an intercept term b and a decision hyperplane normal vector \vec{w} which is perpendicular to the hyperplane. In machine learning, the vector \vec{w} can be referred to as the weight vector (Manning, Raghavan, & Schütze, 2008). To choose among all the possible hyperplanes that are perpendicular to the normal vector, we specify the intercept term b . Since the hyperplane is perpendicular to the normal vector, all points \vec{x} on the hyperplane satisfy $\vec{w} \vec{x} = -b$. We then have a set of training data points $D = \{(\vec{x}_i, y_i)\}$, where each member is a pair of a point \vec{x}_i and a class label y_i corresponding to it. In SVMs, the two data classes are always named +1 and -1, or even 1 and 0; and the intercept term is always explicitly represented as b (Manning et al., 2008; Al Amrani et al., 2018). The linear classifier is then:

$$f(\vec{x}) = \text{sign}(\vec{w}\vec{x} + b) \quad 2.17$$

We were confident in the classification of a point if it was far away from the decision hyperplane (Manning et al., 2008).

Based on Al Amrani et al. (2018) research, we then used a training dataset of n points, such that; $(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)$, a normal vector \vec{w} to the plane and some unknown data

point on the plane point \vec{x} . Our main interest was knowing whether the unknown data point was on the ‘positive’ class or the ‘negative’ class category. At the same time we needed to also employ a constraint (cost) to control and regulate the degree of miss classification. When $C = -b$; we therefore found;

$$\vec{w} \cdot \vec{x} = C \quad 2.18$$

$$\vec{w} \cdot \vec{x} + b \geq 0 \quad 2.29$$

Without loss of generality if Equation 2.19 was true, with b being a constant, then we could make a decision rule that the unknown was a ‘positive’ sample.

In such a case, we don’t know which \vec{w} to use and which constant b to use. However, it is clear that \vec{w} has to be perpendicular to the hyperplane.

As seen in the works done by Manning et al. (2008) and Al Amrani et al. (2018), we further lay additional constraints to allow us to calculate the constant b and the vector \vec{w} . In a case where the data was linearly separable, the closest points from the two classifications, *positive* and *negative*, would be found when;

$$\vec{x} \cdot \overrightarrow{x_{(+)}} + b \geq 1 \quad 2.20$$

With Equation 2.20, any data point on or above this boundary was of *positive* class.

$$\vec{w} \cdot \overrightarrow{x_{(-)}} + b \leq -1 \quad 2.21$$

While Equation 2.21, demonstrates that any data point on or above this boundary was of the *negative* class.

y_i This meant that there was a separation of distance of +1 or -1 for all the data point samples. We then introduced where we considered our linear SVM that separated two classes $y_i = +1$ for positive samples, $y_i = -1$ for negative samples. Where we found the following;

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \text{ and } y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 \quad 2.22$$

The two equations are the same because when you multiply; $(\vec{x}_{(-)})(-1) = +1$

Therefore, we concluded that;

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0 \quad 2.23$$

We then added a constraint to the equation and set that it would be equal to zero for samples that end up in the margin;

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0 \quad 2.24$$

$$\vec{w} \cdot \vec{x} + b = 0 \quad 2.25$$

We then found a hyperplane such that, the set of points could then satisfy the below equation;

2.5.2.4 Finding the Maximum Width of the Geometric Margin

Al Amrani et al. (2018) also adds that to get ρ width of the margin and compute the Euclidean distance (r) as shown in Figure 2.11 we can subtract the negative sample vector $\vec{x}_{(-)}$ from the positive sample vector $\vec{x}_{(+)}$ and multiply it by a unit vector $\vec{w} / \|\vec{w}\|$ which is the normal vector \vec{w} divide by magnitude of \vec{w} denoted as $\|\vec{w}\|$, as shown in the equation below;

$$\vec{x}_{(+)} - \vec{x}_{(-)} \vec{w} / \|\vec{w}\| \quad 2.26$$

Using the equation; $y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$ we found that,

$$\vec{X}_{(+)} \cdot \vec{w} / \|\vec{w}\| \quad 2.27$$

$$y_{+1}(\vec{w} \cdot \vec{X}_{+1}) = 1 - b \quad 2.28$$

And;

$$\overrightarrow{X_{(-)}} \cdot \vec{w} / \|\vec{w}\| \quad 2.29$$

$$y_{-}(\vec{w})\overrightarrow{X_{-1}} = 1 + b \quad 2.30$$

$$\{(1 + b) - (1 - b)\} \cdot \vec{w} / \|\vec{w}\| \quad 2.31$$

Finally, Equation 2.32 was evident as follows;

$$\overrightarrow{X_{(+)}} - \frac{\overrightarrow{X_{(-)}}\vec{w}}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|} \quad 2.32$$

According to Manning et al. (2008) and Al Amrani et al. (2018) geometrically, the distance between the two hyperplanes (width of margin) is $2 / \|\vec{w}\|$, as shown in the Equation 2.32. In addition, the shortest distance between a point and a hyperplane is perpendicular to the plane, hence, parallel to \vec{w} . The dotted line r as shown in Figure 2.11 is the translation of the vector $r\vec{w} / \|\vec{w}\|$. The points on the hyperplane closest to \vec{x} are labelled as \vec{x}' then;

$$\vec{x} = \vec{x}' - yr \frac{\vec{w}}{\|\vec{w}\|} \quad 2.33$$

$$\vec{w} \left(\vec{x}' - yr \frac{\vec{w}}{\|\vec{w}\|} \right) + b = 0 \quad 2.34$$

Where, multiplying by y changes the sign for the two cases of \vec{x} being on either side of the decision surface. Moreover, any data point that lies on the decision boundary will satisfy the Equation 2.25.

Further, to solve r , we had the following;

$$r = y \frac{\vec{w}\vec{x} + b}{\|\vec{w}\|} \quad 2.35$$

As in the Figure 2.11 the points closest to the separating hyperplane are support vectors. The geometric margin of the classifier is the maximum width that can be drawn to separate the support vectors of the two classes. That is, the margin is equivalently, the maximum width of a separating hyperplane. The geometric margin is invariant to scaling of parameter (cost) and we can impose any value of a scaling parameter or constraint we wish on \vec{w} without affecting the margin. This helps prevent data point from falling into the margin. For convenience in solving large SVMs, we can choose to require that the functional margin of all data points is at least 1 and that it is equal to 1 for at least one data vector. That is, Equation 2.22 $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$; for all items in the data (Manning et al., 2008; Al Amrani et al., 2018).

Further Manning et al. (2008) and Al Amrani et al. (2018) explains that there exist support vectors for which the inequality is an equality. Each distance from the hyperplane is;

$$r_i = y_i(\vec{w} \cdot \vec{x}_i + b) / \|\vec{w}\| \quad 2.36$$

The Geometric Margin is

$$p = 2 / \|\vec{w}\| \quad 2.37$$

To maximize the Geometric Margin, we want to find \vec{w} and constant b such that Equation 2.37 is maximized. To maximize $2 / \|\vec{w}\|$, we are tasked with minimizing $\|\vec{w}\|$. Which in turn results to;

$$\frac{1}{2} \|\vec{w}\|^2 \quad 2.38$$

Given the constraint in Equation 2.24; the equation that described the boundary or hyperplane, we were again tasked with finding an optimized function, which was subject to the given constraint. Optimization meant finding the minimum and maximum value of our function (Gunal, 2012).

According to Gunal (2012) we can use the Lagrange Multiplier to find the extreme number of the function (maximum or minimum), subject to our constraint in Equation 2.24. This

resulted to a new expression which we could optimize. Achieving the maximum possible margin was the underlying goal of our SVM classifier. Maximization of the margin required the minimization of the training error

$$f(x) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N e_i \quad 2.39$$

In the Equation 2.39, Gunal (2012) explains that C is the user defined constant while e is the margin error. A margin error occurs if data belonging to a particular class are found on the wrong side of the hyperplane. Minimizing the cost was therefore a trade-off issue between a large margin and a small number of training margin errors. The solution of this optimization problem was obtained as shown in Equation 2.40:

$$f(x) = \sum_{i=1}^N \lambda y_i x_i \quad 2.40$$

The Equation 2.40 is the weighted average of the training features; where λ is the Lagrange Multiplier of the optimizing task, as y_i remains the class label. The values of λ are nonzero (+1 or -1) for all of the points lying outside the margin and which appear on either the positive or negative side of the classifier, and exactly zero for those that appear inside or on the margin (support vectors), resulting to an SVM classifier (Gunal, 2012)

2.5.3 Ensemble Learning.

Ensemble Learning are Machine Learning methods that construct a set of predictive models and combine their outputs into a single prediction. Basically, the purpose of learning is typically to achieve better predictive performance, as it has been exhibited in various works such as Sagi and Rokach (2018) and Zhu, Xie, Wang and Yan (2017) where ensembles are shown be more accurate than single models. Sagi and Rokach (2018) gave three fundamental reasons why ensemble methods are able to outperform any single classifier within the ensemble; which include reasons that are statistical, computational and representational related issues. The concept is made simple: train several models from

the same data set, or from samples of the same data set, and combine the output predictions, typically by voting or stacking for classification problems, and averaging output values for estimation problems.

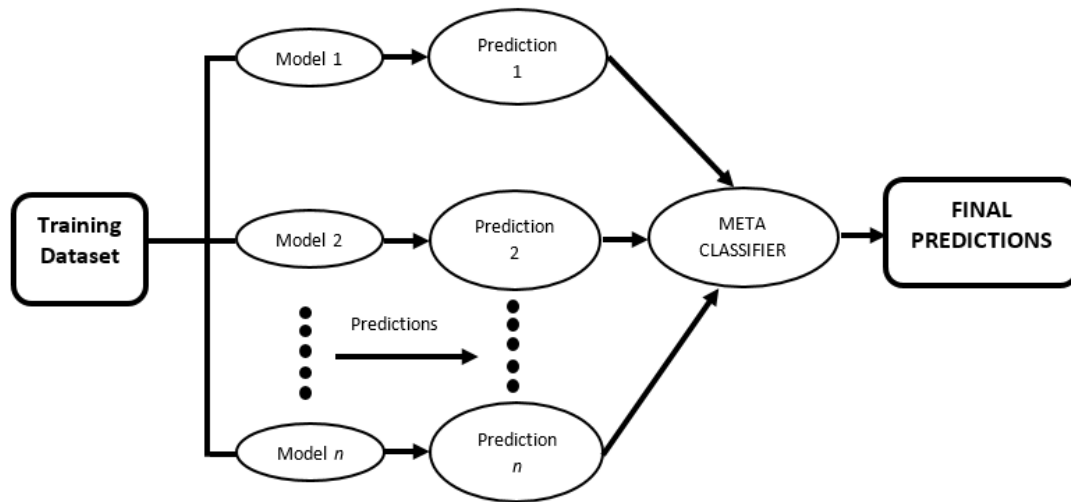


Figure 2.12: Stages of Stacking Algorithm Classification Process

There are various methods of constructing ensembles with different classifiers for data science and analysis. However, just as Wolpert and Macready (1997) puts it, different learning methods can be developed to solve different classification problems. Generally speaking, according to Seijo-Pardo, Porto-Díaz, Bolón-Canedo and Alonso-Betanzos, (2017), there are two kinds of ensembles, homogeneous ensembles and heterogeneous ensembles. Homogeneous ensembles are ensembles where all classifiers are of same type or family, while heterogeneous ensembles are ensembles where the classifiers are of different type or family, that is, diverse (Abuassba, Zhang, Luo, Shaheryar, & Ali, 2017).

Rokach (2005) explains that there are several factors that can be used to differentiate ensemble methods in different dimensions. These include factors to do with Inter-classifier relationship, classifier combining methods, classifier diversity generator and Ensemble size. However, classification can be based on two main dimensions, Inter-classifier relationship and Combining Methods (Araque et al., 2017). Inter-classifier

relationship refer to the methods of achieving learning process, which include Sequential and Concurrent methods (Araque et al., 2017). In Sequential approaches, there is an interaction between the learning runs and it is possible to take advantage of knowledge generated in previous iterations to guide the learning in the next iterations. This approach includes techniques such as Model-guided Instance Selection where the classifiers that were constructed in previous iterations are used for manipulating the training set for the next iteration. Such model-guided algorithms include Boosting, Uncertainty Sampling, among others. Sequential approaches also includes Incremental Batch Learning, where the classifier produced in one iteration is given as “prior knowledge” to the learning algorithm in the following iteration, along with the sub-sample of that iteration. The learning algorithm uses the current sub-sample to evaluate the former classifier, and uses the former one for building the next classifier. The classifier constructed at the last iteration is chosen as the final classifier (Rokach, 2005). On the other hand, in the Concurrent ensemble approaches the original dataset is partitioned into several subsets from which multiple classifiers are induced concurrently. The subsets may be disjoint, meaning mutually exclusive, or overlapping after which a combining procedure is applied in order to produce a single classification for a given instance. Since the method for combining the results of induced classifiers is usually independent of the induction algorithms, it can be used with different inducers at each subset. Concurrent methods aim either at improving the predictive power of classifiers or decreasing the total execution time. Algorithms that can be implemented in this approach include Bagging, Cross-validated Committees among others (Rokach, 2005).

The second dimension Combining Methods include simple multiple classifier combination and Meta combining methods. The simple combining techniques are best suited for classification problems where the individual classifiers perform the same task and have comparable success. However, these combiners are more vulnerable to outliers and to unevenly performing classifiers. The simple combining techniques include techniques such as Uniform Voting where each classifier has the same weight, while a classification of an unlabelled instance is performed according to the class that obtains the

highest number of votes (Rokach, 2005). In addition, in majority voting every classifier makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes (majority). If none of the predictions get more than half of the votes, it can be assumed that the ensemble method could not make a stable prediction for this instance (Demir, 2015). On the other hand, the Meta learning technique means that learning is from the classifiers produced by the base learners (inducers) and from the classifications of these classifiers on training data. This includes techniques such as Stacking and Arbiter Trees (Rokach, 2005).

Rocca (2019) identifies three common ways of creating ensembles from these classifiers or learners. In bagging approach, the original dataset is partitioned into several subsets from which multiple classifiers are concurrently induced. The subsets may be disjoint (mutually exclusive) or overlapping. A combining procedure is then applied in some kind of deterministic averaging process. Bagging approach can be used to decrease the total execution time for classifiers (Rokach, 2005). The second one is boosting, which is also a method for improving the performance of any learning algorithm (Rokach, 2005). The approach learns the weak learners sequentially in a very adaptive way and combines them following a deterministic strategy (Rocca, 2019). According to Rokach (2005) sequential learning can be achieved by repeatedly running a weak learner, on various distributed training data. The classifiers produced by the weak learners are then combined into a single composite strong classifier in order to achieve a higher accuracy. Finally, the third approach is stacking, where meta-learning is done by learning from the classifiers produced by the inducers, and from the classifications of these classifiers on training data. This method tries to induce which classifiers are reliable and which are not. Stacking is usually employed to combine models built by different inducers (Rokach, 2010).

2.5.3.1 Application of Ensemble Learning

Ensemble methods can be used for the creation and building of a more conventional model. For instance, Ensemble methods can also be used to evaluate the relationship and responses in conventional statistical models. As such, Akhtar, Gupta, Ekbal and

Bhattacharyya (2017) presented a cascaded framework of feature selection and classifier ensemble using Particle Swarm Optimization (PSO) for aspect based sentiment analysis, where aspect term extraction and sentiment classification was carried out. The researchers used features that were identified based on the properties of different classifiers and domains. Three classifiers were used as base learning algorithms, namely Maximum Entropy (ME), Conditional Random Field (CRF) and Support Vector Machine (SVM) while identified and implemented various lexical, syntactic or semantic level features for solving their problems. The eligible classifiers identified were combined using either majority or weighted voting. In this study, the ensemble learner was used to find out the most eligible models, that when combined together, maximized some classification quality measure such as F-measure or accuracy.

Additionally, Ensemble learning has also been applicable in areas such as deep learning techniques for Sentiment Analysis to provide automatic feature extraction, rich in both representation capabilities and better performance as compared to traditional feature based techniques in social applications (Araque et al., 2017). The focus of deep learning techniques is to learn complex features extracted from data with minimum external contribution (Bengio, 2009) using deep neural networks (Alpaydin, 2014). Researchers seek to improve performance of deep learning techniques by integrating them with traditional feature based techniques, based on manually extracted features, where ensemble techniques are used to aggregate baseline classifier (a word embedding model and a linear machine learning algorithm) with other traditional classifiers widely used in Sentiment Analysis (Araque et al., 2017). As a result from the research, their statistical study confirmed that the performance of the proposed ensemble models surpassed that of their original baseline models on F1-Score.

In addition, Ensemble learners have also been used to minimize error rate of models in Sentiment Analysis. For instance, Alnashwan, O’Riordan, Sorensen and Hoare (2016) employed a four base learners to investigate the effectiveness of using a combination of existing lexicon resources as meta-level features in ensemble learning for sentiment classification. This offered advantages over using either a single lexicon resource or a

single classifier. Moreover, their experiment showed that, based on a combination of existing lexicon resources, the ensemble learners minimized the error rate by avoiding poor selection from stand-alone classifiers.

Fersini, Messina, and Pozzi (2014), studied Ensemble Learning to reduce the noise sensitivity related to language ambiguity and therefore to provide a more accurate prediction of polarity. The researchers asserted that most of the existing approaches in Sentiment Analysis select the best classification model leading to over-confident decisions that do not take into account the inherent uncertainty of the natural language. They addressed the classifier selection problem by proposing a greedy approach that evaluated the contribution of each model with respect to the ensemble.

2.5.3.2 Contributions of Ensemble Learning

An ensemble learning is a Machine Learning process to get better prediction performance by strategically combining the predictions from multiple learning algorithms (Abuassba et al., 2017; Tuwe, 2015). Further, Sagi and Rokach (2018) states that an ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples. This has truly demonstrated that ensembles are usually significantly more accurate than single learners, as the base classifiers are trained to solve the same original problem but combined to get better results (Araque et al., 2017). Ensemble methods have already achieved great success in many real-world applications (Zhou, 2012).

In addition, according to Sagi and Rokach (2018) the three basic reasons that make it possible to construct good ensembles in practice include statistical reasons, where a learning algorithm or classifier can be viewed as searching a space H of hypotheses to identify the best hypothesis in the space. The statistical problem arises when the amount of training data available is too small compared to the size of the hypothesis space. Without sufficient data, the learning algorithm can find many different hypotheses in H that all give the same accuracy on the training data. By constructing an ensemble out of

all of these accurate classifiers the algorithm can ‘average’ their votes and reduce the risk of choosing the wrong classifier, this can also be referred to as ‘over fitting avoidance’. The second reason is a computational reason where the researcher identified that many learning algorithms or classifiers work by performing some form of local search that may get stuck in local optima. For example, in this study, we demonstrate that classification algorithms such as SVM employ Soft Margin Classification approach where the ‘Cost’ parameter ‘ C ’ which is the cost function, is used to tune the model, then cross validation is carried out to minimize an error function over the training data. In instances where the statistical problem is absent, due to availability of enough training data, it may still be difficult computationally for the learning classifier to find the best hypothesis. An ensemble constructed by running the local search from many different starting points may provide a better approximation to the true unknown function than any of the individual classifiers. Finally, the third reason is representational, which refers to the functions which the model can learn. In most applications of machine learning, the true function f cannot be represented by any of the hypotheses in H . By forming weighted sums of hypotheses drawn from H , it may be possible to expand the space of representable functions. This is a somewhat subtle issue because there are many learning algorithms for which H is, in principle, the space of all possible classifiers (Sagi & Rokach, 2018).

Often times, there are various important factors to consider when creating ensembles such as biasness or variance of base learners (Rocca, 2019). However, Hansen and Salamon (1990) and Sagi and Rokach (2018) points out that accuracy and diversity are some of the key metrics used when developing ensembles. With this, the researchers note that a necessary and sufficient condition for an ensemble of classifiers to perform better than its individual members, is if the individual classifiers are accurate and make independent errors (diverse). Diversity is important while creating ensemble models because it allows stronger learners from different regions to combine their ability to reduce risk of misclassification. While selecting and creating the predictive model in our study, we used classification accuracy – the proportion of correctly classified instances, as one of the metric. An accurate classifier is one that has an error rate of better than random guessing

on new x values. On the other hand, the diversity of the members of an ensemble is known to be an important factor in determining its generalization error (Sagi & Rokach, 2018; Melville, 2003; Rokach, 2010; Moudřík & Neruda, 2015). Two classifiers can also be said to be diverse if they make different errors on new data points, that is, the errors made by the classifiers are uncorrelated (Sagi & Rokach, 2018).

2.5.3.3 Boosting and Bagging Ensemble Methods

Boosting is a sequential ensemble approach that works by repeatedly running weak learners on the training data (Viola & Jones, 2004). Schapire (1990) in his paper described weak learning as a situation where the concept class is weakly learnable if the learner can produce a hypothesis that performs only slightly better than random guessing. The classifiers that were constructed in previous iterations in boosting are then used for manipulating the training set for the next iteration. This makes it possible to take advantage of knowledge generated in previous iterations to guide the learning in the next iterations. Eventually, the classifiers produced by weak learners are combined into a single composite better performing classifier (Sagi & Rokach, 2018; Viola & Jones, 2004).

A weaker model of learnability drops the requirement that the learner be able to achieve arbitrarily high accuracy. Algorithms such as boosting approach can therefore be used to improve the performance of a weak learner, that is, boost the low accuracy of a weak learning algorithm (Schapire, 1990). Due to the reason that it generates a final classifier whose error on the training set is small by combining many hypotheses whose error may be large. In addition, it also produces a combined classifier whose variance is significantly lower than those produced by the weak learner (Rokach, 2005). Furthermore, in boosting, since base classifiers are influenced by the performance of those built previously, the new classifier pays more attention to error that were made in the previous ones and to their performances (Viola & Jones, 2004).

Bagging approach main goal is to improve accuracy by creating an improved composite classifier, by a means of integrating the various outputs of learned classifiers into a single

prediction (Sagi & Rokach, 2018; Rokach, 2005). In comparison to bagging ensemble, which require that learning systems be unstable (Quinlan, 1996; Saugata, 2018). Boosting however, does not require the use of unstable learning systems, provided that their error rate is kept below 50% (Rokach, 2005; Quinlan, 1996). Unstable classifiers are characterized by high Variance (Breiman, Arcing classifiers. , 1998).

Both bagging and boosting reduce bias to some extent, however, their major contribution to accuracy is in the large reduction of variance. As such, boosting (arcing) performs better than bagging because it does better at variance reduction (Breiman, 1998).

A major drawback in boosting is over fitting which may lead to deterioration in performance. This can be brought about by having many learning runs or iterations (Quinlan, 1996). A possible way to avoid this issue of over fitting is by keeping the number of iterations as small as possible, where by, setting iterations too high can lead to overfitting, while setting it too low may result in underfitting. The selection of the most suitable number of iterations is usually done by using a validation set to evaluate the overall predictive performance (Sagi & Rokach, 2018).

2.6 Previous Works and Approaches in Machine Learning to Sentiment detection in Social Media Monitoring - Sentiment Analysis.

Tripathy et al. (2017) considered the problem of classifying documents by overall sentiment that is, determining whether a review was positive or negative. The researchers used a hybrid machine learning approach to perform a document level sentiment analysis by combining two different machine learning algorithms, that is, SVM and ANN, so as to classify the sentiments of review data associated with movie reviews. Using uni-grams SVM, was used to select the best features from the training data. These features were then given as input to ANN method, to process further. Further, they observed that most of authors they reviewed preferred either NB, SVM or combination of them for classification, however, the proposed approach of combining SVM and ANN was found to be better than

results obtained by other authors, as the feature selection process with SVM contributed to creating a more accurate model.

Alayba et al. (2017) studied Twitter as a microblog and introduced a new Arabic dataset for sentiment analysis about health services. The research paper detailed a process of collecting tweets, filtering, a detailed pre-processing of text by removal of unwanted data and some unrelated words, and text normalization. The initial experiments were conducted by utilizing Deep Neural Networks and several other Machine Learning algorithms such as Naïve Bayes, Logistic Regression and Support Vector Machines; while, a combination of “Unigram” and “Bigram” techniques were used for text feature selection. The experiment runs had three phases where different sizes for the training set and the testing set were used. The objective of their research was to investigate the efficiency of utilizing Deep Neural Networks and Machine Learning algorithms in terms of accuracy by using the confusion matrix. On the third phase of experimentation, good results were achieved for accuracy, where, Logistic Regression performed at 86.94%, Multinomial Naive Bayes 90.14%; while Linear Support Vector at 91.37% and Stochastic Gradient Descent performed at 91.87%. The best performing classifiers for the study were SVM using Linear Support Vector Classification and Stochastic Gradient Descent.

Wankhade et al. (2022) discussed sentiment analysis and associated approaches. Their objective was to investigate and complete classification methods with their advantage and disadvantages in sentiment analysis. The researchers stated that supervised machine learning methods were often widely utilized techniques in Sentiment Analysis. Classification using NB and SVM algorithms are commonly used as benchmarks against which newly proposed approaches can be compared.

Drus and Khalid (2019) studied Twitter for Sentiment analysis to offer twitter users a fast and effective way to monitor the publics’ feeling towards their brand, businesses’ among other entities. The researchers showed the method used in analyzing sentiment in social media, and proposed that most common methods uses Lexicon based approaches, while in Machine Learning, Naïve Bayes and SVM are commonly used classifiers. They also

emphasized that choosing the appropriate method for sentiment analysis depends on the data applicable.

Bouazizi and Ohtsuki (2019) studied the task of multi-class sentiment analysis. The researchers analyzed the difficulties of, and the different challenges involved with, multi-class classification, such as opinion object identification, maintaining opinion time, and hidden sentiments identification. They concluded that, even though the task of multi-class analysis is important, it could have been more interesting to perform a sentiment detection task through which all of the sentiments present within a text were to be extracted. The proposed multi-class approach of classification achieved an accuracy of 60.2% for 7 different sentiment classes which, compared to an accuracy of 81.3% for binary classification.

Elango and Narayanan (2014) considered the problem of classifying hotel reviews as 'positive' or 'negative', and thereby analyzing the sentiment of customers. Data used were hotel reviews from TripAdvisor and their finding was that standard machine learning techniques performed better. In the research, they explored various probabilistic models including Naïve Bayes, SVM, Laplace smoothing and semantic orientation to classify the reviews. Extraction of frequent words was done using Term Frequency (TF) and Inverse Document Frequency (IDF) approaches. Further, the accuracy of different strategic models was compared.

With the increase of dependence on technology by societies globally, and the emergence of technologies such as Social networking (Naikoo, Thakur, Guroo, & Lone, 2018), most organizations and individuals are faced with feedback about their companies or processes, from chat, blogs and other opinionated contents such as Twitter. For this reason they are faced with multitudes of decisions to make based on the feedback collected. In this regard, such feedback can be used in Sentiment analysis for Predictions, such as, predictions on market trends on products. Nguyen and Shirai (2015) carried a Prediction research on the Stock Market. Their goal was to develop a model to that would predict a stock price movement using information from social media. Classifiers involved were SVM and

TSLDA (Topic Sentiment Latent Dirichlet Allocation), on the Historical Price and Message Board Datasets.

On the other hand, E-Commerce and other online businesses on the World Wide Web, are popular among customers and other on-line users (Araque et al., 2017). Consequently, large amount of textual data inform of product reviews and comments are posted online by customers and on-line users frequently. These information is very valuable not only to prospective customers who make decisions on buying products, but also for companies who also gather data on customers' satisfaction about their products through Sentiment Analysis (Drus & Khalid, 2019). Dey, Chakraborty, Biswas, Bose and Tiwari (2016) carried out a supervised machine learning research applicable in on-line Commercial ratings, where analysis of hotel and movie reviews toward different entities and products for better services for advertisement, recommendation systems and market trend analysis were carried out. They employed single models using K-NN and Naïve Bayes classifiers.

2.7 Related Work to Unstructured Text Sentiment Analysis on Social Media

According to Vilares et al. (2017) research on Supervised Sentiment analysis in multilingual environments, automatically understanding all the information shared on the Web and transforming it into knowledge is one of the main challenges in this age of Big Data. In terms of NLP (Natural Processing Language), this usually involves comprehending different human languages, which are implicitly related with relevant human aspects such as cultures, countries or regions.

Kataria and Shah (2015) research acknowledged that considering modern writing styles like misspelled words, abbreviations, concatenated words and emoticons can increase the accuracy of sentiment analysis. In addition, they state that, people tend to use different words for a particular feature of a product. Thus identifying frequent nouns and noun phrases automatically will classify more number of reviews, while also help in identifying any new feature of the product that is being talked about.

Most focus currently in sentiment analysis is on online reviews. People express their opinion on social media which consist of product review sites, social networks or blogs such as, Facebook, Amazon, Twitter, Flickr and LinkedIn. Information from these sources are very helpful for the customers to make purchase decisions. According to researchers such as Bhaskar, Sruthi and Nedungadi (2014), Sentiment analysis has become a new knowledge resource after the onset of the Internet and the World Wide Web. Its main purpose is to automatically predict the sentiment polarity of users' opinions on the web. Opinions or preferably, 'Sentiments', play an important role in the understanding of collective sentiments and help to make better decisions. The sentiments can be 'positive', 'negative' or even 'neutral'. Positive sentiments encourage the prospective customer to make positive decisions toward a product; negative sentiments may usually result to negative decisions. None the less, Sentiment analysis of textual communication extracts subjective information in the text.

Bhaskar et al. (2014) carried a research on enhanced sentiment analysis of informal textual communication in social media by considering objective words and intensifiers. Their aim was to improve the sentiment classification by modifying the sentiment values returned by SentiWordNet for intensifiers based on the context to the semantic of the words related to the intensifiers. They also reassigned some of the objective words to either 'positive' or 'negative' sentiment. The researchers tested their sentiment classification method with product reviews of digital cameras gathered from Amazon and e-bay, and showed that the method improved the prediction accuracy.

Abirami, Uma and Prakash (2016) paper proposed an approach to sentiment analysis by addressing three major issues in analyzing social media content; computing the intensity of a polarity, identifying the more effective sentiment in such cases where various sentiments are mentioned, context dependency of opinion words and deliberate spelling errors. The researchers acknowledges that there is an increase in interest in developing improved opinion mining algorithms for accuracy, and developing a more efficient understanding of the dynamics of the human sentiment. Their task was a classification ranking problem with a goal of producing mathematical score for the sentiments observed.

Using Twitter, the idea took into consideration every aspect of the text and used that information to compute final score. The aspects included social media tendencies such as use of emoticons, punctuations, casing, vowel stretches, abbreviations, interjections and the context in which the opinions are mentioned. Evaluation of the algorithm was then done by comparing the result to a baseline produced by manual scoring.

Vilares et al. (2017) tackles the problem of performing multilingual polarity classification on Twitter, comparing three techniques: First, a multilingual model trained on a multilingual dataset, obtained by fusing existing monolingual resources that did not need any language recognition step. Secondly, a dual monolingual model with perfect language detection on monolingual texts, and finally, a monolingual model that acted based on the decision provided by a language identification tool. The techniques were evaluated on monolingual, synthetic multilingual and code-switching corpora of English and Spanish tweets, their goal being to compare the performance of supervised monolingual models based on bag-of-words, with respect to their corresponding multilingual version; a model that is a collection of weights from English and Spanish features. To do this, they relied on standard sets of features. The aim was to show how current state of the art supervised approaches can successfully address or not address, situations where monolingual, multilingual and code-switching texts appear. They also relied on an L2-regularized logistic regression.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter will describe the methodology on data collection, data analysis, data sources and design of the study. The main purpose of the study was to create an ensemble model for Sentiment Analysis using Naïve Bayes and Support Vector classifiers. However, for this chapter, the goal was to create a basic workflow for conducting Sentiment Analysis for the product reviews from Twitter data.

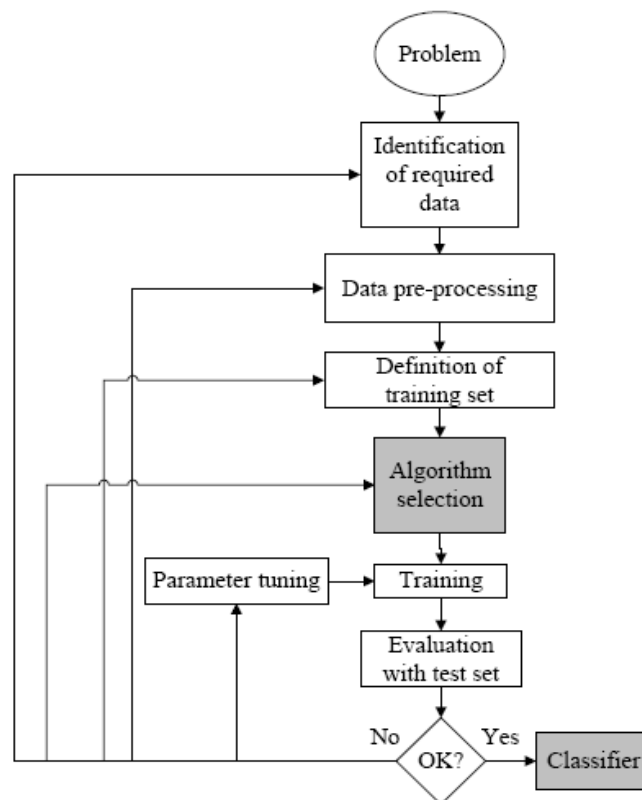


Figure 3.1: The Processes of Supervised Machine Learning

Source: (Osisanwo, et al., 2017)

3.2 Sentiment Analysis Process in R-Studio

To carry out the Sentiment Analysis process in R-Studio, a methodology for the Sentiment analysis model using Naïve Bayes and Support Vector classifiers was created, with an intension of evaluating the performance of the proposed classifiers, investigating their universal reliability and finally, proposing recommendations on how performance improvement could be achieved.

The implementation of the methodology was then achieved in R-Studio environment as elaborated in Chapter 2.3.1 and this following Chapter. To achieve a supportive infrastructure of the needs of the research, the following activities from Chapter 3.2.1 were carried out as depicted in Figure 3.1.

3.2.1 Data Sources and Collection

The datasets inform of tweets, were generated by collecting and mining 38,700 text documents from a social media platform, Twitter, using a twitter API as indicated in Figure 3.2. The tweets were mined in intervals from June 2019 to July 2022. These tweets, were expected to address general topics touching on different product reviews and confidence level measure for "OnePlus 7 Pro" mobile phone, while being categorized as 'positive' and 'negative'.

	tweet	favorited	favorite_replyToSN	created	truncated	replyToSID	replyToID	statusSource	screenName	retweetCount	isRetweet	retweets	logGrade	latitud
1	1 RT	FALSE	0 NA	28/07/2022 06:31	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_rishiilunwar		39	TRUE	FALSE	NA	NA
2	2 @OnePlus	FALSE	0 OnePlus_IN	28/07/2022 06:31	TRUE	1.55E+18	1.55254E+18 1.14E+09	<a href="https://twitter.com_aakashibakshi		0	FALSE	FALSE	NA	NA
3	3 RT @One	FALSE	0 NA	28/07/2022 06:31	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_houdoyoubo		27	TRUE	FALSE	NA	NA
4	4 @stuffint	FALSE	0 stuffintings	28/07/2022 06:31	FALSE	1.55E+18	1.55254E+18 2.9E+09	<a href="https://twitter.com_ajayyadkumar		0	FALSE	FALSE	NA	NA
5	5 OnePlus I	FALSE	0 NA	28/07/2022 06:30	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_@Industannarus		0	FALSE	FALSE	NA	NA
6	6 RT	FALSE	0 NA	28/07/2022 06:30	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_AbdN10n		39	TRUE	FALSE	NA	NA
7	7 @Abhishek	FALSE	0 Abhishekmishra28	28/07/2022 06:30	TRUE	1.55E+18	1.55254E+18 1.05E+08	<a href="https://twitter.com_prabimatha		0	FALSE	FALSE	NA	NA
8	8 RT	FALSE	0 NA	28/07/2022 06:29	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_SoyeedYasinHani		5	TRUE	FALSE	NA	NA
9	9 RT	FALSE	0 NA	28/07/2022 06:29	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_ajayyadkumar		39	TRUE	FALSE	NA	NA
10	10 RT	FALSE	0 NA	28/07/2022 06:29	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_navy02_		39	TRUE	FALSE	NA	NA
11	11 RT	FALSE	0 NA	28/07/2022 06:29	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_Ananisharma_08		39	TRUE	FALSE	NA	NA
12	12 While the	FALSE	0 NA	28/07/2022 06:29	TRUE	NA	1.55254E+18 NA	<a href="https://twitter.com_TrendGindl		0	FALSE	FALSE	NA	NA
13	13 RT @One	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_Priyansu7002986		147	TRUE	FALSE	NA	NA
14	14 RT	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_Ashwin98483291		576	TRUE	FALSE	NA	NA
15	15 RT	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_TechQor24		100	TRUE	FALSE	NA	NA
16	16 RT @ufc:	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_Tyrooyr		21	TRUE	FALSE	NA	NA
17	17 RT	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_TechQor26		91	TRUE	FALSE	NA	NA
18	18 RT @stuff	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_TechQor24		233	TRUE	FALSE	NA	NA
19	19 RT	FALSE	0 NA	28/07/2022 06:28	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_TechQor24		39	TRUE	FALSE	NA	NA
20	20 RT	FALSE	0 NA	28/07/2022 06:27	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_Rishiku_0118		39	TRUE	FALSE	NA	NA
21	21 RT	FALSE	0 NA	28/07/2022 06:25	FALSE	NA	1.55254E+18 NA	<a href="https://twitter.com_sushilboora2		576	TRUE	FALSE	NA	NA

Figure 3.2: A representation of the mined tweets

3.2.2 Setting Working Directory, Install and Load Libraries (Packages) In R.

When working with R-Studio, it is always important to set the working directory because this is where all the charts, graphs, plot images, csv files and other R project files or outputs are stored. This is to ensure that the working environment is well organized. The directory file path and location is normally also set.

R requires installation of various packages which are collections of R functions, data, and compiled code in a well-defined format, that a user installs using the `install.packages()` function, as desired. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, the packages have to be loaded into the session to be used with the function `library()`.

3.2.3 Authorize Access for Twitter API

An API - Application Programming Interface is a software intermediary that allows two softwares or applications to interact. Twitter provides companies, developers and users such as researchers, with programmatic access to twitter data through its APIs. According to Strickland and Chandler (2017) Twitter bases its APIs off the Representational State Transfer (REST) architecture. REST architecture refers to a collection of network design principles that define resources and ways to address and access data. The architecture is a design philosophy, not a set of blueprints, meaning that there is no single prescribed arrangement of computers, servers and cables. Twitter is compatible with two syndication formats used on the Web; Really Simple Syndication (RSS) and Atom Syndication Format (Atom) that retrieve data from one resource and send it to another, thus, allowing third-party developers, and also researchers, partial access to its API.

For the purpose of this study, an API ‘*achieng_rap*’ was created with the aim of interacting with Twitter, to mine data necessary for the study. This was fundamental because Twitter’s API can be used as a Streaming API, to get live statuses or tweets as they are sent, by creating a ‘tweet listener’. Twitter’s Search API is able to give access to a data set that already exists from tweets that have occurred. The API is able to capture large amounts of data, and finally be useful for in house archives such as archiving social discussion about product brand(s).

3.2.4 Tweet Harvesting and Data Mining.

Tweet harvesting was carried out by using the Twitter’s Search API and the *searchTwitter()* function, which involved pulling Twitter’s data through a search or username. The API gave access to a data set that already existed from tweets that had occurred. Through the Search API, requests of tweets that matched some sort of ‘search’ criteria was made. The criteria can be keywords, usernames, locations, and named places, among others. For experiment purposes, we used ‘OnePlus Mobile’ as the key search word to search and mine reviews about OnePlus mobile phone products. Specifications on

language, for instance English, and the number of latest tweets such as 3000 tweet reviews from a specific date, were set. These tweets, were expected to address general topics touching on different product reviews and confidence level measure, while being categorized as ‘positive’ and ‘negative’.

3.2.5 Creating a Data Frame

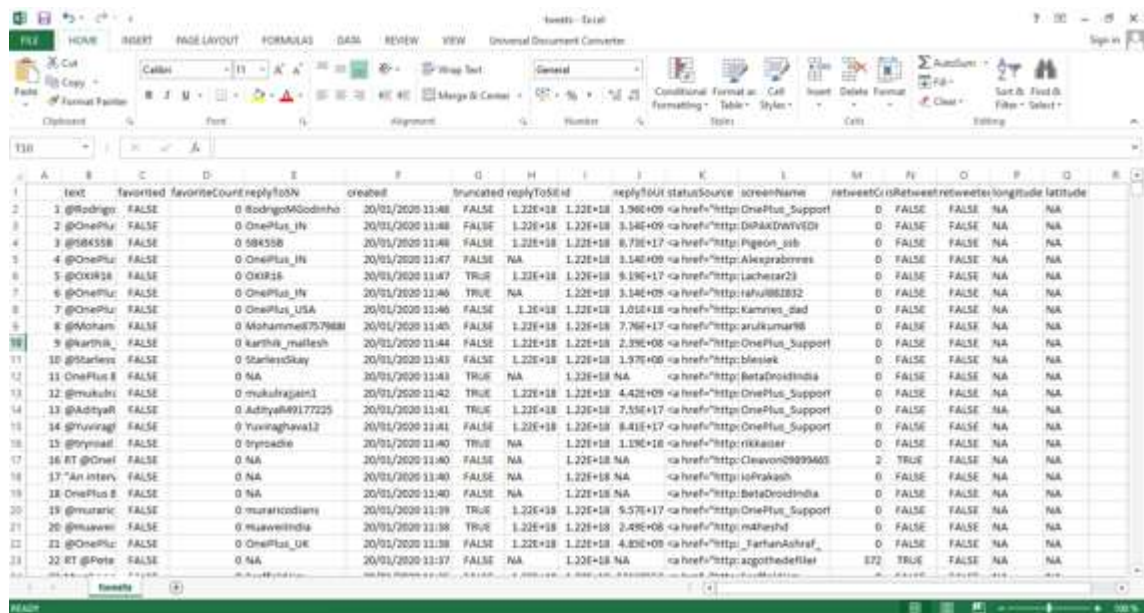
The data of tweets mined were in character string format, and needed to be converted to a data frame in R environment with the *ldply()* function for analysis. Data frames provide a way of grouping a number of related variables into a single data object. The function *DataFrame()* takes a number of vectors and, or factors to returns a single object containing all the variables. Creating a data frame was important because, converting the extracted data to a data frame made the data more readable and easier to work with in the analysis environment, since the data was also organized into rows and columns and saved in a *csv* file format as output – ‘*tweets.csv*’. The raw dataset included a 16 feature variable as shown in Figure 3.3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	text	favorited	favoriteCount	replyToSM	created	truncated	replyToURL	replyToUID	statusSource	screenName	retweetCount	isRetweet	retweeted	longitude	latitude		
2	1	@Rodrigo	FALSE	0	RodrigoMGo	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	1961474503	OnePlus_Sup	B	FALSE	FALSE	NA	NA	
3	2	@OnePlus	FALSE	0	OnePlus_IN	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	3138727082	D9AKDWVVEI	D	FALSE	FALSE	NA	NA	
4	3	@S6K558	FALSE	0	S6K558	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	8.75127E+17	Pigeon_csb	D	FALSE	FALSE	NA	NA	

Figure 3.3: The 16-feature variables raw dataset

3.2.6 Pre-processing and Data Preparation

Once Twitter data mining and collection was complete, only the ‘text’ was stripped from the entire raw data, and cleaned through a clearly defined pre-processing techniques. This was because as shown in Figure 3.3 and Figure 3.4, the raw data contained some unnecessary data that attributed to noise in the dataset. The raw data was a collection of various feature variables in a *csv* file format data frame. To analyse the data, the main focus was the variable ‘text’ that contained tweets of the product reviews that were textual in nature. To accomplish this, it was necessary that only the text was stripped from the entire data with the function *getText()* in R environment, which was applied to the whole raw dataset as shown in Figure 3.5.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		text	favorited	favoriteCount	replyToSN	created	truncated	replyToSID	replyToURL	statusSource	screenName	retweetCount	retweeted	retweetedBy	longitude	latitude		
2	1	@Rohitgo	FALSE	0	@Rohitgo	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	1.96E+09	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
3	2	@OnePlus	FALSE	0	OnePlus_IN	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	3.14E+09	<a href="http://DiPAKDrVVEDI	0	FALSE	FALSE	NA	NA		
4	3	@SBR4558	FALSE	0	SBR4558	20/01/2020 11:48	FALSE	1.22E+18	1.22E+18	8.73E+17	<a href="http://Pigeon_ssb	0	FALSE	FALSE	NA	NA		
5	4	@OnePlus	FALSE	0	OnePlus_IN	20/01/2020 11:47	FALSE	NA	1.22E+18	3.14E+09	<a href="http://Akesrabrnm	0	FALSE	FALSE	NA	NA		
6	5	@OxIR18	FALSE	0	OxIR18	20/01/2020 11:47	TRUE	1.22E+18	1.22E+18	9.19E+17	<a href="http://lachezar2	0	FALSE	FALSE	NA	NA		
7	6	@OnePlus	FALSE	0	OnePlus_IN	20/01/2020 11:46	TRUE	NA	1.22E+18	3.14E+09	<a href="http://rahul882832	0	FALSE	FALSE	NA	NA		
8	7	@OnePlus	FALSE	0	OnePlus_USA	20/01/2020 11:46	FALSE	1.2E+18	1.22E+18	1.01E+18	<a href="http://Kamrns_bad	0	FALSE	FALSE	NA	NA		
9	8	@Moham	FALSE	0	Mohammed757988	20/01/2020 11:45	FALSE	1.22E+18	1.22E+18	7.76E+17	<a href="http://arukumar96	0	FALSE	FALSE	NA	NA		
10	9	@karthi	FALSE	0	karthi_mallesh	20/01/2020 11:44	FALSE	1.22E+18	1.22E+18	2.39E+08	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
11	10	@Starless	FALSE	0	StarlessKay	20/01/2020 11:43	FALSE	1.22E+18	1.22E+18	1.97E+09	<a href="http://blesiak	0	FALSE	FALSE	NA	NA		
12	11	OnePlus E	FALSE	0	NA	20/01/2020 11:43	TRUE	NA	1.22E+18	NA	<a href="http://BetaOndIndia	0	FALSE	FALSE	NA	NA		
13	12	@mukabhi	FALSE	0	mukabhi	20/01/2020 11:42	TRUE	1.22E+18	1.22E+18	4.42E+09	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
14	13	@AdityaR	FALSE	0	AdityaR9177225	20/01/2020 11:41	TRUE	1.22E+18	1.22E+18	7.53E+17	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
15	14	@Yuvrajag	FALSE	0	YuvrajYuvraj2	20/01/2020 11:41	FALSE	1.22E+18	1.22E+18	8.41E+17	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
16	15	@ryyinal	FALSE	0	ryyinal	20/01/2020 11:40	TRUE	NA	1.22E+18	1.13E+18	<a href="http://rikkaiser	0	FALSE	FALSE	NA	NA		
17	16	RT @OneP	FALSE	0	NA	20/01/2020 11:40	FALSE	NA	1.22E+18	NA	<a href="http://Cleaver09099605	2	TRUE	FALSE	NA	NA		
18	17	"An interv	FALSE	0	NA	20/01/2020 11:40	FALSE	NA	1.22E+18	NA	<a href="http://joPrakash	0	FALSE	FALSE	NA	NA		
19	18	OnePlus E	FALSE	0	NA	20/01/2020 11:40	FALSE	NA	1.22E+18	NA	<a href="http://BetaOndIndia	0	FALSE	FALSE	NA	NA		
20	19	@muralic	FALSE	0	muralicodians	20/01/2020 11:39	TRUE	1.22E+18	1.22E+18	9.57E+17	<a href="http://OnePlus_Support	0	FALSE	FALSE	NA	NA		
21	20	@muawwi	FALSE	0	muawwiIndia	20/01/2020 11:38	TRUE	1.22E+18	1.22E+18	2.49E+08	<a href="http://malleshj	0	FALSE	FALSE	NA	NA		
22	21	@OnePlus	FALSE	0	OnePlus_UK	20/01/2020 11:38	FALSE	1.22E+18	1.22E+18	4.82E+08	<a href="http://FarhanAshraf	0	FALSE	FALSE	NA	NA		
23	22	RT @Pete	FALSE	0	NA	20/01/2020 11:37	FALSE	NA	1.22E+18	NA	<a href="http://agthedefties	872	TRUE	FALSE	NA	NA		

Figure 3.4: The raw data feature variables in a *csv* file format data frame

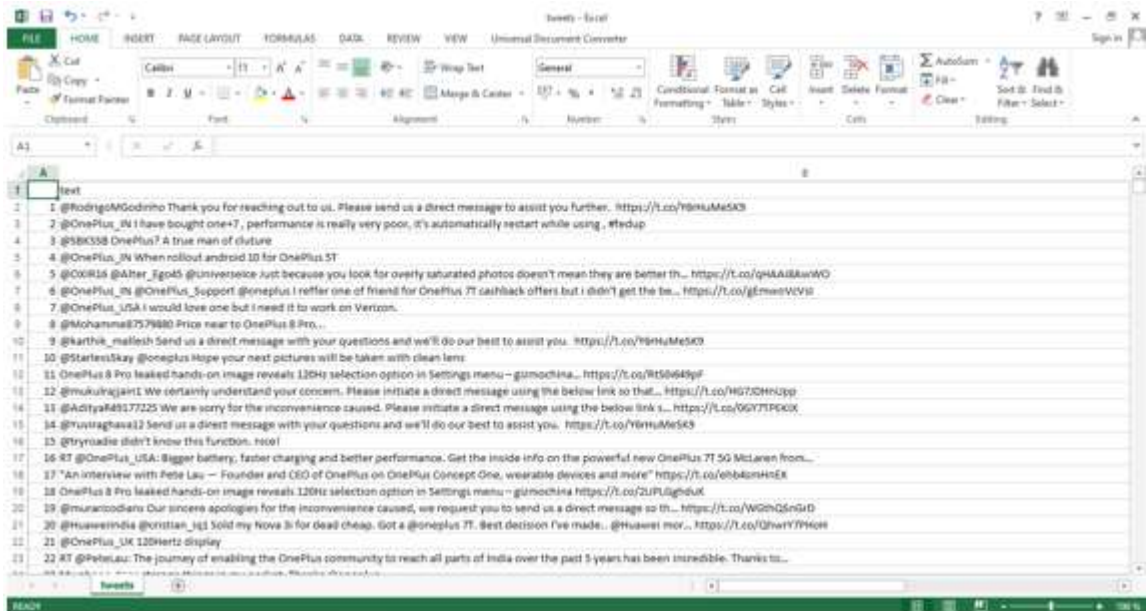


Figure 3.5: The stripped text data in a csv file format

Machine learning is one of the widely used approaches towards sentiment classification in addition to lexicon based methods and linguistic methods (Thelwall, Buckley, & Paltoglou, 2011). According to Iliou, Anagnostopoulos, Nerantzaki and Anastassopoulos (2015), data pre-processing describes any type of processing methods or data preparations that are performed on raw data to prepare it for another processing procedure, and analysis. Commonly used as a preliminary data mining practice, data pre-processing methods transforms the data into a format that can be easily and effectively used for the classification algorithms. These processes are done after the text is ‘stripped’ to enhance performances of the classifiers and facilitate feature extraction (Asghar, Khan, Ahmad, & Kundi, 2014). Haddi, Liu and Shi (2013) state that pre-processing the data is therefore the process of cleaning, that is, removal of data that are not meaningful for the analysis; and preparing the text for classification.

Zin, Mustapha, Murad and Sharef (2017) further states that pre-processing phase in Sentiment analysis is the process of cleaning up the dataset from any noisy data or irrelevant features, thus reducing the complexity of a document in order to prepare the

data for the classification task. If all words or texts were plotted in their raw format, then the text analytics was not going to be accurate, since the analysis would include the noisy data. Noise may be due to typographic errors or colloquialisms always present in natural language or text, which usually result from their unstructured characteristic, eventually lowering the data quality in a way that makes the text less accessible to classifiers and automated processing.

Studies by Haddi et al. (2013) and Zin et al. (2017) experimental results showed that appropriate text pre-processing methods, extraction and selection of features could significantly enhance the algorithm or classifier's performance. Pre-processing techniques will reduce the number of words in reviews and the number of words in vocabulary significantly, thereby improving computational costs. In light of these results, the pre-processing phase was important to help us clean and remove any unnecessary words that might have hindered effective computational processes and analysis as large numbers of features, or inclusion of all the text would have increased the computational cost for the entire analysis, since irrelevant features would have been plotted as features. Schrauwen (2010) states that too many features can cause an algorithm to have a higher chance of relying on idiosyncrasies of the training data that do not generalize well to new examples.

For the purpose of this study, the performance of the classifiers were critically examined based on the results dependent on performance measure of precision, accuracy, and robustness. In order to transform a document into a feature vector, pre-processing was therefore critically needed. Pre-processing was done using the *gsub()* function in R environment. As such, the procedure included;

3.2.6.1 Removing RT retweet texts

This involved removal of retweets (RT) and user names for twitter users who were retweeting, from the text. For instance, in the retweet “RT @OnePlus_USA: Bigger battery, faster charging and better performance. Get the inside info on the powerful new

OnePlus 7T 5G McLaren from...” ‘RT @OnePlus_USA’ was removed. This was because this kind of data was not meaningful for analysis.

3.2.6.2 Removing html links

This activity involved removal of *http* references and links from the text. For example, “https://t.co/LC3fmt4Q6L” was removed because it was not required in the text and not meaningful for the analysis.

3.2.6.3 Removing Twitter users’ names

This was done to remove user names of Twitter sphere users’ such as ‘@muraricodians’.

3.2.6.4 Removal of punctuation marks

This activity was done to remove all the punctuation marks from the text. The process was also used to remove other words that were not desired in the text.

3.2.6.5 Removal of all numbers

This activity was done to remove all the numbers from the text, such as 1, 2, 3, 4, 5, 6, 105 etc.

Finally, the partially clean text dataset was then exported to excel and stored as a *csv* file output ‘tweets1.csv’ as demonstrated in Figure 3.6.

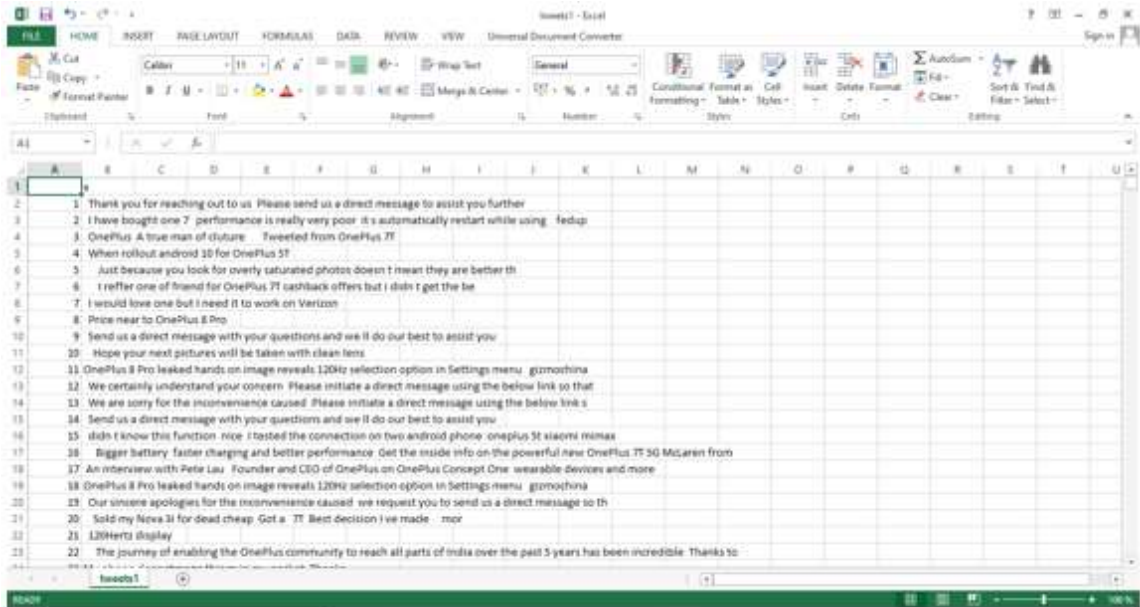


Figure 3.6: Partially clean text data

In an attempt to further extract relevant features, we transformed arbitrary data, the text, into numerical features usable for machine learning. This process involved selection and extraction of subset features into new space, which involved creating a bag-of-word corpus, build a Document-Term Matrix and create N-Gram Tokenizer function.

3.2.7 Creating a Bag-of-Word Corpus

To create the bag-of-word corpus, the individual words or text were taken into account. Bag-of-word is a feature vector representation where each dimension of text corresponds to a feature. The assumption was that all features were independent given the class labels. In this model, texts were represented as a bag of its words, disregarding grammar, context and even word order but keeping multiplicity. The occurrence of each word was used as a feature for training the classifiers (RamaKrishna, Rao, Rao, & Chandan, 2015). This was suitable for the informal or colloquial content in the text.

The bag-of-word model learned a vocabulary from all of the documents, then modeled each document by counting the number of times each word appeared by implementing a

Document Term Matrix (DTM) process. For instance, for demonstrative purposes, we considered the following two sentences:

Sentence 1: “The Samsung foldable phone reinventing the mobile experience”

Sentence 2: “Samsung finally unveils foldable smartphone”

From these two sentences, our vocabulary was as follows:

{the, samsung, foldable, phone, reinventing, mobile, experience, finally, unveils, smartphone}

To get our bags of words, we counted the number of times each word occurred in each sentence; such that in Sentence 1, "the" appears twice, and “Samsung”, “foldable”, “phone”, “reinventing”, “mobile”, “experience” each appear once, so, below is the procedure that was used for determining the feature vectors for Sentence 1:

Vocabularies:

{the, samsung, foldable, phone, reinventing, mobile, experience, finally, unveils, smartphone}

Sentence 1: “the samsung foldable phone reinventing the mobile experience”

DTM for the above corpus would have the following, as also shown in Table 3.1;

features: {2, 1, 1, 1, 1, 1, 1, 0, 0, 0}

Sentence 2: “samsung finally unveils foldable smartphone”

Similarly, the features for *Sentence 2* were; *{0, 1, 0, 0, 0, 0, 0, 1, 1, 1}*

Table 3.1: Document Term Matrix (DTM) for the sentences

	the	samsung	foldable	phone	reinventing	mobile	experience	finally	unveils	smartphone
S 1	2	1	1	1	1	1	1	0	0	0
S 2	0	1	0	0	0	0	0	1	1	1

Further cleaning activities were also carried out on the corpus, which included transforming the entire text to lower case, removal of stop words, stripping white space and stemming the document.

3.2.7.1 Transform texts to lower case

This was a procedure that converted the entire text in the corpus to lower case. This was done because tweets are highly unstructured, while reviewers may capture certain texts as capital letter in order to maybe lay emphasis on the text. For uniformity, the texts were transformed to lower case.

3.2.7.2 Removal of Stop words

Stop words are commonly used words such as ‘me’, ‘a’, ‘the’, ‘who’, ‘them’, ‘shall’, ‘has’, ‘have’, among others, that are not meaningful for the analysis. They do not portray personal sentiments and are merely connecting words. The words were removed and ignored because they were so common that including them would greatly increase the size of the index without improving performance of the classifiers. This process was basically done so as to improve feature extraction as it was expected that this would lead to a significant positive impact on classifier accuracy. In addition, R Studio comes with a stop word corpus that includes a list of English stop words.

3.2.7.3 Strip white space

There were elements of white spaces in the corpus of text. This was difficult for R Studio to comprehend, thus, it was important to strip the white spaces to once again enhance the classifier performance.

3.2.7.4 Stemming

Stemming is the process of reducing a word to its word stem or parent word that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is part of information extraction, hence feature extraction, a process of linguistic normalization, in which the variant forms of a word are reduced to a common form, for example;

connection

connections

connective ---> connect

connected

connecting

Stemming was also done with the intension of improving classifier performance.

3.2.8 Build a Document Term Matrix

A document-term matrix or term-document matrix is a two dimensional mathematical matrix that describes the frequency/ occurrence of terms or word that occur in a collection of documents. In a document-term matrix, rows correspond to the terms and columns correspond to the documents, so each entry (i, j) represents the frequency of term ' i ' in document ' j '. For each entry in the matrix, the term frequency measures the number of

times that term i appears in document j , and the inverse document frequency measures the number of documents in the corpus which contain term i (Dalis, 2014).

Document Term Matrix function was also used to remove sparse terms by ignoring terms that had a document frequency lower than the given threshold ($min=1, max=1$) to help in generalization and prevent over fitting. This was used to eliminate words with very low frequency and was done by determining the least number of times a word appeared in a document or entry. The ' min/max ' was set to '1' to set the terms in unigram or n-gram format.

Sparse words were also removed using the *removeSparseTerms()* function in R environment, this means that, low frequency words were removed. Sparsity refers to the proportion of cells with 0's. This reduced the dimension of the Document Term Matrix.

3.2.8.1 Creating a N-gram Tokenizer Function

According to Schmidt and Heckendorf, (2017) an n-gram is an ordered sequence of n "words" taken from a body of text. In natural language processing, tokenization is the process of breaking human-readable text into machine readable components. The most obvious way to tokenize a text is to split the text into words.

We then used the *NgramTokenizer()* function to allow us specify our words in the Document Term Matrix as unigrams.

3.2.8.2 Word Frequencies and Selection of Feature Vectors

Adequate feature selection techniques in Sentiment analysis have significant impact for identifying relevant features, while also increasing classification accuracy (Asghar et al., 2014; Konez & Paralic, 2011).

Statistical feature selection methods are one of categories in feature selection methods that can be applicable in feature selection. These techniques are further divided into sub-types

such as univariate, multivariate and hybrid. Univariate methods, which are also referred to as Feature filtering methods, take attributes separately, and includes techniques such as Occurrence Frequency and Minimum Frequency thresholds (Asghar et al., 2014). Univariate methods have computational efficiency, however, they ignore attribute interactions (Asghar et al., 2014; Konez & Paralic, 2011).

Saraee and Bagheri (2013) also indicate that Feature Selection methods sort features on the basis of a numerical measure computed from the documents in the dataset collection, and select a subset of the features by thresholding that measure. Various information measures such as Document Frequency (DF), Term Frequency Variance (TFV) and Mutual Information (MI) among others can also be implemented for Feature selection in sentiment analysis.

Word Frequencies is a filtering and weighting process. The term-frequency is used as a counting function to return how many times the term ' t ' or word, is present in the document. The most frequent words are then used as feature vectors, basing on their scores. This is demonstrated in Figure 3.7 and Figure 3.8. A feature vector is just a vector that contains information describing an object's important characteristics (Girard, 2015). For the purpose of this study, we employed an Occurrence Frequency technique, a Document-Term Matrix (Term-Document Matrix). The selection was basically a process for filtering irrelevant or redundant features from the dataset. A measure of term-frequency was then used as a counting function to return how many times the term t (word) was present in the documents.

	word	freq
oneplus	oneplus	1803
pro	pro	1663
120hz	120hz	342
get	get	229
will	will	223
display	display	220
phone	phone	193
now	now	156
compani	compani	149
geekbench	geekbench	145
new	new	143
screen	screen	134
certain	certain	131
wqhd	wqhd	131
almost	almost	130
man	man	128
adopt	adopt	127
young	young	126
aggress	aggress	125
sam	sam	125
camera	camera	117
can	can	106
one	one	101
ram	ram	100
samsung	samsung	99

Figure 3.7: Word Frequencies

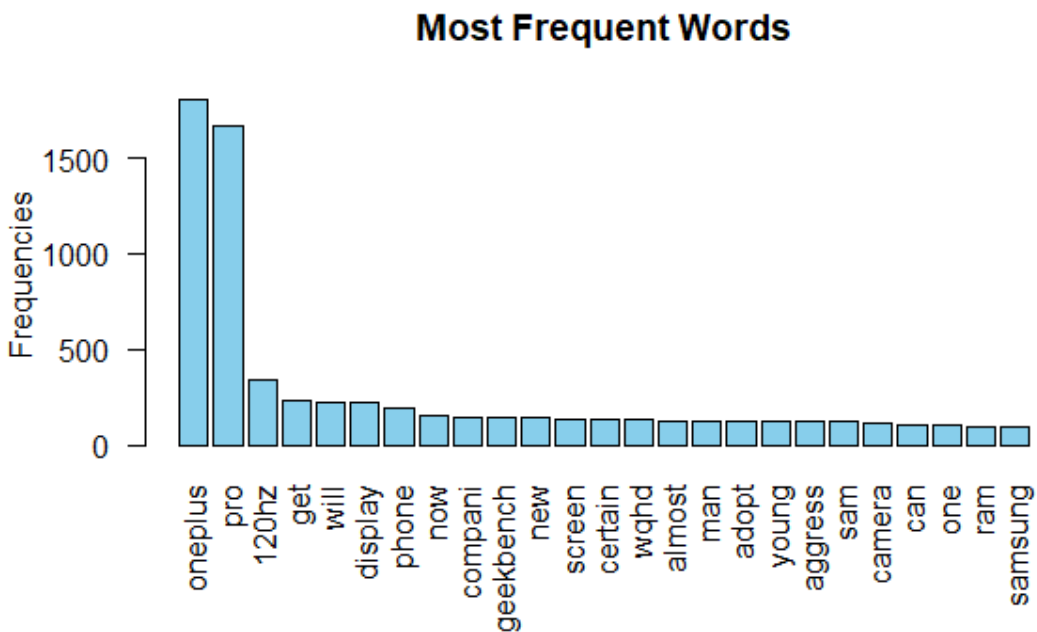


Figure 3.8: A graphical presentation of Word Frequencies

3.2.9 Analysis for the Tweets

According to Ahmad et al. (2017) and Alayba et al. (2017) one approach to Sentiment analysis is to use a lexicon with information about which words are *positive* or *negative*. The lexicon is used to assign each word a sentiment (positive or negative). This method uses a variety of words annotated by polarity score, to decide the general assessment score of a given content. The lexicon used in this study were acquired automatically in RStudio environment from the *tidytext* package, as shown in Figure 3.9.

3.2.9.1 Lexicon Based Sentiment Classification Process

At this point of the study, text data were already tokenized by individual words. When human readers approach a text, we normally use our understanding of the emotional intent of words to infer as to whether a section of text is ‘positive’ or ‘negative’, or perhaps characterized by some other more nuanced emotion like surprise or disgust. To accomplish this analysis in R environment, we used the *tidytext* package that contains some useful tools (Silge and Robinson, 2019).

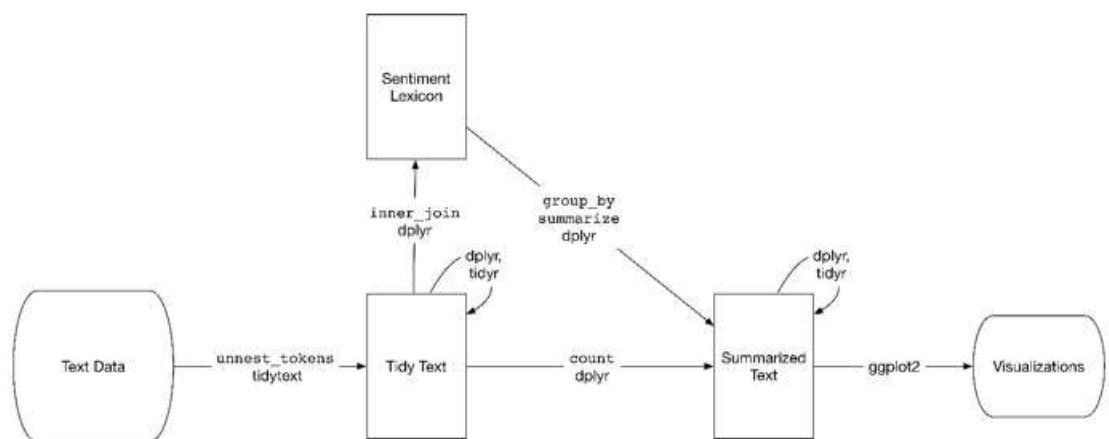


Figure 3.9: A flowchart of a typical text analysis that uses tidy data principles with *tidytext* package

Source: (Silge & Robinson, 2019)

Lexicons can either be manually compiled or acquired automatically. The annotation of lexica or corpora is usually done by hand, and classifiers are then trained with large sets of features to classify a new batch of words or phrases. However, according to Silge and Robinson (2019) there are a variety of methods and dictionaries that exist for evaluating the opinion or emotion in text, in R environment. For this reason, the *tidytext* package contains several sentiment lexicons in the sentiments Dataset. The three general-purpose lexicons in R environment are *AFINN* from (Nielsen, 2011), *bing* from (Ding, Liu, & Yu, 2008) and (Hu & Liu, 2004), and *nrc* from (Mohammad & Turney, 2013) as shown in Figure 3.10 to Figure 3.12. All three of these lexicons are based on unigrams, that is, single words. These lexicons contain many English words and the words are assigned scores for ‘positive’ or ‘negative’ sentiment, and also possibly emotions like joy, anger, sadness among others.

The *nrc* lexicon categorizes words or text in a binary fashion into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust (Mohammad & Turney, 2013). The *bing* lexicon also categorizes words in a binary fashion into positive and negative categories (Ding et al., 2008). The *AFINN* lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment (Nielsen, 2011). All of this information is tabulated in the sentiments dataset, and *tidytext* provides a function `get_sentiments()` to get specific sentiment lexicons without the columns that are not used in that lexicon. Silge and Robinson (2019) states that the sentiment lexicons were constructed via either crowdsourcing or manually by the labour of one of the authors, and were validated using some combination of crowdsourcing again, restaurant or movie reviews, or Twitter data.


```

> #afinn
> get_sentiments("afinn")
# A tibble: 2,477 x 2
  word      value
  <chr>    <dbl>
1 abandon      -2
2 abandoned    -2
3 abandons     -2
4 abducted     -2
5 abduction    -2
6 abductions   -2
7 abhor        -3
8 abhorred     -3
9 abhorrent    -3
10 abhors       -3
# ... with 2,467 more rows

```

Figure 3.10: The *AFINN* Lexicon

```

> #bing
> get_sentiments("bing")
# A tibble: 6,786 x 2
  word      sentiment
  <chr>    <chr>
1 2-faces  negative
2 abnormal negative
3 abolish negative
4 abominable negative
5 abominably negative
6 abominate negative
7 abomination negative
8 abort    negative
9 aborted  negative
10 aborts  negative
# ... with 6,776 more rows

```

Figure 3.11: The *bing* Lexicon

```

> #nrc
> get_sentiments("nrc")
# A tibble: 13,901 x 2
  word      sentiment
  <chr>     <chr>
1 abacus    trust
2 abandon  fear
3 abandon  negative
4 abandon  sadness
5 abandoned anger
6 abandoned fear
7 abandoned negative
8 abandoned sadness
9 abandonment anger
10 abandonment fear
# ... with 13,891 more rows

```

Figure 3.12: The *nrc* Lexicon

We then used the *syuzhet* package in R environment to incorporate the three sentiment lexicons *AFINN*, *bing* and *nrc* with a goal to introduce the main functions in the package so that one could quickly extract plots and sentiment data from the text files. Thus, *syuzhet* package extracts sentiment and sentiment-derived plot arcs from text using a variety of sentiment dictionaries conveniently packaged for consumption by R environment users (R Core Team, 2018).

According to R Core Team (2018), the package *syuzhet* has various basic functions. However, for the purpose of the study, our main focus was on *get_nrc_sentiment()* function. This is a function in *syuzhet* used to get emotions and Valence from the *nrc* dictionary. The function calls the *nrc* sentiment dictionary to calculate the presence of eight different emotions; "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise" and "trust", with their corresponding valence in a text file. It then provide a *Value* data frame where each row represents a sentence from the original file. As shown in Figure 3.13, the columns include one for each emotion type as well as a positive or negative valence. The ten columns were as follows: "anger", "anticipation", "disgust", "fear", "joy", "sadness", "surprise", "trust", "negative", and "positive."

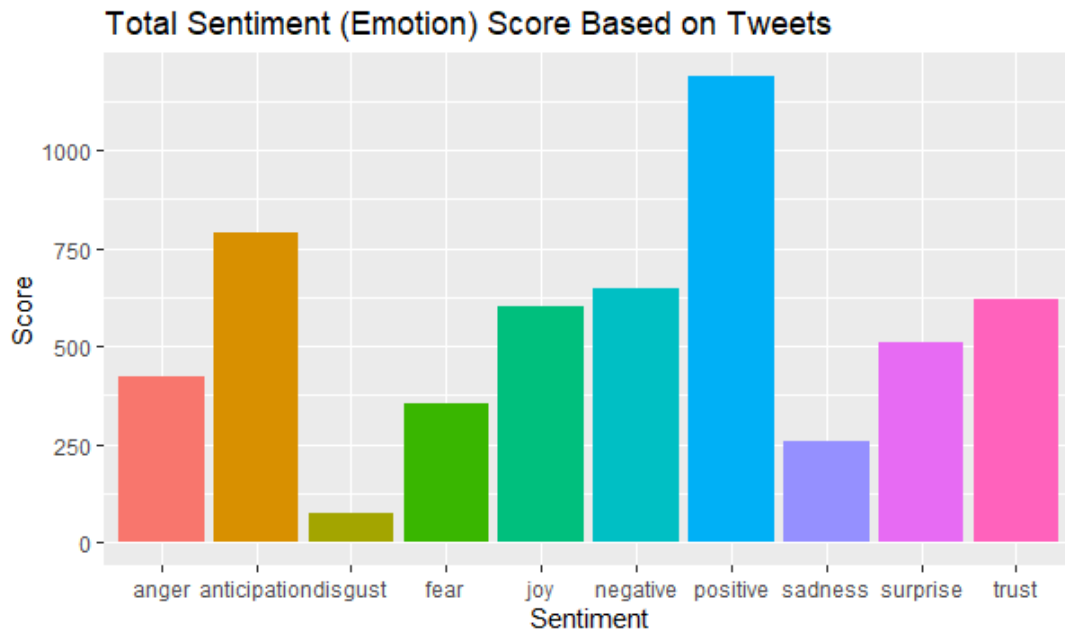


Figure 3.13: A Sample of Sentiment Emotion Types with Their Scores

The Sentiment Score data were then plotted on a data frame using *nrc* lexicon and *syuzhet* package so as to summarize the *nrc* values using the *nrc* lexicon, which provided the sentiment scores for each row of text and rated words based on specific emotions.

3.2.9.2 Segregating Positive and Negative Tweets

We further sum up the emotions from the entire dataset into specific categories of ‘positive’ and ‘negative’. To achieve this, we carried out a text classification process which identified the ‘Most Common Positive’ and ‘Most Common Negative’ sentiments using *bing* lexicon, *tidytext()* and *inner_join()* function in R.

```

> sentiments %>%
+   filter(lexicon %in% c("nrc", "bing")) %>%
+   group_by(lexicon, sentiment) %>%
+   summarise(noofwords = n())
# A tibble: 12 x 3
# Groups:   lexicon [?]
  lexicon sentiment    noofwords
  <chr>    <chr>         <int>
1 bing     negative         4782
2 bing     positive         2006
3 nrc      anger            1247
4 nrc      anticipation      839
5 nrc      disgust          1058
6 nrc      fear             1476
7 nrc      joy              689
8 nrc      negative         3324
9 nrc      positive         2312
10 nrc     sadness          1191
11 nrc     surprise          534
12 nrc     trust            1231

```

Figure 3.14: Comparison of Number of words by Sentiment Lexicon

From the Figure 3.14, both *bing* and *nrc* lexicons have more negative than positive words, but the ratio of negative to positive words is higher in the *bing* lexicon than in the *nrc* lexicon. This formed the basis as to why *bing* lexicon was used to find the most common positive and negative sentiments.

The *bing* lexicon from the *syuzhet* Package, as discussed earlier, categorizes words in a binary fashion into ‘positive’ and ‘negative’ categories. This was done to find a sentiment score for each word using the lexicon, then count the number of ‘positive’ and ‘negative’ words in the dataset. *Bing* was originally created to evaluate the sentiment of social media such as twitter data, reviews, forum discussions, and blogs (Ding et al., 2008). One way to analyze the sentiment of a text is to consider the text as a combination of its individual words and the sentiment content of the whole text as the sum of the sentiment content of the individual words. This is an often-used approach that naturally takes advantage of the *tidy* tool in R environment (Silge & Robinson, 2019).

Using tidy data principles *tidytext* makes text mining tasks easier, more effective, and consistent with tools that exist in R environment. The infrastructure needed for text mining with tidy data frames already existed in packages like *dplyr*, *tidyr* and *ggplot2*. To work

with the data as a tidy dataset, it was restructured as one-token-per-row format. This function uses the tokenizer's package to separate each line into words (Silge & Robinson, 2019). The default tokenizing is for words, but for this research analysis, it was tokenized to n-grams as shown in Figure 3.15.

```
> library(tidytext)
> dtm_tidy <- tidy(dtm)
> head(dtm_tidy, 15)
# A tibble: 15 x 3
  term      document count
  <chr>    <chr>    <dbl>
1 120hz     1         1
2 chip     1         1
3 custom   1         1
4 display  1         1
5 memc     1         1
6 motion   1         1
7 oneplus  1         1
8 smooth   1         1
9 sport    1         1
10 teas    1         1
11 alreadi 2         1
12 avail   2         1
13 can     2         1
14 case    2         1
15 deliveri 2         1
```

Figure 3.15: Words tokenized to n-grams for Analysis

Now that the data was in one-word-per-row format, it was then manipulated with tidy tools like *dplyr*, that for instance, enabled us to count using *count()* function. First, we found a sentiment score for each word using the *bing* lexicon and *inner_join()* function.

The *inner_join()* function is a mutating join that combines variables from the two data frames, Document Term Matrix data frame (*dtm_tidy*) and the *bing* lexicon. *inner_join()* function returned all rows from the 'one-token-per-row' format data (*dtm_tidy*) where there were matching values in *bing* lexicon, and all columns from *dtm_tidy* and *bing* lexicon. If there were multiple matches between *dtm_tidy* and *bing* lexicon, all combination of the matches were returned as shown in the Figure 3.16 and Figure 3.17, thereby creating a four variable *dtm_tidy_sentiments* data frame. We finally used the

`count()` function to find the ‘Most Common Positive’ sentiments and the ‘Most Common Negative’ sentiments.

```
> head(dtm_tidy_sentiments, 15)
# A tibble: 15 x 4
  word    document count sentiment
  <chr>  <chr>    <dbl> <chr>
1 smooth 1         1 positive
2 best   4         1 positive
3 smooth 5         1 positive
4 smooth 6         1 positive
5 smooth 7         1 positive
6 better 8         1 positive
7 stuck  8         1 negative
8 great  12        1 positive
9 boost  14        1 positive
10 great 15        1 positive
11 great 17        1 positive
12 best  18        1 positive
13 punch 19        1 negative
14 hell  20        1 negative
15 punch 20        1 negative
```

Figure 3.16: Head return of matching rows and columns from *dtm_tidy* and *bing* lexicon

```

> tail(dtm_tidy_sentiments, 15)
# A tibble: 15 x 4
  word    document count sentiment
  <chr>  <chr>      <dbl> <chr>
1 good   1979         1 positive
2 hell   1979         1 negative
3 like   1979         1 positive
4 leak   1980         1 negative
5 punch  1981         1 negative
6 rumor  1981         1 negative
7 like   1984         1 positive
8 proper 1984         1 positive
9 regard 1985         1 positive
10 love  1987         1 positive
11 better 1988         1 positive
12 sharp  1989         1 positive
13 thank  1989         1 positive
14 smart  1995         1 positive
15 hot    1996         1 positive

```

Figure 3.17: Tail return of matching rows and columns from *dtm_tidy* and *bing* lexicon

CHAPTER FOUR

EXPERIMENTS AND RESULTS

4.1 Introduction

Experiments in this study were devised with the aim of producing a Sentiment Analysis model that can categorize twitter product reviews into ‘positive’ and ‘negative’ classes based on a supervised learning.

In order to assess the performance and behaviour of the model, bag-of-words and topic modelling techniques were evaluated with respect to Machine Learning algorithms. The objective of the experiments at this stage was to assess and validate the performance of the single model classifiers, and ascertain patterns of accuracy, error rate and precision of the machine learning algorithms with respect to the value of the classification models. Finally, it was also necessary to assess whether training a classification ensemble model would results to better performance measures as opposed to the single classifier models. With this, evaluation of performance trend of Naïve Bayes, Support Vector Machine and the ensemble was carried out.

The evaluation results of the study were then evaluated based on various performance metrics as shown in Table 4.3 in Chapter 4.2.1.6. These performance criteria were chosen because they were commonly used in evaluation metrics of text classification researches (Kalaivani & Shunmuganathan, 2013). However, with regard to the study, emphasis was made on accuracy, precision and robustness. Robustness is the ability of a model to cope with errors (Rodriguez, 2019).

4.2 Models Creation in R

As depicted in the Literature Review in Chapter 2.4, it was evident from the researches that every Machine Learning algorithm or classification model had its own benefits and drawbacks. There was no solution or one approach that could fit all Machine Learning

problems. As such, different algorithms were demonstrated to have been developed to solve different problems for different tasks. Normally, several factors can affect a researcher's choice or selection of an algorithm for a model. For instance, selection of a classification model may mostly be made on the basis of factors such as resources available, accuracy requirement, and training time available among other factors (Gupte, et al., 2014). In addition, Wolpert and Macready (1997) also poses a fundamental question in their research; "how should we assess the performance of algorithms on problems so that we may programmatically compare those algorithms?"

According to Osisanwo et al., (2017) each classification algorithm has its inherent biases, and no single classification model enjoys superiority if no assumptions are made about the task. The researchers further suggest that, it is essential to first decide upon a metric to measure performance, then, compare at least a handful of different algorithms in order to train and select a best performing model. The commonly used metric is classification accuracy, which is defined as the proportion of correctly classified instances.

This chapter therefore describes the experiment criteria and model creation for the research in R programming language environment.

4.2.1 Naïve Bayes Classifier Model Creation in R

There are various types of Naïve Bayes, such as, Gaussian Naïve Bayes, Complement Naïve Bayes and Bernoulli Naïve Bayes that are used for various implementation of the Naïve Bayes classifier. For the purpose of implementation of the classifier in this study, our focus was the Multinomial Naïve Bayes.

4.2.1.1 Multinomial Naïve Bayes

As discussed in the Literature Review chapter 2.5.1, the Multinomial Naïve Bayes classifiers is mostly used in text classification. It ultimately makes two simplifying independence assumptions; the Bag-of-words assumption, where it is assumed that the position of words does not matter, and the conditional independence assumption, where it

is assumed that the probabilities of features $P(f_i/c)$ are independent given the class c (Jurafsky & Martin, 2017; Khan, et al., 2016). The intuition of Multinomial Naïve Bayes classifier is that text documents are represented as if they were a bag-of-words. This means that we have unordered set of words with their positions ignored, while only keep their frequency in the document.

In R environment, Naïve Bayes Classifier model creates a binomial or multinomial probabilistic classification model. It creates a relationship between a set of predictor variables and a categorical target variable. The classifier assumes that all predictor variables are independent of one another and predicts, based on an input and a probability distribution over a set of classes, thus calculating the probability of belonging to each class of the target variable.

4.2.1.2 Data Preparation

We had a data frame *dtm_tidy_sentiments* from our previous chapter that was used to further carryout experimentations for this chapter. To further analyze *dtm_tidy_sentiments* data frame as shown in Figure 3.16 and Figure 3.17 with Naïve Bayes Multinomial Probabilistic classification model in R, a necessary data preparation process was carried out to prepare it for the model analysis procedure (Iliou et al., 2015). Therefore, for Naïve Bayes training, the *dtm_tidy_sentiments* data frame was converted to a *csv* file using the *unlist()* function in R which simply converts the list (the data set) to vectors with all the atomic components in the data frame being preserved, followed by creating the *csv* file *classified_sentiments_df.csv* as shown in Figure 4.1. This was important because conversion to a *csv* file was crucial in the data preparation and analysis.

	word	document	count	sentiment
1	smooth	1	1	positive
2	best	4	1	positive
3	smooth	5	1	positive
4	smooth	6	1	positive
5	smooth	7	1	positive
6	better	8	1	positive
7	stuck	8	1	negative
8	great	12	1	positive
9	boost	14	1	positive
10	great	15	1	positive
11	great	17	1	positive
12	best	18	1	positive
13	punch	15	1	negative
14	hell	20	1	negative
15	punch	20	1	negative
16	temper	22	1	negative
17	like	23	1	positive
18	good	25	1	positive
19	right	28	1	positive
20	good	30	1	positive
21	loyal	30	1	positive
22	right	30	1	positive

Figure 4.1: The `classified_sentiments_df.csv` data frame

The csv file `classified_sentiments_df.csv` was then stored in another data frame `classifiedwords_df` in R environment.

We then used the `sample` command and `set.seed()` functions in R to randomize and sample the dataset. Sample command was used to generate a random sample from the `classifiedwords_df` data frame, randomly selecting rows from the larger set of observations in the data frame. `set.seed()` function on the other hand was used for setting the random number generator. A sample of the randomized data was then stored in the index, `sample_index` as shown in the tables below to depict some of the random data or documents.

Sampling is a method that allows researchers to get information about the population based on the statistics from a subset of the population or sample, without having to investigate every individual component (Gangwal, 2019). This was done to draw conclusions about the dataset population from samples, and it enabled us to determine the population’s characteristics by directly observing only a portion or sample of the population.

```

> classifiedwords_df[sample_index, ]
      X      word document count sentiment
1002 1002  jealous    1539     1  negative
1257 1257   smart    1977     1  positive
1221 1221  temper    1937     1  negative
 923   923   dark    1454     1  negative
 910   910   good    1439     1  positive
 986   986  punch    1526     1  negative
 817   817    top    1327     1  positive
1114 1114   great    1745     1  positive
 405   405   super     534     1  positive
 781   781   fuck    1276     2  negative
 416   416  punch     551     1  negative
 495   495   good     701     1  positive
 351   351  better     471     1  positive
1029 1029   like    1573     1  positive
 565   565  rumor     875     1  negative
 966   966  punch    1500     1  negative
1256 1256  enough    1977     1  positive
1179 1179   good    1873     1  positive
1110 1110   work    1739     1  positive
 654   654   hot    1034     1  positive
 585   585  rumor     912     1  negative
1188 1188 superior    1880     1  positive
1023 1023   like    1564     1  positive

```

Figure 4.2: Randomized Observations Corresponding to the Sample Index

Factorizing Variables

Factors are statistical data type that stores categorical variables (Ngo, 2020). These are used to represent categorical data such as ‘positive’ and ‘negative’, as was in the case of this study. The data frame *classifiedwords_df* was a five variable data frame as shown in Figure 4.2. The variables ‘word’ and ‘sentiment’ from the *classifiedwords_df* data frame were inform of character data types. For the purpose of analysis, we changed them to factors with *factor()* or *as.factor()* functions in R, as was appropriate. These functions were used to convert character or integer variables into factor variables. The *as.factor()* function creates data objects which represent sentiment variables containing unordered categorical data with predefined set values or levels such as *positive* and *negative*. The function takes a character vector as an argument and returns a factor. Categorical variable are variables that have two or more categories with no intrinsic ordering, such as ‘positive’ and ‘negative’ categories. Using factors also made it easier to generate graphical plots.

Factors in R are stored as a vector of integer values with a corresponding set of character values to use when the factor is displayed. The only required argument to a factor was a vector of values which would be returned as a vector of factor values.

```
> glimpse(classifiedwords_df)
Observations: 1,274
Variables: 5
$ X          <int> 110, 1037, 1038, 4, 663, 512, 741, 702, 1244, 158, 399, 874, 11...
$ word       <chr> "award", "leak", "punch", "smooth", "cashback", "good", "stun",...
$ document   <int> 215, 1591, 1591, 6, 1051, 758, 1217, 1138, 1962, 294, 531, 1393...
$ count      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ sentiment  <chr> "positive", "negative", "negative", "positive", "positive", "po...
```

Figure 4.3: Data frame with character data types

```
> glimpse(classifiedwords_df)
Observations: 1,274
Variables: 5
$ X          <int> 110, 1037, 1038, 4, 663, 512, 741, 702, 1244, 158, 399, 874, 11...
$ word       <fct> award, leak, punch, smooth, cashback, good, stun, thank, best, ...
$ document   <int> 215, 1591, 1591, 6, 1051, 758, 1217, 1138, 1962, 294, 531, 1393...
$ count      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ sentiment  <fct> positive, negative, negative, positive, positive, positive, neg...
```

Figure 4.4: A Factorized data frame with two Factor levels ‘positive’, ‘negative’

From the above output, factor level “*negative*” is assigned the value 1 while factor level “*positive*” is assigned the value 2. This is because factors are stored as vectors of integer values, with a set of character values, and the integer values also correspond to a category. This saves a lot of memory space and was also useful during model training.

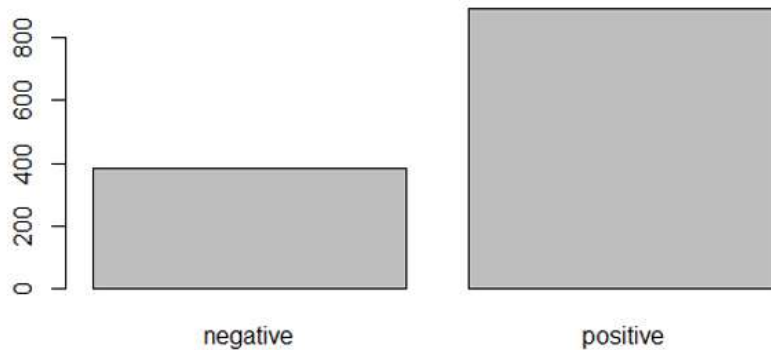


Figure 4.5: Bar plot of Sentiment grouping for 2000 Product Reviews from the *classifiedwords_df* data frame

Figure 4.5 indicated that the ‘*positive*’ class category was more than the ‘*negative*’ class category. This was important because if one class was less than or more than the other, as depicted in the summary of the variable ‘sentiment’ in Table 4.1, then a classification model could be created. The interpretation of Figure 4.5 and Table 4.1 created a good condition that advised our model training procedure.

Table 4.1: Summary of the data frame *classifiedwords_df*

x	word	document	count	Sentiment
Min : 1.0	best: 187	Min. : 1.0	Min.: 1.000	negative: 290
1st Qu. : 327.8	better :115	1st Qu.: 443.8	1st Qu.:1.000	positive:1018
Median : 654.5	win : 94	Median : 901.0	Median :1.000	
Mean. : 654.5	hard : 84	Mean : 919.6	Mean : 1.009	
3rd Qu.: 981.2	work : 74	3rd Qu.: 1354.5	3rd Qu.: 1.000	
Max. : 1308.0	super : 69	Max. : 2000.0	Max. : 3.000	

4.2.1.3 Naïve Bayes Classifier Model Training and Partitioning of the Data Frame *classifiedwords_df* for Analysis

The Naïve Bayes Classifier model was built using a training data (training set), while prediction was carried out by analyzing the test data (test set) which contained data not seen by the model during training. The test set was basically used for estimating the

model's generalization performance. The action of separating the test data normally creates a difficult trade-off between having more statistical power in estimating generalization performance versus selecting better parameters and fitting a better model (Korjus et al., 2016).

The goal of a Supervised Machine Learning model for a classification problem is normally to find a model that accurately classifies and generalizes. Korjus et al. (2016) and Alpaydin (2014) emphasizes that Supervised Machine Learning processes require data splitting. To finally test the generality of our learned model, the model was typically applied to the test data and the prediction outcome was then used to informed us about the performance of the model. Finding an optimally performing model as per our objective required a set of assumption and also trade-offs (bias-variance trade-offs) in model complexity as explained in chapter 2.4.2 in the Literature Review.

We therefore split the dataset *classifiedwords_df* into a training set at a portion of 67% of the population, while the remaining portion of 33% was set for the test set. The proportions varied in other various researches; however, it was advisable to partition a larger portion as a training set.

We then trained our Naïve Bayes Model *nb_model* in R environment, with the target variable (dependent variable) being '*sentiment*' trained as a function of the variable '*word*' which was the predictor (independent variable) in the formula. According to Kumar (2011) the independent variable is the causal variable responsible for bringing about change in the study, while the dependent variable is the outcome variable or change brought about by the introduction of the independent variable (predictors). The dataset used was the training set.

```

> nb_model <- naiveBayes(sentiment ~ word, data = nbtrain_set)
> nb_model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = x, y = y, laplace = laplace)

A-priori probabilities:
Y
negative positive
0.2231121 0.7768879

Conditional probabilities:
Y
word
negative positive
addict 0.005128205 0.000000000
afford 0.000000000 0.001472754
attack 0.025641026 0.000000000
bad 0.015384615 0.000000000
best 0.000000000 0.188512518
better 0.000000000 0.107511046

Y
word
negative positive
boil 0.005128205 0.000000000
bravo 0.000000000 0.001472754
brilliant 0.000000000 0.001472754
broken 0.005128205 0.000000000
bug 0.112820513 0.000000000
bump 0.000000000 0.000000000

Y
word
negative positive
champion 0.000000000 0.000000000
cheaper 0.000000000 0.004418262
cheer 0.000000000 0.001472754
clear 0.000000000 0.005891016
cloud 0.010256410 0.000000000
compact 0.000000000 0.001472754

Y
word
negative positive
complaint 0.005128205 0.015384615
concern 0.015384615 0.000000000
contend 0.005128205 0.000000000
cool 0.000000000 0.008836524
coolest 0.000000000 0.001472754
correct 0.000000000 0.001472754

Y
word
negative positive
crap 0.005128205 0.000000000
dead 0.000000000 0.000000000
decent 0.000000000 0.001472754
delay 0.000000000 0.000000000
delight 0.000000000 0.001472754
dim 0.097435897 0.000000000

```

Figure 4.6: A Portion of the Naïve Bayes Classifier output for 2000 Product Reviews

The above output in Figure 4.6 was an output showing the prior probabilities of the class *positive* and *negative*, also referred to as A-priori probabilities, and conditional probability tables - the likelihoods, from the Naïve Bayes model *nb_model*. The prior probabilities indicated the class '*negative*' was at 22.31% while the '*positive*' was at 77.69%. The conditional probabilities demonstrated the probabilities of how likely (likelihoods) a word was categorized as either *positive* or *negative*. The initial training was done without the application of Laplace Smoothing (add-1 approach). It was also evident that certain word

likelihoods had zero probabilities, such as; '*bump*'

```

bump
0.000000000
0.000000000

```



```
> summary(nb_model)
      Length Class Mode
apriori  2     table numeric
tables   1  -none- list
levels   2  -none- character
isnumeric 1  -none- logical
call     4  -none- call
```

Figure 4.7: Summary of the trained model for 2000 Product Reviews

Figure 4.7 demonstrate a view of the summary of the trained model in R for 2000 Product Reviews.

4.2.1.4 Naïve Bayes Classifier – Model Prediction

The trained *nb_model* was then predicted with the unseen test data. We created the prediction model *pred_nb* by using the *predict()* function in R environment. The function was used to predict the class of the words in the test set document. This was to determine as to whether the predicted words could be categorized as either *'positive'* or *'negative'* sentiments, based on the learning experience of the *nb_model* model that was trained with the training data.

```

> pred_nb <- predict(nb_model, nbtest_set)
> pred_nb
 [1] positive positive positive positive positive positive positive positive
 [9] positive positive positive positive negative positive negative positive positive
[17] positive positive positive positive positive positive negative positive positive
[25] positive positive positive positive positive positive negative positive positive
[33] positive positive positive positive positive positive negative positive positive
[41] positive positive positive positive positive positive negative positive positive
[49] positive negative positive positive positive positive positive positive negative
[57] negative positive positive positive negative positive positive positive positive
[65] positive positive negative positive positive positive negative positive negative
[73] positive positive positive positive positive positive positive positive positive
[81] positive positive positive positive positive positive positive positive negative
[89] positive positive positive negative positive positive positive positive positive
[97] positive negative negative positive positive negative negative positive positive
[105] positive positive positive positive positive positive positive positive negative
[113] positive positive positive positive positive positive positive negative positive
[121] positive negative positive positive positive positive positive positive positive
[129] negative positive positive positive positive positive positive positive positive
[137] positive positive negative positive positive negative positive positive positive
[145] positive positive positive negative positive positive positive positive negative
Levels: negative positive

```

Figure 4.8: A Sample of Naïve Bayes Prediction Outcome for the Test Document for 2000 Product Reviews

By further using an argument *type = "raw"* in the *predict()* function code, we were able to identify the actual probability measures for the first 20 individual words.

```

> head(pred_nb, 20) # this returns a view of the first 20 rows of the predicted model
      negative positive
[1,] 0.002664116 0.99733588
[2,] 0.002664116 0.99733588
[3,] 0.061032864 0.93896714
[4,] 0.002664116 0.99733588
[5,] 0.005874379 0.99412562
[6,] 0.003239472 0.99676053
[7,] 0.223112128 0.77688787
[8,] 0.002664116 0.99733588
[9,] 0.006056841 0.99394316
[10,] 0.012040753 0.98795925
[11,] 0.001521120 0.99847888
[12,] 0.004733584 0.99526642
[13,] 0.986601946 0.01339805
[14,] 0.027102154 0.97289785
[15,] 0.929848125 0.07015188
[16,] 0.005874379 0.99412562
[17,] 0.223112128 0.77688787
[18,] 0.001521120 0.99847888
[19,] 0.006056841 0.99394316
[20,] 0.986601946 0.01339805

```

Figure 4.9: pred_nb model Probability Measures for the first 20 Individual Words

4.2.1.5 Naïve Bayes Classifier Performance Metrics for the Classification Problem - The Confusion Matrix

Our study was a binary classification problem in which sentiments were classified as either *positive* or *negative*. According to Hossin and Sulaiman (2015), the discrimination evaluation of the optimal solution during a classification training was defined based on the Confusion Matrix as depicted in the Table 4.2. The confusion matrix was used for determining the correctness and accuracy of the model. The rows of the confusion matrix table represented the predicted class, while the column represented the actual class.

Table 4.2: Confusion Matrix Table for Binary Classification and the Array Representation Used in the Study

	Actual negative Class	Actual Positive Class
Predated Negative Class	True Negative Class (TN)	False Negative Class (FN)
Predated Positive Class	False Positive (FP)	True Positive (TP)

True Negatives (TN): The cases in which we predicted Negative and the actual output was also negative.

True Positive (TP): The cases in which we predicted Positive and the actual class of the data point was also positive.

False Negative (FN): False Negatives were the cases in which the actual class of the data point were Positive and the predicted were Negative. False because the model predicted incorrectly and Negative, because the class predicted was a negative.

False Positive (FP): False Positives were the cases in which the actual class of the data point was Negative and the predicted was Positive. False because the model predicted incorrectly and Positive, because the class predicted was positive.

Table 4.3: Performance Metrics for Model Evaluation

Metrics	Formula	Evaluation
Accuracy (acc)	$\frac{tp + tn}{tp + fp + tn + fn}$	The accuracy metric measures the ratio of correct predictions over the total number of instances evaluated
Error Rate (err)	$\frac{fp + fn}{tp + fp + tn + fn}$	The metric is a misclassification error measure that measures the ratio of incorrect predictions over the total number of instances evaluated
Sensitivity (sn)	$\frac{tp}{tp + fp}$	The metric measures the fraction of positive patterns that are correctly classified
Specificity (sp)	$\frac{tn}{tn + fp}$	The metric was used to measure the fraction of negative patterns that are correctly classified
Precision (p)	$\frac{tp}{tp + fp}$	Precision was used to measure the positive patterns that are correctly predicted from the total predicted

```
> confusionMatrix(table(pred_nb, nbtest_set$sentiment))
Confusion Matrix and Statistics
```

```

pred_nb   negative positive
negative   88          0
positive    7         339

      Accuracy : 0.9839
      95% CI   : (0.9671, 0.9935)
No Information Rate : 0.7811
P-value [Acc > NIR] : < 2e-16

      Kappa : 0.9515
McNemar's Test P-value : 0.02334

      Sensitivity : 0.9263
      Specificity : 1.0000
      Pos Pred Value : 1.0000
      Neg Pred Value : 0.9798
      Prevalence : 0.2189
      Detection Rate : 0.2028
      Detection Prevalence : 0.2028
      Balanced Accuracy : 0.9632

      'Positive' Class : negative

```

Figure 4.10: Confusion Matrix Results for Training Naïve Bayes Model without Laplace Smoothing

Figure 4.10 indicated the Confusion Matrix results for the first round of training without Laplace Smoothing.

Table 4.4: Naïve Bayes Confusion Matrix for 2000 Product Review Analysis

Predicted	Actual	
	Negative	Positive
Negative	88	0
Positive	7	339

Accuracy

From Table 4.4, 7 words were incorrectly classified. Accuracy for the matrix was calculated by taking average of the values lying across the “main diagonal” as shown in the Formula 4.1.

$$Accuracy = \frac{\text{True Negatives (TN)} + \text{True Positive (TP)}}{\text{Total Number of Rows in the Test Set}} \quad 4.1$$

$$\text{Total Number of Rows in the Test Set} = \text{TN} + \text{FN} + \text{FP} + \text{TP}$$

$$= 88 + 0 + 7 + 339$$

$$= 434$$

$$Accuracy = \frac{88 + 339}{434}$$

$$= 0.9839$$

$$= 98.4\%$$

4.2.1.6 Kappa Statistic

Kappa statistic is the measure of agreement between the predictions and the actual labels (*precision*). Kappa can be used to compare the ability of different raters to classify subjects into one of several groups. From the confusion matrix table, columns correspond to one "rater" (actual labels) while rows correspond to another "rater" (predicted labels). Kappa is calculated from the observed (actual) and expected (predicted/ random chance) frequencies on the diagonal of the confusion matrix. It is also interpreted as a comparison of the overall accuracy to the expected random chance accuracy.

Table 4.5: Summary of Confusion Matrix for 2000 Product Review Analysis

Predicted	Actual		Row Total
	Negative	Positive	
Negative	88	0	88
Positive	7	339	346

From the confusion matrix in Table 4.5 we were able to see that there were 434 total instances in the test set ($88 + 0 + 7 + 339 = 434$). The first column 95 were labelled as negative ($88 + 7 = 95$), and according to the second column 339 were labelled as positive ($0 + 339 = 339$). We also observed that the model classified 88 instances as *negative* ($88 + 0 = 88$) and 346 instances as *positive* ($7 + 339 = 346$), while the model's *Observed accuracy* was 0.9839.

We were then required to calculate the value of the *Expected Accuracy*. This value is defined as the accuracy that any random classifier would be expected to achieve based on the confusion matrix. The Expected Accuracy is directly related to the number of instances of each class (*negative and positive*), along with the number of instances that the Machine Learning Classifier (predicted labels) agreed with the ground truth label (actual labels).

To calculate Expected Accuracy for our confusion matrix, we first multiplied the marginal frequency of *negatives* for one "rater" by the marginal frequency of *negatives* for the second "rater", and divided by the total number of instances.

Actual label negative ($88 + 7 = 95$)

Predicted label negative ($88 + 0 = 88$)

$$\frac{(95 * 88)}{434} = 19.26$$

We then multiplied the marginal frequency of *positive* for one "rater" by the marginal frequency of *positives* for the second "rater", and divided by the total number of instances.

Actual positive (0 + 339 = 339)

Predicted positive (7 + 339 = 346)

$$\frac{(339 * 346)}{434} == 270.26$$

Finally, we added all the outcome values together, and finally divided again by the total number of instances, resulting in an *Expected Accuracy*.

$$\text{Expected Accuracy} = (19.26 + 270.26) / 434 = 0.6671$$

As previous, *Observed Accuracy* was 0.9839

$$\text{Kappa} = (\text{observed accuracy} - \text{expected accuracy}) / (1 - \text{expected accuracy}) \quad 4.2$$

$$= \frac{(0.9839 - 0.6671)}{(1 - 0.6671)}$$

$$= 0.9516$$

Further, according to Viera and Garrett (2005) the Kappa statistic varies from 0 to 1 as shown in Table 4.6, where;

Table 4.6: Kappa Statistic Interpretation

Range	Interpretation
0	Agreement Equivalent to Chance
0.1 – 0.20	Slight Agreement
0.21 – 0.40	Fair Agreement
0.41 – 0.60	Moderate Agreement
0.61 – 0.80	Substantial Agreement
0.81 – 0.99	Near Perfect Agreement
1	Perfect agreement

The higher the Kappa metric is, the better a classifier is compared to a random chance classifier. With the interpretations as demonstrated in Table 4.6, a Kappa Statistic output

of 0.9516 indicate that our Naïve Bayes classifier is a model with a ‘Near Perfect Agreement’. Further, the overall performance in terms of accuracy was at 98.4% while, the error rate was at 1.6%, a clear indication of a range below 50%, hence a stable model.

4.2.1.7 Improving the Model Training Using Laplace Smoothing

When carrying out the experiments, there were instances where *negative* words were predicted and classified as *positive* words (incorrectly classified). This could have caused significant problems for the model, such as misclassification. We therefore investigated to see whether we could slightly improve the model to achieve better performance. This was carried out by introducing Laplace Smoothing (add-1 approach) to our model. We then set a value for the Laplace estimator to 1 while training our model. This now allowed words that had zero probabilities to have some sort of significance in the classification process. Laplace Smoothing was used as a parameter smoothing, and for the purpose of improving our model training in this study. We applied the add-1 smoothing approach to prevent cases where missing, unknown or rarely occurring features inappropriately dominated the probability estimates in Multinomial Naïve Bayes as shown in the Figure 4.11.

```

> nb_model2 <- naiveBayes(sentiment ~ word, data = nbtrain_set, laplace = 1)
> nb_model2

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = x, y = Y, laplace = laplace)

A-priori probabilities:
Y
  negative positive
0.2231121 0.7768879

Conditional probabilities:
word
Y
  addict afford attack bad best better
negative 0.006079027 0.003039514 0.018237082 0.012158055 0.003039514 0.003039514
positive 0.001230012 0.002460025 0.001230012 0.001230012 0.158671587 0.091020910
word
Y
  boil bravo brilliant broken bug bump
negative 0.006079027 0.003039514 0.003039514 0.006079027 0.069908815 0.003039514
positive 0.001230012 0.002460025 0.002460025 0.001230012 0.001230012 0.001230012
word
Y
  champion cheaper cheer clear cloud compact
negative 0.003039514 0.003039514 0.003039514 0.003039514 0.009118541 0.003039514
positive 0.004920049 0.004920049 0.002460025 0.006150062 0.001230012 0.002460025
word
Y
  complaint concern contend cool coolest correct
negative 0.006079027 0.012158055 0.006079027 0.003039514 0.003039514 0.003039514
positive 0.001230012 0.001230012 0.001230012 0.008610086 0.002460025 0.002460025
word
Y
  crap dead decent delay delight dim
negative 0.006079027 0.003039514 0.003039514 0.003039514 0.003039514 0.060790274
positive 0.001230012 0.001230012 0.002460025 0.001230012 0.002460025 0.001230012

```

Figure 4.11: A Sample of Training 2000 Product Reviews with Laplace Smoothing

From Figure 4.11, it was clear that with the add-1 smoothing, words such as *'bump'*

```

      bump
0.003039514
0.001230012

```

no longer had word likelihood with zero probabilities.

However, the Figure 4.12 of the Confusion Matrix indicated a similar performance when we compared our training with or without Laplace Smoothing.

```

> confusionMatrix(table(pred_nb2, nbtest_set$sentiment))
Confusion Matrix and Statistics

pred_nb2   negative positive
negative    88         0
positive    7         339

      Accuracy : 0.9839
      95% CI   : (0.9671, 0.9935)
No Information Rate : 0.7811
P-Value [Acc > NIR] : < 2e-16

      Kappa   : 0.9515
McNemar's Test P-Value : 0.02334

      Sensitivity : 0.9263
      Specificity : 1.0000
      Pos Pred Value : 1.0000
      Neg Pred Value : 0.9798
      Prevalence : 0.2189
      Detection Rate : 0.2028
      Detection Prevalence : 0.2028
      Balanced Accuracy : 0.9632

      'Positive' Class : negative

```

Figure 4.12: Confusion Matrix Results with Laplace Smoothing

From the output of Figure 4.11, it was also clear that there were no zero probability words

anymore. In addition, the words *'bump'* and *'delay'* now had probabilities; bump
0.003039514
0.001230012

and delay
0.003039514
0.001230012 respectively, and were classified as *'negative'* sentiments. This was an early indicator of slight model improvement, however, further verification was done while building the Confusion Matrix, so as to establish further, the performance. The only difference between *nb_model* in Figure 4.6 and *nb_model2* in Figure 4.11 was that we no longer had zero probabilities in the second improved *nb_model2* model.

4.2.2 Support Vector Machine Classifier Model Creation in R

SVM is based on the idea of finding a hyper plane that best divide a dataset into two classes, that is, an optimal separating hyper plane, which is a hyper plane that is as far as possible from data points from each category (Osisanwo, et al., 2017).

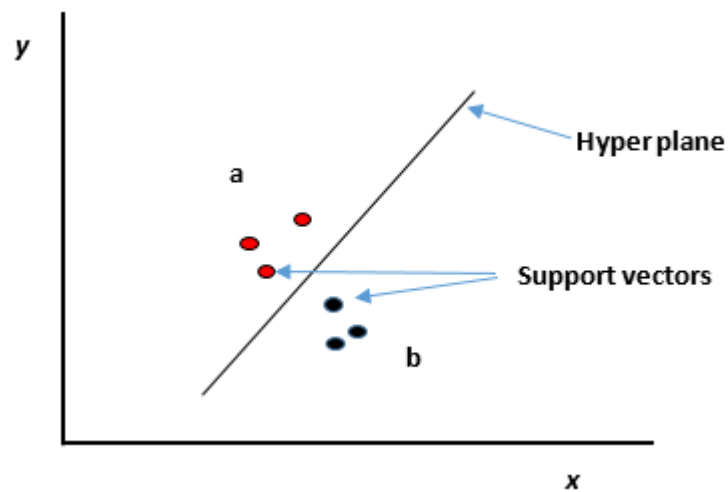


Figure 4.13: Given a particular data point (y and x) a classification *a* or *b* is made

Burges (1998) explains that support vectors as shown in Figure 4.13, are the data points that are nearest to the hyper plane, that is, the points of data sets that if removed, would alter the position of the dividing hyper plane. This implies that Support vectors are the critical elements of a data set. As described in the Literature Review Chapter 2.5, a hyper plane however, is a line that linearly separates and classifies a set of data. Intuitively, the further from the hyper plane the data points lie, the more exist a confidence that correct classification has been achieved. Data points are therefore expected to be as far away from the hyper plane as possible, while still being on the correct side of the hyper plane.

The ultimate goal of a Support Vector Machine is therefore to find the optimal separating hyper plane which maximizes the ‘margin’ of the training data, as depicted in Figure 2.8 in the Literature Review. A margin, given a particular hyper plane, is where one can compute the distance between the hyper plane and the closest data point. Once a value is

established, and it is doubled, the outcome is a margin. The margin is “a no man’s land”, meaning that there will never be any data point inside the margin. Certainly, the following observations can therefore be made; if a hyper plane is very close to a data point, its margin will be small, while also, the further a hyper plane is from a data point, the larger its margin will be. It can therefore, be concluded that the optimal hyper plane will be the one with the biggest margin and can correctly classify the training data, while at the same time, finally generalize better with unseen data (Burgess, 1998).

Datasets for our study were textual in nature, and most Text Categorization problems are linearly separable. SVM only seems to work when two classes are linearly separable (Patil, Galande, Kekan, & Dange, 2014; Joachims, 1998). Another relevant information that encouraged the use of SVM was that the study was carried out in a highly dimensional space. For instance, according to Tripathy et al. (2017) SVMs have been universally reported to work better for text classification, as textual data are known to be highly dimensional. A remarkable property of SVM is that their ability to learn can be independent of dimensionality of feature space. SVM measures the complexity of hypothesis based on margin that separates the plane and not the number of features. SVM also uses over fitting protection which helps in minimizing over fitting tendencies during learning (Patil et al., 2014; Joachims, 1998).

As demonstrated earlier in the Literature Review Chapter 2.5, the goal of our Support Vector Machine (SVM) was to find a hyperplane that best separated our two possible independent categorical variables ‘*positive class*’ and ‘*negative class*’, this was the classification problem. Intuitively, the model tried to find a decision boundary that could ‘best’ split the data values based on the potential target value of our ‘*positive*’ and ‘*negative*’ classes.

Maximizing the margin was good for SVM because the support vectors near the decision boundary represented uncertain classification decisions, meaning there was almost a 50% chance of the classifier deciding either way. Any SVM classifier with a large margin makes certain or good classification decisions, this is because a slight error in

measurement or a slight document variation will not cause a misclassification, as the large margin takes care of that. This can also be referred to as, a classification safety margin. SVM classifiers insists on a large margin around the decision boundary, where, the optimal hyperplane is at the maximum distance from the *positive* and *negative* data points. To satisfy this, and also to classify the data points accurately, the margin is maximized, this is why SVM is popularly known as the large margin classifier. Large margins also help to avoid overfitting (Manning et al., 2008).

4.2.2.1 Data preparation and Feature Engineering for SVM Model

The main data frame from our experiments, *classifiedwords_df* as shown in Figure 4.2 contained unordered categorical features, *positive* and *negative*. SVM learning required that we prepare our data in a specific way before fitting the Machine Learning model. This was because the SVM classifier required that all input variables and output variables be transformed to numeric as it could not operate on label data directly. This meant that we had to convert the categorical data to a numerical form. However, it was evident even from our experiments with Naïve Bayes classifiers that some algorithms in Machine Learning can work with categorical data directly.

A two-step process of experimentation; One-Hot Encoding and Integer Encoding was therefore implemented to convert the categorical data to a numerical form.

4.2.2.2 One-Hot Encoding

This is the process where the categorical variables were converted into binary variables (0, 1) for SVM representation, also known as “dummy variables” in other fields such as statistics. The fundamental idea behind dummy variables is to replace a categorical variable with one or more new features that can have the values 0 and 1. One-Hot Encoding assumes that all features are categorical. The values 0 and 1 are useful in the formula for linear binary classification.

To create dummy variables for the object ‘`dummysvMM`’ from the data frame `classifiedwords_df` in Figure 4.2 in the R environment, with the feature “sentiment”, we had possible values of *positive* and *negative*. To encode these two possible values, we created two new features, called “sentimentpositive” and “sentimentnegative” as shown in the Figure 4.15. A feature 1 (one) was assigned if the “sentiment” for the feature had the corresponding value and 0 (zero) otherwise, so exactly one of the two new features was 1 for each data point. This is what resulted to One-hot encoding.

```
> dummysvMM <- model_matrix(classifiedwords_df, word ~ sentiment )
> dummysvMM
# A tibble: 1,274 x 2
#   (Intercept) sentimentpositive
#   <dbl> <dbl>
1         1         1
2         1         0
3         1         0
4         1         1
5         1         1
6         1         1
7         1         0
8         1         1
9         1         1
10        1         1
# ... with 1,264 more rows
> summary(dummysvMM)
(Intercept) sentimentpositive
Min.      :1      Min.      :0.0000
1st Qu.:1      1st Qu.:0.0000
Median  :1      Median  :1.0000
Mean    :1      Mean    :0.7009
3rd Qu.:1      3rd Qu.:1.0000
Max.    :1      Max.    :1.0000
```

Figure 4.14: One-Hot Encoding for 2000 Product Reviews from the `classifiedwords_df` data frame

We further used the `cbind()` function to bind columns and recreated `classifiedwords_df` and `dummysvMM` in order to create a data frame `classifiedwords_df2` as shown in the Figure 4.15. The `cbind()` function is used to combine columns of the same dimension.

```

> classifiedwords_df2 <- cbind(classifiedwords_df[,1:5], dummiesvmm)
> head(classifiedwords_df2)
      X      word document count sentiment sentimentnegative sentimentpositive
110  110    award     215      1  positive                0                1
1037 1037    leak     1591      1  negative                1                0
1038 1038    punch     1591      1  negative                1                0
4      4    smooth        6      1  positive                0                1
663   663 cashback    1051      1  positive                0                1
512   512    good      758      1  positive                0                1

```

Figure 4.15: A Sample Output for the Data Frame *classifiedwords_df2*

While using the above data in Figure 4.15 in a Machine Learning algorithm, one can drop the original “sentiment” feature and only keep the 0–1 features.

sentimentnegative	sentimentpositive	Linear Relationships
0	1	$0+1=1$
1	0	$1+0=1$
1	0	$1+0=1$
0	1	$0+1=1$
0	1	$0+1=1$
0	1	$0+1=1$

Figure 4.16: A One-Hot Encoding Output

One-Hot encoding adds a heavy amount of dimensionality to datasets (Andre, 2020). Based on the study’s assumptions, the optimal dataset Figure 4.2 consists of features whose information are independently valuable. However, from the above output in Figure 4.16 as generated from Figure 4.15, each of the information-sparse columns have a linear relationship with each other. This leads to a ‘Dummy Variable Trap’, where one variable can be easily predicted using the others, this can eventually cause a problem of multicollinearity in high dimensions (Krishna, 2019; Andre, 2020). Multicollinearity, a serious issue in Machine Learning models, which occurs where there is a dependency between the independent features.

Granted, if there were maybe only three or even four classes, one-hot encoding may have not been a bad choice, as in order to overcome the problem of multicollinearity, one of the dummy variables would have been dropped. On the other hand, with a large number of categories or classes, one-hot encoding can lead to high memory consumption.

For this reason, further encoding experimentations were carried out using Integer Encoding.

4.2.2.3 Integer Encoding

The study employed the use of only two categorical variables, *positive* and *negative*. Often, categorical variables are encoded as integers for ease of storage or because of the way the data is collected. For status, each unique category value *positive* and *negative*, was assigned an integer value. For instance, for the purpose of our study, we transformed the output of the data frame *classifiedwords_df* in Figure 4.2 and encoded values for the feature “sentiment” with binary variables (0, 1). To do this, we first extracted the second, third, fourth and fifth columns of the data frame *classifiedwords_df* as shown below to create *classifiedwords_df3* data frame.

```

> classifiedwords_df
      X      word document count sentiment
110  110    award     215     1 positive
1037 1037     leak    1591     1 negative
1038 1038    punch    1591     1 negative

> classifiedwords_df3 <- classifiedwords_df[,c(2,3,4,5)]
> classifiedwords_df3
      word document count sentiment
110    award     215     1 positive
1037   leak    1591     1 negative
1038   punch    1591     1 negative

```

Figure 4.17: Creation of `classifiedwords_df3` data frame

To then encode the dependent variable ‘sentiment’ values as binary variables (0, 1), we removed the data frame variable titles by using a method; `header = FALSE`, and created a data frame `dataset_svm` as shown below.

```

> dataset_svm <- read.csv("classifiedwords_df3.csv",header = FALSE)
> dataset_svm
   V1      V2      V3      V4      V5
1  NA      word document count sentiment
2  110    award     215     1 positive
3  1037   leak    1591     1 negative
4  1038   punch    1591     1 negative
5    4    smooth      6     1 positive
6  663 cashback    1051     1 positive
7  512    good     758     1 positive
8  741    stun    1217     1 negative
9  702   thank    1138     1 positive
10 1244    best    1962     1 positive

```

Figure 4.18: `dataset_svm` data frame

Finally, we encoded the ‘sentiment’ feature value *negative* as integer 0 and *positive* as 1. This was shown in the output Figure 4.19 in variable `v5`. In addition, during Integer Encoding in R the variables were renamed as features ‘word’ to ‘V2’, ‘document’ to ‘V3’, ‘count’ to ‘V4’ and ‘sentiment’ to ‘V5’ respectively.

```

> dataset_svm$v5[dataset_svm$v5 == "negative" ] <- 0
> dataset_svm$v5[dataset_svm$v5 == "positive" ] <- 1
> dataset_svm
  V1      V2      V3      V4      V5 v5
1  NA      word document count sentiment NA
2  110     award      215      1 positive 1
3 1037     leak      1591      1 negative 0
4 1038     punch      1591      1 negative 0
5   4      smooth      6      1 positive 1
6  663     cashback    1051      1 positive 1
7  512     good      758      1 positive 1
8  741     stun      1217      1 negative 0
9  702     thank     1138      1 positive 1
10 1244     best     1962      1 positive 1

```

Figure 4.19: Sample Observations for Encoded *dataset_svm* data frame

Further, to have an accurate representation of the features necessary for training SVM, we removed the first observation (row) as shown in Figure 4.19 and captured only four variables; V2, V3, V4 and V5 for our new training dataset *dataset_svm1* as shown in Figure 4.19.

```

> dataset_svm1 <- dataset_svm[-1,c(2,3,4,6)]
> dataset_svm1
  V2      V3      V4      v5
2  award  215      1      1
3  leak  1591      1      0
4  punch  1591      1      0
5  smooth   6      1      1
6  cashback 1051      1      1
7  good    758      1      1
8  stun   1217      1      0
9  thank  1138      1      1
10 best   1962      1      1

```

Figure 4.20: Sample Observations for Encoded *dataset_svm1* data frame

While viewing the structure of the data frame *dataset_svm1* in R environment, we noticed that the target variable *v5* data type was in form of *numbers* as shown in Figure 4.20.

```

> str(dataset_svm1)
'data.frame': 1274 obs. of 4 variables:
 $ V2: Factor w/ 186 levels "alarm","annoy",...: 3 97 129 145 17 70 152 165 7 7 ...
 $ V3: Factor w/ 913 levels "1","1005","1007",...: 516 327 327 744 18 814 87 51 486 555 .
 ..
 $ V4: Factor w/ 4 levels "1","2","3","count": 1 1 1 1 1 1 1 1 1 1 ...
 $ v5: num 1 0 0 1 1 1 0 1 1 1 ...
> write.csv(dataset_svm1, "dataset_svm1.csv")

```

Figure 4.21: dataset_svm1 data frame Structure

However, for SVM algorithm training purposes, it was fundamental that the target variable was encoded as a factor variable. This was achieved as shown in the Figure 4.21.

```

> dataset_svm1$v5 = factor(dataset_svm1$v5, levels = c(0, 1))
> str(dataset_svm1)
'data.frame': 1274 obs. of 4 variables:
 $ V2: Factor w/ 186 levels "alarm","annoy",...: 3 97 129 145 17 70 152 165 7 7 ...
 $ V3: Factor w/ 913 levels "1","1005","1007",...: 516 327 327 744 18 814 87 51 486 555 .
 ..
 $ V4: Factor w/ 4 levels "1","2","3","count": 1 1 1 1 1 1 1 1 1 1 ...
 $ v5: Factor w/ 2 levels "0","1": 2 1 1 2 2 2 1 2 2 2 ...

```

Figure 4.22: Factorised dataset_svm1 data frame Structure

4.2.2.2 Support Vector Machine Classifier – Model Training and Partitioning of the Data Frame *dataset_svm1* for Analysis

To embark on training the SVM model, the dataset *dataset_svm1* was first split into a training and a test set. The model was built using a training data, while validation was carried out by using a test set which contained data not seen by the model during training. We therefore split the dataset *dataset_svm1* into a training set at a portion of 67% of the population, while the remaining portion of 33% was set for the test set as shown in Figure 4.23.

```

> ind <- sample(2, nrow(dataset_svm1), replace = TRUE, prob = c(0.67, 0.33))
> trainsv_set <- dataset_svm1[ind == 1,]
> testsv_set <- dataset_svm1[ind == 2,]
> head(trainsv_set)
  v2  v3 v4 v5
2  like 847 1 1
3  better 1596 1 1
6  support 1762 1 1
8  better 1172 1 1
9  win 68 1 1
10 support 456 1 1
> head(testsv_set)
  v2  v3 v4 v5
4  work 1007 1 1
5  win 1585 1 1
7  hard 886 1 0
13 better 522 1 1
18 protect 1664 1 1
22 best 1169 1 1

```

Figure 4.23: Partitioning of the Data Frame *dataset_svm1* for Analysis with Support Vector Machine

The SVM model was trained with different values of the cost parameter ‘C’ so as to tune the classifier model as shown in Figure 4.24 to Figure 4.26. The higher the value of ‘C’ during training, the less likely it was that a misclassification would occur. This approach is called *Soft Margin Classification*, as illustrated and discussed in the Literature Review Chapter 2.5.2.

```

> sv_model = svm(formula = v5 ~ .,
+               data = trainsv_set,
+               type = 'C-classification',
+               kernel = 'linear', cost = 0.1,
+               scale = FALSE)
> sv_model

```

```

Call:
svm(formula = v5 ~ ., data = trainsv_set, type = "C-classification",
     kernel = "linear", cost = 0.1, scale = FALSE)

```

```

Parameters:
  SVM-type: C-classification
  SVM-kernel: linear
           cost: 0.1

```

```

Number of Support Vectors: 680

```

Figure 4.24: Training SVM model at a value of 0.1 for the Cost Parameter

```
> sv_model = svm(formula = v5 ~ .,  
+               data = trainsv_set,  
+               type = 'C-classification',  
+               kernel = 'linear',  
+               scale =FALSE)  
> sv_model  
  
Call:  
svm(formula = v5 ~ ., data = trainsv_set, type = "C-classification",  
     kernel = "linear", scale = FALSE)  
  
Parameters:  
  SVM-Type: C-classification  
 SVM-Kernel: linear  
       cost: 1  
  
Number of support vectors: 481
```

Figure 4.25: Training SVM model at a value of 1 for the Cost Parameter

```
> sv_model = svm(formula = v5 ~ .,  
+               data = trainsv_set,  
+               type = 'C-classification',  
+               kernel = 'linear',cost =10,  
+               scale =FALSE)  
> sv_model  
  
Call:  
svm(formula = v5 ~ ., data = trainsv_set, type = "C-classification",  
     kernel = "linear", cost = 10, scale = FALSE)  
  
Parameters:  
  SVM-Type: C-classification  
 SVM-Kernel: linear  
       cost: 10  
  
Number of support vectors: 474
```

Figure 4.26: Training SVM model at a value of 10 for the Cost Parameter

To identify a ‘best’ SVM model from our experimented models of interest, we carried out a 10-fold cross validation process using the *tune()* function in the *e1071 library* in R environment as shown in Figure 4.27 and Figure 4.28. This was a technique for evaluating and validating the stability of the model, an assurance that the model had got most of the patterns from the data correct, and was low on bias.

```

> set.seed(1)
> tune.out <- tune(svm ,v5 ~ .,
+                 data = trainsv_set,
+                 type = 'C-classification',
+                 kernel = 'linear',
+                 ranges =list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100) ))
> summary(tune.out)

```

```

Parameter tuning of 'svm':
- sampling method: 10-fold cross validation

- best parameters:
  cost
    5

- best performance: 0.05609756

- Detailed performance results:
  cost      error dispersion
1 1e-03 0.31585366 0.07349746
2 1e-02 0.31585366 0.07349746
3 1e-01 0.14756098 0.04042621
4 1e+00 0.05731707 0.02574169
5 5e+00 0.05609756 0.02646963
6 1e+01 0.05609756 0.02646963
7 1e+02 0.05609756 0.02646963

```

Figure 4.27: A ten-fold Cross Validation for the ‘best’ SVM Model

```

> #to access the best model obtained
> best_model <- tune.out$best.model
> summary(best_model)

Call:
best.tune(method = svm, train.x = v5 ~ ., data = trainsv_set, ranges = list(cost = c(0.001,
0.01, 0.1, 1, 5, 10, 100)), type = "C-classification", kernel = "linear")

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel: linear
           cost: 5

Number of Support Vectors: 474

( 218 256 )

Number of classes: 2

Levels:
0 1

```

Figure 4.28: The ‘best’ Model obtained

From the output in Figure 4.27, it was clear that we were able to achieve a ‘best’ model outcome at a cost parameter of the value 5 onwards.

Equally, when we fit the Support Vector classifier and plot the resulting hyperplane, using a very large value of cost (1e5) so that no observations were misclassified in Figure 4.29, we obtained an output similar to the ‘best’ model output in Figure 4.28.

```
> sv_model = svm(formula = v5 ~ .,
+               data = trainsv_set,
+               type = 'C-classification',
+               kernel = 'linear', cost = 1e5,
+               scale = FALSE)
> summary(sv_model)

Call:
svm(formula = v5 ~ ., data = trainsv_set, type = "C-classification",
     kernel = "linear", cost = 1e+05, scale = FALSE)

Parameters:
  SVM-Type:  C-classification
 SVM-kernel: linear
      cost:  1e+05

Number of Support Vectors:  474
( 218 256 )

Number of Classes:  2

Levels:
 0 1
```

Figure 4.29: A trained SVM Model with the Largest Cost value

4.2.2.3 Support Vector Machine Classifier – Model Prediction

The trained SVM *best_model* was then predicted with the unseen test data, *testsv_set*. We then also created the SVM prediction model *sv_pred* by using the *predict()* function in R environment. The function was used to predict the class of the support vectors in the test set document. This was to determine as to whether the predicted support vectors could be categorized as either 1 (*positive*) or 0 (*negative*) sentiments, based on the learning experience of the SVM’s ‘best’ *best_model* that was trained with the *trainsv_set* data.


```

> ## SVM PREDICTION (using the test set and best_model)
> ## test the validity
>
> sv_pred <- predict (best_model ,testsv_set )
> sv_pred
  2   3   6   7  10  14  16  17  18  20  32  35  38  39  48  54  57
  1   0   1   1   1   1   1   1   1   1   0   1   1   0   1   0   0
 58  59  62  66  76  78  83  84  87  90  97 102 106 107 108 110 111
  1   1   1   1   0   1   1   1   1   1   1   1   0   1   1   1   0
115 116 117 118 120 123 129 130 133 135 142 143 152 154 156 163 166
  1   1   0   1   1   1   0   1   0   1   1   1   1   0   1   1   0
170 182 184 185 186 188 189 193 198 200 201 206 208 210 213 216 220
  1   1   1   1   1   1   1   0   1   1   1   0   1   0   1   1   0
221 223 225 229 230 231 234 237 248 250 253 254 255 258 259 260 261
  1   1   1   1   1   1   1   1   1   1   1   1   1   0   1   1   1
262 266 267 268 269 271 273 279 282 288 293 296 298 301 311 313 321
  1   0   1   1   1   0   1   1   1   0   0   1   0   0   1   1   1
322 324 325 326 327 328 329 335 341 343 344 346 348 350 351 366 374
  1   0   1   1   0   1   1   0   0   1   1   1   1   1   1   1   0
379 381 382 385 389 390 394 395 400 401 402 403 404 409 414 415 416
  0   1   1   0   1   0   1   1   1   0   1   1   1   1   1   1   1
418 423 424 425 431 433 434 438 441 448 452 453 455 456 457 462 463
  1   1   1   1   1   0   1   1   1   1   1   1   1   1   0   0   1
467 470 471 473 476 480 483 485 488 490 493 494 498 500 504 512 513
Levels: 0 1

```

Figure 4.30: A sample of the Predicted SVM Model

4.2.2.4 Support Vector Machine Classifier Performance Metrics for the Classification Problem - The Confusion Matrix

A Confusion Matrix as in the Table 4.2 was used to present a discriminative evaluation of an optimal solution during the classification training for SVM. In addition, the evaluation results for the SVM model were then based on various performance metrics as shown in Table 4.3 in Chapter 4.2.1.6.

Table 4.7: SVM Confusion Matrix for 2000 Product Review Analysis

pred_nb	Actual	
	0	1
0	95	3
1	8	338

From Table 4.7, 8 support vectors were incorrectly classified. Accuracy for the matrix was calculated by taking the average of the values lying across the “main diagonal” (TN + TP/ Total Number of Rows in the Test Set)

$$\text{Accuracy} = \frac{\text{True Negatives (TN)} + \text{True Positive (TP)}}{\text{Total Number of Rows in the Test Set}}$$

1.

$$\text{Total Number of Rows in the Test Set} = \text{TN} + \text{FN} + \text{FP} + \text{TP}$$

$$= 95 + 3 + 8 + 338$$

$$= 444$$

$$\text{Accuracy} = \frac{95 + 338}{444}$$

$$= 0.9752$$

$$= 97.5\%$$

$$\text{Error rate (err)} = \frac{\text{False Positive (FP)} + \text{False Negative (FN)}}{\text{Total Number of Rows in the Test Set}}$$

$$\text{Error rate (err)} = \frac{8 + 3}{444}$$

$$= 0.0247$$

$$= 2.5\%$$

$$\text{Sensitivity (sn)} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

$$\text{Sensitivity (sn)} = \frac{338}{338+3}$$

$$= 0.9912$$

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Posives (FP)}}$$

$$\text{Specificity (sp)} = \frac{95}{95 + 8}$$

$$= 0.9223$$

$$\text{Kappa} = \frac{(\text{observed accuracy} - \text{expected accuracy})}{(1 - \text{expected accuracy})}$$

Actual label negative (95 + 8 = 103)

Predicted label negative (95 + 3 = 98)

$$\frac{103 * 98}{444}$$

$$= 22.73$$

Actual positive (3 + 338 = 341)

Predicted positive (8 + 338 = 346)

(341 * 346 / 444 = 265.73)

$$\text{Expected Accuracy} = \frac{(22.73 + 265.73)}{444} = 0.6496$$

As previous, *Observed Accuracy* was 0.9752

$\text{Kappa} = (\text{observed accuracy} - \text{expected accuracy}) / (1 - \text{expected accuracy})$

$$= \frac{(0.9752 - 0.6496)}{(1 - 0.6496)}$$

$$= 0.9292$$

From the various outputs as demonstrated above, SVM proved to be a strong and stable classifier. For instance, performance in relation to accuracy was at 97.5% and the error

rate was also kept below 50%. On the other hand, the interpretations from Kappa Statistic indicated a ‘Near Perfect Agreement’ model.

4.2.3 The Ensemble Model Creation in R

The experimentations of the Ensemble Model was done using the Caret package, a different modelling function in R-Studio environment. Kuhn (2019) explains that Classification And REgression Training (Caret) is a set of functions in R-Studio that attempt to streamline the process for creating predictive models. This was achieved by using tools for model tuning using resampling, and variable importance estimation tools in Caret package.

The Caret package has several functions that attempt to streamline the model building and evaluation process. For instance, the *train()* function was be used to;

- Evaluate, using resampling, the effect of model tuning parameters on performance
- Choose the “optimal” model across these parameters
- Estimate model performance from a training set

This was also achieved by implementing the steps in the algorithm proposed by Kuhn (2019) as shown in Figure 4.31 for model tuning.

```

1 Define sets of model parameter values to evaluate
2 for each parameter set do
3   for each resampling iteration do
4     Hold-out specific samples
5     [Optional] Pre-process the data
6     Fit the model on the remainder
7     Predict the hold-out samples
8   end
9   Calculate the average performance across hold-out predictions
10 end
11 Determine the optimal parameter set
12 Fit the final model to all the training data using the optimal parameter set

```

Figure 4.31: Ensemble Model Tuning Algorithm (Kuhn, 2019)

Initial experiment results as shown in Table 4.8 and Table 4.9 demonstrated that the errors made by the classifiers were uncorrelated. These outcomes indicated that the classifiers were strong, yet very diverse, a condition that supported the building of the predictive ensemble model. A class of concepts is strongly learnable if there exist a polynomial-time algorithm that achieves low error with high confidence for all concepts in the class (Schapire, 1990).

Table 4.8: Comparative Results for Classifiers Accuracy

Sr. No.	No. of Tweet Reviews	Accuracy	
		Naïve Bayes	SVM
1	500	89.10%	90.80%
2	700	94%	95.70%
3	1,000	95.80%	93.50%
4	1,500	97.90%	95.80%
5	2,000	98.40%	97.50%
6	3,000	93.80%	97.50%
7	5,000	97.40%	80.40%
8	10,000	98.20%	82.30%
9	15,000	98.90%	86.70%
Total	38,700		

Table 4.9: Comparative Results for Classifiers Error Rates (*err*)

Sr. No.	No. of Tweet Reviews	Error Rate (<i>err</i>)	
		Naïve Bayes	SVM
1	500	10.90%	9.20%
2	700	6%	4.30%
3	1,000	4.20%	6.50%
4	1,500	2.10%	4.20%
5	2,000	1.60%	2.50%
6	3,000	6.20%	2.50%
7	5,000	2.60%	19.60%
8	10,000	1.80%	17.70%
9	15,000	1.10%	13.30%
Total	38,700		

With this in mind, the main assumption was that if Naïve Bayes and SVM classifiers were adequately and correctly combined, an even more accurate and robust model could be obtained from these results with an intension of improving predictions.

4.2.3.1 The Meta Classifier Model

A Meta classifier technique was therefore implemented, the outputs of the base classifiers were treated as inputs for the meta-learning model. In this approach, the model was expected to learn and adapt to different situations (Araque et al., 2017).

4.2.3.2 Stacking Ensemble

Stacking is an ensemble approach where predictions by each different model is given as input for a Meta level classifier whose output is the final class (Wolpert, 1992). This has been depicted in Figure 4.32 (as adopted from Saugata, 2018; Tang, Alelyani, & Liu, 2015). Stacking is usually employed to combine models built by different inducers as in Figure 2.1, that is, heterogeneous models (Rokach, 2010) and is equally regarded as a technique for achieving the highest generalization accuracy for strong learners (Wolpert, 1992). Using a meta-learner, the method tries to induce which classifiers are reliable vis-

a-vie the ones that are not. Most importantly, Stacking actively seeks to improve the performance of an ensemble as the Meta classifier is trained to learn and correct the errors of the base classifiers (Breiman, 1996; Wolpert, 1992). The pseudo code in Figure 4.32 elaborates the stacking algorithm (as adapted from Saugata, 2018; Tanget al., 2015).

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^m$ ($\mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

- 1: Step 1: Learn first-level classifiers
- 2: **for** $t \leftarrow 1$ to T **do**
- 3: Learn a base classifier h_t based on \mathcal{D}
- 4: **end for**
- 5: Step 2: Construct new data sets from \mathcal{D}
- 6: **for** $i \leftarrow 1$ to m **do**
- 7: Construct a new data set that contains $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \dots, h_T(\mathbf{x}_i)\}$
- 8: **end for**
- 9: Step 3: Learn a second-level classifier
- 10: Learn a new classifier h' based on the newly constructed data set
- 11: **return** $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_T(\mathbf{x}))$

Figure 4.32: Stacking Algorithm

Source: (Saugata, 2018; Tanget al., 2015)

This study focused on how the two heterogeneous classifiers, Naïve Bayes and SVM could be improved by combining them to get an even better performance. Stacking has been used by various researchers to achieve greater predictive accuracy for strong learners (Džeroski & Ženko, 2004; Moudřík & Neruda, 2015). As depicted by experimental results in Table 4.8 and Table 4.9, the study uses the two strong yet diverse base learners which are then aggregated by a Stacking ensemble.

Mathematically, if an ensemble has H base classifiers with an error rate of $e < \frac{1}{2}$, while at the same time, the base classifiers' errors are uncorrelated, then the probability that the ensemble will make an error is the probability that more than $\frac{H}{2}$ base classifiers misclassify the example. Simply, if an input-output pair is (x, y) left out of the training set of h_i , $h_i(x)$ after training is complete for h_i , the output y can still be used to evaluate the

classifier's error. Since (x, y) was not in the training set of h_i , $h_i(x)$ may differ from the desired output y . A new classifier is then trained to estimate this discrepancy, given by $y - h_i(x)$.

Essentially, a second classifier is trained to learn the error the first classifier made. Adding the estimated errors to the outputs of the first classifier can improve generalization (Oza & Tumer, 2008).

4.2.3.3 Training the Ensemble Model

A binary classification model was then fit with `caretEnsemble` package in R-Studio. `caretEnsemble` has three primary functions: `caretList`, `caretEnsemble` and `caretStack`. `caretList` is used to build lists of caret models on the same training data, with the same re-sampling parameters. `caretEnsemble` and `caretStack` are used to create ensemble models from such lists of Caret models. `caretEnsemble` uses a Generalized Linear Models (GLMs) to create a simple linear blend of binary classification models, while `caretStack` uses a Caret model to combine the predicted outputs from several component Caret models (Mayer, 2019). Using the `caretList` function, the base caret models Naïve Bayes “nb” and SVM “svmLinear” were fit to the same datasets. The function returned a list of objects which were then passed to `caretEnsemble` and `caretStack` respectively.

(a) Defining the training control

The first step in this process was to define the training control by setting up a 10-fold cross-validation in 3 iterations, and saving the resulting predictions and class probabilities as indicated in the Figure 4.33.

(b) Learning the First Base Classifier – the Naïve Bayes Model

To compute the Ensemble model, we followed the steps in the Stacking Algorithm as shown in Figure 4.32 and first trained the base level classifiers *nbmodel* on the train set,

made predictions on the test set and generated the Confusion Matrix to help identify the performance measures. This is depicted in the Figures below.

```
> ##### 1. Computing Naive Bayes using caret R package
> library(klar)
> # Build the model
> #Defining the training control
> control <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE,
+                          classProbs=TRUE) # To save the class probabilities of the out
of fold predictions
> set.seed(123)
> nbmodel <- train(as.factor(sentiment)~ word, data = nbtrain_set, method = "nb",
+                  trControl=control)

There were 32 warnings (use warnings() to see them)
> nbmodel
Naive Bayes

849 samples
 1 predictor
 2 classes: 'negative', 'positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 764, 764, 764, 764, 764, 764, ...
Resampling results across tuning parameters:

 usekernel Accuracy Kappa
  FALSE      NaN  NaN
  TRUE       0.7043594  0

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust'
 was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Figure 4.33: Defining the Training control and a training output for the first base classifier *nbmodel*

```

> summary(nbmodel)
      Length Class      Mode
apriori      2      table    numeric
tables     184    -none-    list
levels       2    -none-    character
call         6    -none-    call
x           184  data.frame list
usekernel    1    -none-    logical
varnames     184  -none-    character
xNames       184  -none-    character
problemType  1    -none-    character
tuneValue    3    data.frame list
obsLevels    2    -none-    character
param        0    -none-    list

```

Figure 4.34: Summary of the trained *nbmodel*

During training, the resampling indices were chosen using random numbers as shown in Figure 4.33. We therefore used *set.seed()* function just prior to calling *train()* to control the randomness in order to assure reproducible results and have a possible guarantee that the same random numbers are used during training.

The next aspect was to carry out predictions. Prediction for the *nbmodel* was carried out on the test set as shown on Figure 4.35.

```

> # Make predictions
> predicted.classes <- nbmodel %>% predict(nbtest_set)
> predicted.classes
 [1] positive positive positive positive positive positive positive positive
[10] positive positive positive positive positive positive positive positive
[19] positive positive positive positive positive positive positive positive
[28] positive positive positive positive positive positive positive positive
[37] positive positive positive positive positive positive positive positive
[46] positive positive positive positive positive positive positive positive
[55] positive positive positive positive positive positive positive positive
[64] positive positive positive positive positive positive positive positive
[73] positive positive positive positive positive positive positive positive
[82] positive positive positive positive positive positive positive positive
[91] positive positive positive positive positive positive positive positive
[100] positive positive positive positive positive positive positive positive
[109] positive positive positive positive positive positive positive positive
[118] positive positive positive positive positive positive positive positive
[127] positive positive positive positive positive positive positive positive
[136] positive positive positive positive positive positive positive positive
[145] positive positive positive positive positive positive positive positive
[154] positive positive positive positive positive positive positive positive
[163] positive positive positive positive positive positive positive positive
[172] positive positive positive positive positive positive positive positive
[181] positive positive positive positive positive positive positive positive
Levels: negative positive

```

Figure 4.35: A sample of the Predicted *nbmodel* base Model

Finally, we established the performance metrics for the first base classifier - *nbmodel* by using a Confusion Matrix as shown in Table 4.10 and performance measure shown in Figure 4.36.

Table 4.10: *nbmodel* Confusion Matrix for the Ensemble Model Computation

	Actual	
predicted	negative	positive
negative	0	0
positive	130	295

```

> # confusion -matrix
> confusionMatrix(predicted.classes, nbtest_set$sentiment )
Confusion Matrix and Statistics

          Reference
Prediction negative positive
negative      0         0
positive    130       295

      Accuracy : 0.6941
    95% CI : (0.6479, 0.7376)
  No Information Rate : 0.6941
    P-value [Acc > NIR] : 0.5237

      Kappa : 0

  McNemar's Test P-value : <2e-16

      Sensitivity : 0.0000
      Specificity : 1.0000
    Pos Pred Value :      NaN
    Neg Pred Value : 0.6941
      Prevalence : 0.3059
    Detection Rate : 0.0000
  Detection Prevalence : 0.0000
    Balanced Accuracy : 0.5000

      'Positive' Class : negative

```

Figure 4.36: A 2000 Product Review Analysis Sample of the *nbmodel* Performance for the Ensemble Model Computation

Interpretations of Table 4.10 and Figure 4.36 indicated that the *nbmodel* base classifier achieved a performance of 69.4% in terms of accuracy and an error rate of 30.6% among other performance measures.

(c) Learning the Second Base Classifier – the SVM Model

The second base classifier – the SVM Model was then trained with the train set as per the steps depicted in Figure 4.32. At this point, the sentiment levels for SVM were “0” and “1”, while at this stage these weren’t valid variable names in R-Studio, meaning we could not compute “0”. This posed as a challenge hence an error “*Error: At least one of the class levels is not a valid R variable name*” occurred. To resolve this, we renamed the levels of the response variable back to “*negative*”, “*positive*” for both the train and test set as shown in Figure 4.37 and Figure 4.38, and also removed missing values if there were any.

```
> #train set
> head(trainsv_set)
      v2  v3 v4 v5
4  punch 1591 1 0
5  smooth  6 1 1
8   stun 1217 1 0
9  thank 1138 1 1
11 best  294 1 1
12 best  531 1 1
> levels(trainsv_set$v5) <- c("negative", "positive")
> head(trainsv_set)
      v2  v3 v4      v5
4  punch 1591 1 negative
5  smooth  6 1 positive
8   stun 1217 1 negative
9  thank 1138 1 positive
11 best  294 1 positive
12 best  531 1 positive
```

Figure 4.37: Renaming the Levels of the Response Variable for the Train Set

```

> #test set
> head(testsv_set)
      V2  V3 V4 v5
2    award 215 1 1
3     leak 1591 1 0
6  cashback 1051 1 1
7     good  758 1 1
10    best 1962 1 1
14     mad 1792 1 0
> levels(testsv_set$v5) <- c("negative", "positive")
> head(testsv_set)
      V2  V3 V4      v5
2    award 215 1 positive
3     leak 1591 1 negative
6  cashback 1051 1 positive
7     good  758 1 positive
10    best 1962 1 positive
14     mad 1792 1 negative

```

Figure 4.38: Renaming the Levels of the Response Variable for the Test Set

We again defined the training control for SVM Model by setting up a 10-fold cross-validation in 3 iterations, and saving the resulting predictions and class probabilities as indicated in the Figure 4.39.

```

> # Training the SVM Model by selecting c value(Cost) in Linear classifier
> svcontrol <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE,
classProbs=TRUE)
> grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5)
)
> set.seed(3233)
> svm_Linear_Grid <- train(v5 ~ V2, data = trainsv_set, method = "svmLinear",
+                          trControl= svcontrol,
+                          preprocess = c("center", "scale"),
+                          tuneGrid = grid,
+                          tuneLength = 10)

```

Figure 4.39: Defining the Training control for the SVM Model

```

> svm_Linear_Grid
Support Vector Machines with Linear Kernel

820 samples
 1 predictor
 2 classes: 'negative', 'positive'

Pre-processing: centered (185), scaled (185)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 739, 738, 738, 738, 738, 738, ...
Resampling results across tuning parameters:

C      Accuracy  Kappa
0.00   NaN       NaN
0.01  0.966661  0.9255223
0.05  0.966661  0.9255223
0.10  0.966661  0.9255223
0.25  0.966661  0.9255223
0.50  0.966661  0.9255223
0.75  0.966661  0.9255223
1.00  0.966661  0.9255223
1.25  0.966661  0.9255223
1.50  0.966661  0.9255223
1.75  0.966661  0.9255223
2.00  0.966661  0.9255223
5.00  0.966661  0.9255223

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was c = 0.01.

```

Figure 4.40: The training output for the Second base classifier SVM Model

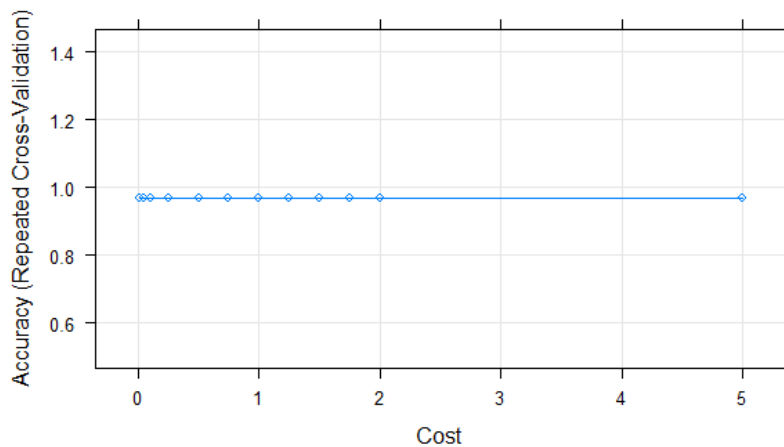


Figure 4.41: A Training plot for the SVM Model

Finally, we carried out predictions for the second base classifier – the SVM Model and generated a confusion matrix to generate performance measures as shown in Figure 4.42 and Figure 4.43.

```

> # prediction
> test_pred_grid <- predict(svm_Linear_Grid, newdata = testsv_set)
> test_pred_grid
 [1] positive negative positive positive positive negative positive positive positive
[10] positive negative positive positive negative positive negative negative positive
[19] positive positive positive negative positive positive positive positive positive
[28] positive positive negative positive positive positive negative positive positive
[37] negative positive positive positive negative positive negative positive positive
[46] positive positive negative positive positive negative positive positive positive
[55] positive positive positive positive negative positive positive positive negative
[64] positive negative positive positive negative positive positive positive negative
[73] positive positive positive positive positive positive positive positive positive
[82] negative positive positive positive positive negative positive positive positive
[91] negative positive positive positive negative negative positive negative negative
[100] positive positive positive negative negative positive positive negative positive
[109] positive negative negative positive positive positive positive positive positive
[118] negative negative negative positive positive negative positive negative positive
[127] positive positive negative positive positive positive positive positive positive
[136] positive positive positive positive positive positive negative positive negative
[145] positive positive positive positive positive positive negative negative positive
Levels: negative positive

```

Figure 4.42: A sample of the Predicted SVM base Model

```

> confusionMatrix(test_pred_grid, testsv_set$v5 )
Confusion Matrix and Statistics

          Reference
Prediction negative positive
negative    122     12
positive     0     320

      Accuracy : 0.9736
      95% CI   : (0.9543, 0.9863)
 No Information Rate : 0.7313
 P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.9348

 Mcnemar's Test P-value : 0.001496

      Sensitivity : 1.0000
      Specificity : 0.9639
   Pos Pred Value : 0.9104
   Neg Pred Value : 1.0000
     Prevalence   : 0.2687
  Detection Rate : 0.2687
Detection Prevalence : 0.2952
 Balanced Accuracy : 0.9819

 'Positive' Class : negative

```

Figure 4.43: A 2000 Product Review Analysis Sample of the SVM Model Performance for the Ensemble Model Computation

In summary, the Interpretations of Figure 4.43 indicated that the SVM Model base classifier achieved a performance of 97.4% in terms of accuracy and an error rate of 2.6% among other performance measures.

(d) Learning the Meta Classifier – the Ensemble Model

Eventually, the outputs of the base classifiers (in terms of predictions) were treated as inputs for the meta-learning model as shown in Figure 4.32 and Figure 2.12. In this approach, the model was expected to learn and adapt to different situations as the demonstrations in training the ensemble process in chapter 4.2.3.2 indicated highly diverse base learners.

We then defined the training control for Ensemble Model by setting up a 10 fold cross-validation in 3 iterations, and saving the resulting predictions and class probabilities as indicated in the Figure 4.44.

```
> enscontrol <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=
TRUE, classProbs=TRUE)
> # List of algorithms/ submodels to use in ensemble
> algorithmList <- c("nb", "svmLinear")
> models <- caretList(v5 ~ V2, data=trainsv_set, trControl=enscontrol, methodList=algori
thmList)
There were 50 or more warnings (use warnings() to see the first 50)
```

Figure 4.44: Defining the Training control for the Ensemble Model


```

> models
$nb
Naive Bayes

820 samples
 1 predictor
 2 classes: 'negative', 'positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 738, 737, 738, 738, 738, 738, ...
Resampling results across tuning parameters:

  usekernel Accuracy Kappa
  FALSE      NaN    NaN
  TRUE       0.6841501  0

Tuning parameter 'tL' was held constant at a value of 0
Tuning parameter 'adjust'
was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were tL = 0, usekernel = TRUE and adjust = 1.

$svmLinear
Support Vector Machines with Linear Kernel

820 samples
 1 predictor
 2 classes: 'negative', 'positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 738, 737, 738, 738, 738, 738, ...
Resampling results:

  Accuracy Kappa
  0.9662694 0.9250091

Tuning parameter 'C' was held constant at a value of 1

attr(,"class")
[1] "caretList"

```

Figure 4.45: The Trained Ensemble Output for 2000 Reviews

Further, predictions were carried out and a summary for the resulting output is as shown in Figure 4.46.

```

> summary(results)

Call:
summary.resamples(object = results)

Models: nb, svmLinear
Number of resamples: 30

Accuracy
      Min.   1st Qu.   Median     Mean   3rd Qu.   Max. NA's
nb      0.6829268 0.6829268 0.6829268 0.6841501 0.6829268 0.695122  0
svmLinear 0.9390244 0.9424736 0.9634146 0.9662694 0.9878049 1.000000  0

Kappa
      Min.   1st Qu.   Median     Mean   3rd Qu.   Max. NA's
nb      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000  0  0
svmLinear 0.8661006 0.8725066 0.9180546 0.9250091 0.9721278  1  0

```

Figure 4.46: The Ensemble Summary Prediction Performance Results for 2000 Reviews

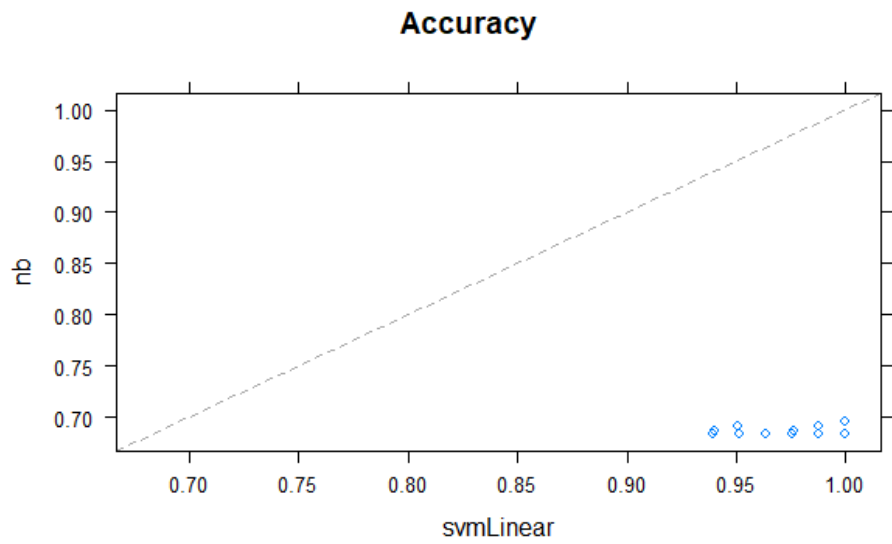


Figure 4.47: The Resampling Results in terms of Accuracy

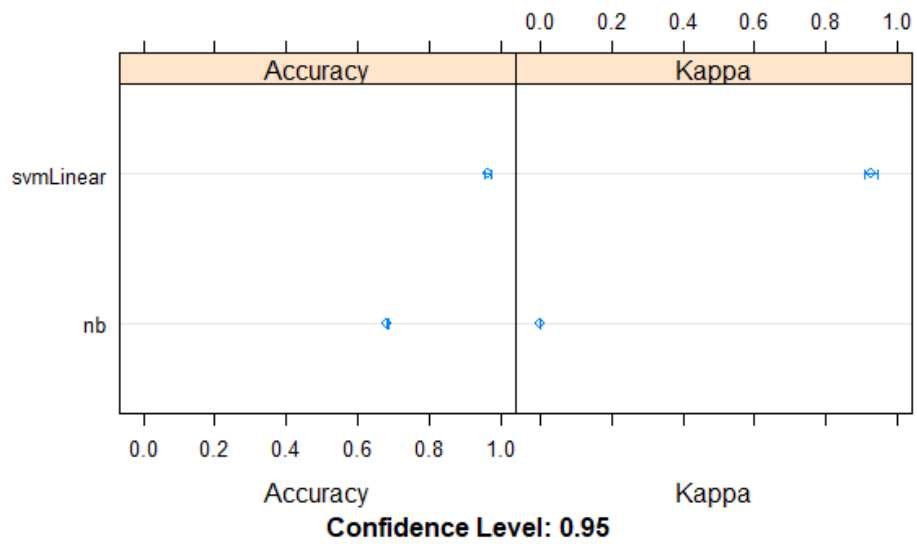


Figure 4.48: Comparison of the Base Models

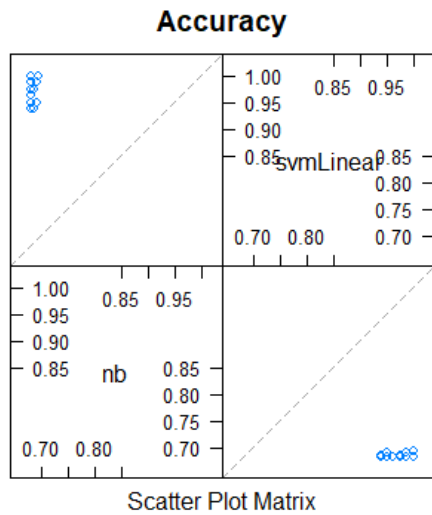


Figure 4.49: The Correlation between the Base Classifiers Results

(e) Learning the Stacked Ensemble Model

Finally, the models were aggregated by a Stacking Ensemble. The Stacked Ensemble was trained by defining the training control, setting up a 10-fold cross-validation in 3 iterations, and saving the resulting predictions and class probabilities as indicated in the Figure 4.50. This was done using a Generalized Linear Model (GLM) to create a simple linear blend of binary classification model. The outputs are as depicted in the figures below.

```

>
> # caretStack via Generalized Linear Model or GLM
>
> stackControl=trainControl(
+   method="repeatedcv",
+   number=10,
+   repeats=3,
+   savePredictions=TRUE,
+   classProbs = TRUE,
+   summaryFunction=twoClassSummary
+ )
> stacked_ensemble <- caretStack(
+   models,
+   method="glm",
+   metric="ROC",
+   trControl=stackControl
+ )
> stacked_ensemble
A glm ensemble of 2 base models: nb, svmLinear

Ensemble results:
Generalized Linear Model

2460 samples
  2 predictor
  2 classes: 'negative', 'positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2215, 2213, 2215, 2214, 2215, 2214, ...
Resampling results:

   ROC      Sens      Spec
0.9945695 0.9656621 0.9589967

```

Figure 4.50: Defining the Stacked Ensemble Training control and the Training output

```

> summary(stacked_ensemble)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.37263  -0.12281   0.02251   0.02266   2.01593

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.328e+00  8.359e-01   9.963  <2e-16 ***
nb           3.349e+08  9.975e+08   0.336   0.737
svmlinear   -1.322e+01  1.181e+00  -11.196 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3068.63  on 2459  degrees of freedom
Residual deviance:  344.52  on 2457  degrees of freedom
AIC: 350.52

Number of Fisher Scoring iterations: 9

```

Figure 4.51: Summary of the Resulting Stacked Ensemble Model

In addition, predictions were again carried out with the Stacked Ensemble Model to give a final output for the Model. Hence, the Confusion Matrix was also generated to demonstrate the varied performances as shown in Figure 4.52 and Table 4.11.

```

> confusionMatrix(stack_prediction, testsv_set$V5 )
Confusion Matrix and Statistics

          Reference
Prediction negative positive
negative    122     12
positive     0     320

      Accuracy : 0.9736
      95% CI   : (0.9543, 0.9863)
No Information Rate : 0.7313
P-Value [Acc > NIR] : < 2.2e-16

      Kappa : 0.9348

McNemar's Test P-Value : 0.001496

      Sensitivity : 1.0000
      Specificity : 0.9639
      Pos Pred Value : 0.9104
      Neg Pred Value : 1.0000
      Prevalence : 0.2687
      Detection Rate : 0.2687
      Detection Prevalence : 0.2952
      Balanced Accuracy : 0.9819

      'Positive' Class : negative

```

Figure 4.52: The Confusion Matrix Result for the Stacked Ensemble Model

Table 4.11: The Stacked Ensemble Model Confusion Matrix Table

	Actual	
predicted	negative	positive
negative	122	12
positive	0	320

From the Confusion Matrix in Figure 4.52 and Table 4.11, the Stacked Ensemble accuracy performance was at 97.4% while the error rate was at 2.6% in the analysis of 2000 reviews.

4.3 Experiment Results and Discussions

Finally, a summary based on the confusion matrix as depicted in Table 4.2, for the varied performances for the various datasets and number of reviews, which were carried out in the experiments and shown in Table 4.12 and Table 4.13. Figure 4.53 also gives a brief summary of the Resulting Stacked Ensemble Model performance.

```

> print(stacked_ensemble)
A glm ensemble of 2 base models: nb, svmLinear

Ensemble results:
Generalized Linear Model

2460 samples
  2 predictor
  2 classes: 'negative', 'positive'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2215, 2213, 2215, 2214, 2215, 2214, ...
Resampling results:

ROC      Sens      Spec
0.9945695 0.9656621 0.9589967

```

Figure 4.53: Resulting Stacked Ensemble Model Performance

Table 4.12: Comparative Results for Classifiers Performance in terms of Accuracy – with Caret Package

Sr. No.	No. of Tweet Reviews	Accuracy (<i>acc</i>)		
		Naïve Bayes	SVM	Stacked Ensemble
1	500	91.60%	97.50%	97.50%
2	700	81.10%	96.90%	96.10%
3	1,000	80.90%	96.60%	96.60%
4	1,500	87.40%	98.10%	98.10%
5	2,000	69.40%	97.40%	97.40%
6	3,000	75.90%	96.50%	96.50%
7	5,000	32.00%	98.20%	97.80%
8	10,000	32.10%	98.70%	98.70%
9	15,000	33.00%	99.40%	99.40%
Total	38,700			

Table 4.13: Comparative Results for Classifiers Performance in terms of Error Rate – with Caret Package

Sr. No.	No. of Tweet Reviews	Error Rate (<i>err</i>)		
		Naïve Bayes	SVM	Stacked Ensemble
1	500	8.40%	2.50%	2.50%
2	700	18.90%	3.10%	3.90%
3	1,000	19.10%	3.40%	3.40%
4	1,500	12.60%	1.90%	1.90%
5	2,000	30.60%	2.60%	2.60%
6	3,000	24.10%	3.50%	3.50%
7	5,000	68.00%	1.80%	2.20%
8	10,000	67.90%	1.30%	1.30%
9	15,000	67.00%	0.60%	0.60%
Total	38,700			

R programming language allows implementation and experimentation of various modelling functions or packages in one environment. For this reason, analytical experiments were carried out in different phases in R-Studio environment, with an aim of validating the model. As depicted in Chapter 4.2, the model creation experiments were first carried out with the Multinomial Naïve Bayes model found in the Naïve Bayes package. Secondly, the SVM model was then implemented with the *e1071* package, while, to carry out experiments for the ensemble model, we finally fit the model with caretEnsemble package R environment.

The experimental analysis of these classifier models involved setting up of various parameters for each phase, as shown in Chapter 4.2. This provided a varied comparison of results as shown in Table 4.8, Table 4.9, Table 4.12 and Table 4.13. Further, Table 4.14 through to Table 4.16 demonstrate the classifier’s performances in terms of precision, recall and F-Measure.

Table 4.14: Comparative Results for Classifiers Performance in terms of Precision (p)

Sr. No.	No. of Tweet Reviews	Precision (p)		
		Naïve Bayes	SVM	Stacked Ensemble
1	500	91.63%	97.31%	97.31%
2	700	81.12%	96.41%	1.00%
3	1,000	80.97%	95.97%	95.97%
4	1,500	87.35%	1.00%	1.00%
5	2,000	69.41%	1.00%	1.00%
6	3,000	75.86%	1.00%	95.60%
7	5,000	0.00%	97.40%	1.00%
8	10,000	0.00%	98.20%	98.20%
9	15,000	0.00%	1.00%	1.00%
Total	38,700			

Table 4.15: Comparative Results for Classifiers Performance in terms of Recall (r)

Sr. No.	No. of Tweet Reviews	Recall (r)		
		Naïve Bayes	SVM	Stacked Ensemble
1	500	1.00%	93.78%	93.78%
2	700	1.00%	86.35%	83.00%
3	1,000	1.00%	83.71%	83.71%
4	1,500	1.00%	88.76%	88.76%
5	2,000	1.00%	72.40%	72.40%
6	3,000	1.00%	74.10%	77.71%
7	5,000	0.00%	67.44%	65.48%
8	10,000	0.00%	67.39%	67.39%
9	15,000	0.00%	65.91%	65.91%
Total	38,700			

Table 4.16: Comparative Results for Classifiers Performance in terms of F-Measure (FM)

Sr. No.	No. of Tweet Reviews	F-Measure (FM)		
		Naïve Bayes	SVM	Stacked Ensemble
1	500	1.98%	95.51%	95.51%
2	700	1.98%	91.10%	1.98%
3	1,000	1.98%	89.42%	89.42%
4	1,500	1.98%	1.98%	1.98%
5	2,000	1.97%	1.97%	1.97%
6	3,000	1.97%	1.97%	85.73%
7	5,000	0.00%	79.70%	1.97%
8	10,000	0.00%	79.93%	79.93%
9	15,000	0.00%	1.97%	1.97%
Total	38,700			

Table 4.17: Comparative Results for Feature Vectors

Sr. No.	No. of Tweet Reviews	Feature Vectors				
		Most common (+) Features	Most common (-) Features	Total No. of Feature	Percentage (%) of Most common (+) Features	Percentage (%) of Most common (-) Features
1	500	565	54	619	91%	9%
2	700	610	130	740	82%	18%
3	1,000	806	192	998	81%	19%
4	1,500	1733	198	1931	90%	10%
5	2,000	893	381	1274	70%	30%
6	3,000	1549	514	2063	75%	25%
7	5,000	2920	1401	4321	68%	32%
8	10,000	4879	2372	7251	67%	33%
9	15,000	6621	3265	9886	67%	33%
Total	38,700					

The classifiers' performance results with regard to precision, as shown in Table 4.14 indicated that best performance for SVM and the Stacked Ensemble was competitive at 98.20% compared to Naive Bayes whose best performance was at 91.63%. On the other hand, SVM and the Stacked Ensemble also performed best at 93.78% and 95.51% on recall and F-Measure (FM) respectively.

However, as shown in Table 4.14 there were instances where the performance on precision was at 1% for both SVM and the Stacked Ensemble. This was attributed to the classifiers yielding no false positive instances on their respective confusion matrices. At the same time there were also instances where Naive Bayes recorded 0%. This on the other hand was attributed to the classifier yielding no false positive and true positive instances as depicted on the confusion matrices.

The goal of the final phase of experiments was to evaluate the performance for Sentiment classification in terms of accuracy and robustness, while subjecting the Ensemble Model to a different modelling function in R-Studio - the Caret package. We therefore compared the two supervised Machine Learning algorithms, for sentiment classification of 38,700 Twitter product reviews. From the experiments, as much as there existed significant amount of errors, Naïve Bayes could not be regarded as an unstable learning system. Arguably, the experiments indicated that SVM and Naïve Bayes were good classifiers for Sentiment analysis and Text Classification due to the relatively good performances achieved in Table 4.12 and Table 4.13. The Stacked Ensemble Model additionally demonstrated a good ability to cope with errors as indicated in Table 4.13. Generally speaking, all the classifiers coped well with errors, as the highest record of error rate measure was realized by Naïve Bayes at 68%. However, it was also noticed that there was a reduction in percentage of the error rate for Naïve Bayes with progressive increase of reviews to be analyzed from 10,000 reviews onwards.

The relatively good performance results for SVM and the Stacked Ensemble with regard to recall, as depicted Table 4.15 indicated that the models significantly estimated the probability that a randomly selected true positive instance was predicted as positive. The minimal performance for Naive Bayes on the other hand, was because there were instances where Naive Bayes classifier yielded no false positive and true positive instances for reviews from 5,000 to 15,000 reviews. Additionally, Naive Bayes recorded performances of 1% for 500 to 3,000 reviews since the classifier yielded no true negative instances on the respective confusion matrices.

As mentioned earlier there were good performance results for SVM and the Stacked Ensemble with regard to F-Measure while Naïve Bayes' best performance was at 1.98%. This was attributed to the fact that Naïve Bayes had no performance on recall and precision in the analysis of 5,000 to 15,000 reviews.

With regard to the varied result on the performance measure in Table 4.12 to 4.13, information on Table 4.17 depicts that in terms of structure, our datasets were significantly imbalanced. This was evidenced by the fact that the target class '*sentiment*' had an uneven distribution of observations, that is, the '*positive*' class (the majority class) had high numbers of observations, as compared to the '*negative*' class (the minority class) which had low numbers of observations. With this, our classification model was therefore biased towards the predictions as seen in Figure 4.8, Figure 4.30 and Figure 4.50. In this case, the confusion matrix for the classification problem was used to show how well our model classified the target classes and arrived at the performance accuracies in Table 4.8 and 4.12. These were based on the confusion matrices principles in Table 4.2 and 4.3 respectively.

Using accuracy as a performance measure may be good enough for a well-balanced class problem, but not preferable for an imbalanced class problem (Mazumder, 2021). Therefore, the building of the Ensemble model together with the use of F-Measure (FM) as a performance metric was adequately used to deal with the class imbalance problem. F-Measure (FM) was used as an evaluation metric, since it is a harmonic mean of precision and recall. This means that if the classifier predicts the minority class but the prediction is erroneous and false-positive increases, the precision metric and F-Measure will be low. In addition, if the classifier identifies the minority class poorly, in a way that more of the minority class is wrongfully predicted as the majority class, then false negatives will increase, so recall and F-Measure will be low. At the same time, F-Measure only increases if both the number and quality of prediction improves (Tripathi, 2019; Mazumder, 2021).

On the other hand, the class imbalance problem as depicted in Table 4.17 was relatively low from 5,000 to 15,000 reviews. This resulted in minimal impact on our model's performance evaluation.

CHAPTER FIVE

SUMMARY, CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

5.1 Introduction

This section covers all the work done in summary, achievements, conclusions, recommendations, limitations of the research study, knowledge contribution and future work.

5.2 Summary

In the analysis of the two classification approaches, as shown in Chapter 4 Table 4.12 and Table 4.13, the results in the summary indicated that in terms of error rate, Naïve Bayes seemed to be generating a significant amount of errors during computation as compared to SVM. From the Confusion Matrices as depicted in Chapter 4 Figure 4.36 and Figure 4.43, basically, there were two kinds of errors, False Negative (FN) and False Positive (FP), while the experiments in Chapter 4 Figure 4.36 indicated that for instance, Naïve Bayes had a significant rate of False Positive (FP) with 130 instances. As indicated in Chapter 4, Table 4.12 and Table 4.13, SVM performances were better than Naïve Bayes in terms of accuracies and error rate measure. However, the Stacked Ensemble Model's performances were similar to that of SVM performances, except for experiments carried out for 700 and 5,000 reviews where the ensemble had a significant amount of errors at 3.9% and 2.2% respectively. A 10-fold cross validation was also carried out during classifier training with the intension of tuning and developing a *'best'* model – a model with minimum amount of errors. The aim for the study was to experiment on the accuracies and the error rates of the base classifiers, to establish the impact and effectiveness of building the ensemble model. From the outcomes in Tables 4.8, 4.9, 4.12 and 4.13, generally speaking, the results indicated that Naïve Bayes and SVM were considerably strong, yet diverse good text classifiers.

Finally, R-Studio was used as a tool for the research study, analysis, implementation of

the research methodology and generation of graphical plots. It is a free, open-source, cross-platform programming environment designed for statistical analysis, which makes it highly suitable for Data science applications such as Sentiment Analysis. The models were validated with data sets of tweet reviews of "OnePlus 7 Pro" mobile phone product, in the rounds of experiments where the data was categorized as *positive* and *negative*.

5.3 Achievements

The general objective of this research was to develop an ensemble model for target Sentiment classification of product reviews using Naïve Bayes and Support Vector classifiers. This was achieved by assembling tools provided for in R-Studio platform that were then run to experiment with an aim of producing a Sentiment Analysis model that could categorize twitter product reviews into ‘positive’ and ‘negative’ classes based on supervised learning. The conclusions of the experiments are as discussed in Chapter 4.3.

The first specific objective was to identify a basic workflow for conducting sentiment analysis for product reviews from Twitter data. This was achieved by creating a step-by-step Sentiment Analysis methodology that enabled the segregation of ‘positive’ sentiments from ‘negative’ sentiments. This has been described in Chapter 3.2.

The second specific objective was to assess and validate the performance of Naïve Bayes and Support Vector Machine as supervised Machine learning algorithms in sentiment classification. The classifiers were subjected to metrics for binary classification problems, which were derived from various confusion matrices from the different datasets. The confusion matrices depicted two by two contingency tables of the predicted and observed class labels. However, emphasis was made on accuracy, precision and robustness. The final results for the performance metrics as shown in Table 4.12 through to Table 4.17 are discussed in Chapter 4.3.

The third specific objective was to experiment with the proposed algorithms singularly, and finally based on the resulting outcome, determine a suitable approach of creating an

ensemble model of the two classifiers. This was achieved by carrying out experiments with an aim of assessing and validating the performance of the single model classifiers, and ascertaining patterns of accuracy, error rate and precision of the machine learning algorithms with respect to analyzing the impact of the classifiers in terms of performance, during the creation of the ensemble model. With the evaluation of Naïve Bayes and Support Vector Machine classifiers, for instance, the error rate results demonstrated that the errors made by the classifiers were uncorrelated. These were shown in Tables 4.8 and 4.9. In addition, a class imbalance problem during experimentations, depicted in Table 4.17, was also very significant in this study. The outcomes of these experiments in Chapter 4.3 supported the building of the predictive ensemble model.

The fourth specific objective was to finally develop and validate the performance of the proposed ensemble model, based on Naïve Bayes and Support Vector Machine, to be used in Sentiment classification of products reviews. This was achieved by using tools for model tuning, using resampling; and variable importance estimation tools, in Caret package - R-Studio. The results and discussions are also as discussed in Chapter 4.3.

Other achievements with regards to journal publications for the research study are shown in Table 5.1.

Table 5.1: Publication Achievements

TITLE	JOURNAL	STATUS
Target Sentiment Analysis Ensemble for Product Review Classification	Journal of Information Technology Research (JITR) 15(1), 1-13. http://doi.org/10.4018/JITR.299382 (ISSN 1938-7857) – EI Indexed	Published
Target Sentiment Analysis Model with Naïve Bayes and Support Vector Machine for Product Review Classification	International Journal of Computer Science and Information Security (IJCSIS), Vol. 17, No. 7, July 2019	Published
Towards the Creation of an Ensemble Model for Sentiment Analysis Based on Naïve Bayes and Support Vector Machine for Product Review Classification: A Literature Survey	The RUFORUM Sixth Biennial Regional Conference 22 - 26 October 2018, Nairobi, Kenya. RUFORUM Working Document Series (ISSN 1607-9345), 2018, No. 17 (3): 736-749 Retrieved from http://repository.ruforum.org	Published

5.4 Conclusion

As evidenced from our experiments, the key assumption when dealing with Machine Learning classifiers is not whether a learning algorithm is superior to others, but the ability to investigate under which conditions a particular classifier can significantly outperform others on a given application problem.

Our Sentiment Analysis model was regarded to be a stable model since the performance on the error rates were kept at a minimal range. These resulted to the development of a robust model as demonstrated in Table 4.13.

Finally the Stacked Ensemble model was also adequately used to deal with the class

imbalance problem, as the ensemble technique was used to combine the resulting performance of the two base classifiers so as to improve performance as shown in Table 4.16. F-Measure was also adequately used to keep the balance between precision and recall, and to improve the performance score only if the classifier identified more of a certain class correctly. In addition, while computing the Ensemble model using the Caret package, the meta-learner, induced which classifier was reliable as compared to the one that was not. The outcome indicated that the reliable classifier was SVM, since it generally performed better in the text classification process. These outcomes resulted to the contributions that the Stacked Ensemble made to the research study.

5.5 Recommendations

Based on the conclusive results of the experiments carried out with Naïve Bayes and SVM as single classification models, the research recommends the adoption of these classifier in Text Mining and Sentiment analysis, as it was found out that both the classifiers are strong and very good text classification models. In addition, the research can also advise on the use of these classifiers to reduce costs sourced from complexity with keeping success rates in Sentiment Analysis.

Secondly, to adequately carryout the Sentiment Analysis with machine learning approaches during the research study, a basic workflow for conducting sentiment analysis for product reviews from Twitter data was identified and implemented. The research therefore recommends the adoption of this workflow, or a similar one, so as to facilitate data preparation and cleaning tasks in the domain of Machine Learning.

Further, the experimental results from this study are not dependent on any specific language. For this reason, proposed method is implementable for Sentiment Analysis in other language.

5.6 Limitations of the Research Study

The research study encountered a number of challenges. To begin with, Complexity of

Natural Language was a big challenge in Text Mining and Sentiment Analysis. The natural language is not free from the problem of ambiguity. One word may have multiple meanings and multiple words can have same meaning.

Secondly, analysis of colloquial/ multilingual languages was also a major challenge as the researchers faced a challenge of building resources, that is, lexicons and dictionaries for these languages. The research therefore only relied on the available lexicons in R-Studio.

Finally, instead of relying on natural existence of language as data or evidence in Natural Processing Language as they occur, in the research study, the text classification process with Machine Learning aimed to make classifications based on past observations, that is, data training had to be done.

5.7 Knowledge Contribution to the field of study

Based on the new experiments carried out in this research study with Naïve Bayes and Support Vector Machine classification models, our accomplished tasks contributed some knowledge in the field of Sentiment Analysis, Text Mining and Machine Learning. This has been outlined from our main research task that was to develop an ensemble model for target sentiment classification of product reviews using Naïve Bayes and Support Vector Machine classifiers. An ensemble comprises of individually trained base classifiers whose predictions are combined when classifying instances. Some of the currently popular ensemble methods in prior research works include Boosting, Bagging and Stacking. In this research study, we reviewed these methods and demonstrated why ensembles may perform better than single models, however, from our results, SVM produced better performances as seen in Chapter 4.3. Additionally, some new experiments were presented to demonstrate the computational ability of Naïve Bayes classifier, SVM classifier and finally, aggregation by Stacking approach.

Secondly, another goal in the research study was to identify a basic workflow for conducting sentiment analysis for product reviews from Twitter data. This was

accomplished by the basic aspect of creating a methodology that produced relevant features for the Sentiment classification model, using data preparation techniques, Naïve Bayes and Support Vector Machine classifiers, while also evaluating the performance of the classifiers and investigating their universal reliability. The outlined methodology in Chapter 3 therefore also contributed some knowledge in this field of study.

5.8 Future Work/ Further Gaps in Related Research

For status, implementation of experiments on R-Studio lexicons ('bing' and 'nrc') constituted to a significant limiting factor. For this reason, we are planning to implement experiments on other English and multilingual corpuses as an extension of this study.

Secondly, from our analysis and experiments, Feature Engineering and Data Preparations came out as very important tasks in the domain of Machine Learning, and generally Sentiment Analysis, since converting original documents to feature vectors is critical in this area of study. As a result, we are also planning to carry out further research and experimentation to identify ways and criteria of engineering features for Machine Learning. Additionally, experiments carried out at the Ensemble phase indicated that further research can be done to investigate the correlations between meta-attributes and the performance of the learning algorithms.

Finally, other text classification methods except for the ones employed in this research study should be used and their effects on the research problem in Sentiment Analysis should be observed and studied.

REFERENCES

- Abbas, M., Memon, K. A., Jamali, A. A., Memon, S., & Ahmed, A. (2019). Multinomial Naive Bayes classification model for sentiment analysis. *IJCSNS Int. J. Comput. Sci. Netw. Secur*, 19(3), 62.
- Abirami, M. S., Uma, M., & Prakash, M. (2016). Sentiment Analysis of Informal text using a Rule based Model. *Journal of Chemical and Pharmaceutical Sciences (JCPS) Volume 9 Issue 4*, pp. 2854 – 2858.
- Abuassba, A. O., Zhang, D., Luo, X., Shaheryar, A., & Ali, A. (2017). Improving Classification Performance through an Advanced Ensemble Based Heterogeneous Extreme Learning Machines. *Computational intelligence and neuroscience*, 2017.
- Ahmad, M., Aftab, S., Muhammad, S. S., & Ahmad, S. (2017). Machine Learning Techniques for Sentiment Analysis. *A review. International Journal of Multi-disciplinary science and Engineering*, 8(3), 27-35.
- Akhtar, M. S., Gupta, D., Ekbal, A., & Bhattacharyya, P. (2017). Feature Selection and Ensemble Construction: A two-step method for aspect based sentiment analysis. *Knowledge-Based Systems Volume 125, 1 June 2017* , 116-135.
- Al Amrani, Y., Lazaar, M., & El Kadiri, K. E. (2018). Random Forest and Support Vector Machine Based Hybrid Approach to Sentiment Analysis. *Procedia Computer Science*, 127, (pp. 511-520).
- Alayba, A. M., Palade, V., England, M., & Iqbal, R. (2017). Arabic Language Sentiment Analysis on Health Services. *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, pp. 114-118,
- Alnashwan, R., O’Riordan, A., Sorensen, H., & Hoare, C. (2016). Improving Sentiment Analysis Through Ensemble Learning of Meta-level Features. *In: KDWEB 2016:*

2nd International Workshop on Knowledge Discovery on the Web. Sun SITE Central Europe (CEUR)/RWTH Aachen University (2016).

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Althnian, A., AlSaeed, D., Al-Baity, H., Samha, A., Dris, A. B., Alzakari, N., Elwafa, A. A., & Kurdi, H. (2021). Impact of dataset size on classification performance: an empirical evaluation in the medical domain. *Applied Sciences*, *11*(2), 796.

Anderson, B. &. (2005). Active Learning for Hidden Markov Models: Objective Functions and Algorithms. *In Proceedings of the 22nd international conference on Machine learning*, (pp. 9-16).

Andre, Y. (2020, August 28). Stop One-Hot Encoding. Your Categorical Variables – There are Many Better Alternatives. *Towardsdatascience.com*. Retrieved from <https://towardsdatascience.com/stop-one-hot-encoding-your-categoricalvariables-bbb0fba89809>.

Araque, O., Corcuera-Platas, I., Sánchez-Rada, J. F., & Iglesias, C. A. (2017). Enhancing Deep Learning Sentiment Analysis With Ensemble Techniques in Social Applications. *Expert Systems with Applications*, *77*,, 236-246.

Asghar, M. Z., Khan, A., Ahmad, S., & Kundi, F. M. (2014). A Review of Feature Extraction in Sentiment Analysis. *Journal of Basic and Applied Scientific Research* *4*(3), 181-186.

Asif, M., Ishtiaq, A., Ahmad, H., Aljuaid, H., & Shah, J. (2020). Sentiment Analysis of Extremism In Social Media From Textual Information. *Telematics and Informatics*, *48*, 101345.

- Awachate, P. B., & Kshirsagar, V. P. (2016). Improved Twitter Sentiment Analysis Using N-Gram Feature Selection and Combinations. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(9),
- Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., & ... & Zhang, J. D. (2020). An Introduction to Machine Learning. *Clinical pharmacology & therapeutics*, 107(4), 871-885.
- Bagheri, A., Saraee, M., & de Jong, F. (2013). Sentiment Classification in Persian: Introducing A Mutual Information-Based Method for Feature Selection. In *2013 21st Iranian conference on electrical engineering (ICEE) IEEE*. (pp. 1-6).
- Balahur, A., & Perea-Ortega, J. M. (2015). Sentiment Analysis System Adaptation for Multilingual processing: The case of Tweets. *Information Processing and Management* 51(2015), 547–556.
- Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(2009), 1-127.
- Bhadane, C., Dalal, H., & Doshi, H. (2015). Sentiment Analysis: Measuring Opinions. *International Conference on Advanced Computing Technologies and Applications (ICACTA-2015)*. *Procedia Computer Science* 45(2015), 808 – 814
- Bhaskar, J., Sruthi, K., & Nedungadi, P. (2014). Enhanced Sentiment Analysis of Informal Textual Communication in Social Media by Considering Objective Words and Intensifiers. *IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, Jaipur, India.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3, 993–1022.

- Bortolussi, L., Gallo, G. M., Křetínský, J., & Nenzi, L. (2022). Learning Model Checking and the Kernel Trick for Signal Temporal Logic on Stochastic Processes. *In International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Cham.* (pp. 281-300).
- Bose, R., Dey, R. K., Roy, S., & Sarddar, D. (2020). Sentiment Analysis on Online Product Reviews. *In Information and Communication Technology for Sustainable Development (pp. 559-569). Springer, Singapore.*
- Bouazizi, M., & Ohtsuki, T. (2019). Multi-Class Sentiment Analysis on Twitter: Classification Performance and Challenges. *Big Data Mining and Analytics. 2019, 2(3), 181-194*
- Boumans, J. W., & Trilling, D. (2016). Taking Stock of the Toolkit: An Overview of Relevant Automated Content Analysis Approaches and Techniques for Digital Journalism Scholars. *Digital Journalism, 4(1), 8–23.*
- Breiman, L. (1996). Stacked Regressions. *Machine Learning, 24, 49-64.*
- Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics, 26(3), 801-824.*
- Brownlee, J. (2019, August 12). Supervised and Unsupervised Machine Learning Algorithms. *Machine Learning Mastery - Making Developers Awesome at Learning.* Retrieved from <https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/>.
- Burges, C. J. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery 2(1998), 121-167.*
- Carrillo-de-Albornoz, J., Rodriguez Vidal, J., & Plaza, L. (2018). Feature Engineering for Sentiment Analysis in e-Health Forums. *PloS one, 13(11), e0207996.*

- Chandni, Chandra, N., Gupta, S., & Pahade, R. (2015). Sentiment Analysis and its Challenges. *International Journal of Engineering Research & Technology (IJERT)*, Vol. 4 Issue 03, March-2015, 968-970.
- Chatsiou, K., & Mikhaylov, S. J. (2020). Deep Learning for Political Science. *arXiv preprint arXiv:2005.06540*.
- Chaturvedi, I., Cambria, E., Welsch, R. E., & Herrera, F. (2018). Distinguishing Between Facts and Opinions For Sentiment Analysis: Survey and challenges. *Information Fusion*, 44, 65-77.
- Cho, S., Vasarhelyi, M. A., Sun, T., & Zhang, C. (2020). Learning from Machine Learning in Accounting and Assurance. *Journal of Emerging Technologies in Accounting*, 17(1), 1-10.
- Ciaccia, P., Patella, M., & Zezula, P. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Vldb (97, pp. 426-435)*.
- Dalis, M. (2014, July 10). What is a Term-Document Matrix? [Msg 1]. Retrieved from <https://www.quora.com/What-is-a-term-document-matrix>.
- Das, S. R., Mishra, D., & Rout, M. (2019). Stock Market Prediction Using Firefly Algorithm with Evolutionary Framework Optimized Feature Reduction for OSELM Method. *Expert Systems with Applications: X(4)*, 100016.
- Dashtipour, K., Poria, S., Hussain, A., Cambria, E., Hawalah, A. Y., Gelbukh, A., & Zhou, Q. (2016). Multilingual Sentiment Analysis: State of the Art and Independent Comparison of Techniques. *Cogn Comput* 8(2016), 757–771.
- Demir, N. (2015, November 17). Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results. *Data Science and Databases*. Retrieved

from <https://www.toptal.com/machine-learning/ensemble-methods-machine-learning>.

Deniz, A., & Kiziloz, H. E. (2017). Effects of Various Preprocessing Techniques to Turkish Text Categorization Using N-Gram Features. *In 2017 International Conference on Computer Science and Engineering (UBMK)*. (pp. 655-660). IEEE.

Devika, M. D., Sunitha, C., & Ganesh, A. (2016). Sentiment Analysis: A Comparative Study On Different Approaches. *Fourth International Conference on Recent Trends in Computer Science & Engineering, Chennai, Tamil Nadu, India, Procedia Computer Science* 87(2016), 44-49.

Dey, L., Chakraborty, S., Biswas, A., Bose, B., & Tiwari, S. (2016). Sentiment Analysis of Review Datasets Using Naïve Bayes' and K-NN Classifier. *I.J. Information Engineering and Electronic Business*, 4, 54-62. Retrieved from <http://www.mecspress.org/>

Dhiraj, K. (2019, May 26). Top 5 Advantages and Disadvantages of Decision Tree Algorithm. Retrieved from <https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a>.

Ding, X., Liu, B., & Yu, P. (2008). A Holistic Lexicon-Based Approach to Opinion Mining. *WSDM'08 - Proceedings of the 2008 International Conference on Web Search and Data Mining*. (pp. 231-240).

Drus, Z., & Khalid, H. (2019). Sentiment Analysis in Social Media and its Application: Systematic Literature Review. *Procedia Computer Science*, 161, (pp. 707-714).

Ducange, P., Fazzolari, M., Petrocchi, M., & Vecchio, M. (2019). An Effective Decision Support System for Social Media Listening Based on Cross-Source Sentiment Analysis Models. *Engineering Applications of Artificial Intelligence*,

78, 71-85. Retrieved from <https://doi.org/10.1016/j.engappai.2018.10.014>.

Duwairi, R. M., & Qarqaz, I. (2014). Arabic Sentiment Analysis using Supervised Classification. *The 1st International Workshop on Social Networks Analysis, Management and Security (SNAMS - 2014), August 2014, Barcelona, Spain.*

Džeroski, S., & Ženko, B. (2004). Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning, 54*, 255–273.

Džeroski, S., Panov, P., & Ženko, B. (2009). Ensemble Methods in Machine Learning. *Encyclopedia of Complexity and Systems Science, 5317-5325.*

Ebrahimi, M., Yazdavar, A. H., & Sheth, A. (2017). Challenges of Sentiment Analysis for Dynamic Events. *IEEE Intelligent Systems, 32(5)*, 70-75.

Elango, V., & Narayanan, G. (2014). Sentiment Analysis for Hotel Reviews. *CS 229 Machine Learning, Final Projects Stanford University, Autumn 2014.*

Fersini, E., Messina, E., & Pozzi, F. A. (2014). Sentiment Analysis: Bayesian Ensemble Learning. *Decision Support Systems, 68*, 26-38

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC medical informatics and decision making, 12(1)*, 1-10.

Forouzandeh, S., Sheikahmadi, A., Rezaei Aghdam, A., & Xu, S. (2018). New Centrality Measure for Nodes Based on User Social Status and Behaviour on Facebook. *International Journal of Web Information Systems, 14(2)*, 158-176.

Fragos, K., Belsis, P., & Skourlas, C. (2014). Combining Probabilistic Classifiers for Text Classification. *Procedia - Social and Behavioral Sciences, 147*, 307-312.

- Gagolewski, M. (2017). R Package Stringi: Character String Processing Facilities [Computer software manual]. Retrieved from <http://www.gagolewski.com/software/stringi/>.
- Gaikwad, S., Chaugule, A., & Patil, P. (2014). Text Mining Methods and Techniques. *International Journal of Computer Applications (0975 – 8887)*, 85(17)
- Gangwal, R. (2019, September 11). A Data Scientist's Guide to 8 Types of Sampling Techniques. Retrieved from, <https://www.analyticsvidhya.com/blog/2019/09/data-scientists-guide-8-types-of-sampling-techniques/>.
- Gao, F., & Li, L. (2017). Examining a One-Hour Synchronous Chat in a Microblogging-Based Professional Development Community. *British Journal of Educational Technology*, 48(2), 332-347.
- Gao, Z., Feng, A., Song, X., & Wu, X. (2019). "Target-Dependent Sentiment Classification With BERT,". *In IEEE Access*, 7, 154290-154299
- Girard, J. M. (2015). Meaning of Feature Vector [Msg 1]. Retrieved from https://www.researchgate.net/post/in_simple_words_what_do_you_mean_by_feature_vector_in_image_processing.
- Github. (2019, June 18). [www.github.com](https://github.com/). Retrieved from: <https://github.com/>
- Gollapalli, S. D., & Ng, S. K. (2021). Modeling Emoji Generation for Emotion Analysis of Social Media Short Texts. *In ICWSM Workshops. Institute of Data Science, National University of Singapore, Singapore*
- Graves, A., & Jaitly, N. (2014). Towards End-To-End Speech Recognition with Recurrent Neural Networks. *In International conference on machine learning - PMLR*. (pp. 1764-1772).

- Grimmer, J., & Stewart, B. M. (2013). Text as data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 21(3), , 267–297
- Gunal, S. (2012). Hybrid Feature Selection for Text Classification. *Turk J Elec Eng & Comp Sci*, 20(Sup.2)
- Gupta, M. K., & Chandra, P. (2020). A Comprehensive Survey of Data Mining. *International Journal of Information Technology*, 12(4), 1243-1257.
- Gupte, A., Joshi, S., Gadgul, P., Kadam, A., & Gupte, A. (2014). Comparative Study of classification Algorithms Used in sentiment Analysis. *International Journal of Computer Science and Information Technologies*, 5(5), 6261-6264.
- Haddi, E., Liu, X., & Shi, Y. (2013). The Role of Text Pre-processing in Sentiment Analysis. *In Procedia Computer Science*, 17, 26–32
- Hansen, L., & Salamon, P. (1990). Neural Network Ensembles. *IEEE Trans. Pattern Analysis and Machine Intell. IEEE Trans. Pattern Analysis and Machine Intell*, 12, 993-1001.
- Hasan, B. M., & Abdulazeez, A. M. (2021). A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. *Journal of Soft Computing and Data Mining*, 2(1), 20-30.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. *Springer Series in Statistics 2009*.
- Hira, Z. M., & Gillies, D. F. (2015). A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data. *Advances in Bioinformatics*, vol. 2015, Article ID 198363, 13 pages, 2015.

- Hornak, C. (2009, May 05). Micro-blogging for businesses: *Benefits and tips*, Retrieved from <http://eyeflow.com/micro-blogging-for-businesses-benefits-and-tips/>.
- Hossin, M., & Sulaiman, M. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 1.5(2)
- Hovy, E. H. (2015). *What are Sentiment, Affect, and Emotion? Applying the Methodology of Michael Zock to Sentiment Analysis*. Switzerland:
- Hu, M., & Liu, B. (2004). *Mining and Summarizing Customer Reviews*. Department of Computer Science University of Illinois at Chicago .
- Iliou, T., Anagnostopoulos, C., Nerantzaki, M., & Anastassopoulos, G. (2015). A Novel Machine Learning Data Preprocessing Method for Enhancing Classification Algorithms Performance. *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS), ACM, September 2015*. At: RHODES, GREECE.
- Ishtiaque Ahmed, N., & Nasrin, F. (2022). Reducing Error Rate for Eye-Tracking System by Applying SVM. In *Machine Intelligence and Data Science Applications*. Springer, Singapore., 35-47.
- Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proc. of ECML-98, 10th European Conference on Machine Learning, Springer Verlag, Heidelberg, DE, 1998.*, (pp. 137-142).
- Jotheeswaran, J., & Koteeswaran, S. (2015). Sentiment Analysis: A Survey of Current Research and Techniques. *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, 3(5), 3749-3757.
- Jurafsky, D., & Martin, J. H. (2017). *Speech and Language Processing, 3rd Edition*.

- Kalaivani, P., & Shunmuganathan, K. L. (2013). Sentiment Classification of Movie Reviews by Supervised Machine Learning Approaches. *Indian Journal of Computer Science and Engineering (IJCSE)*, 4(4), 285-292.
- Kapoor, K. K., Tamilmani, K., Rana, N. P., Patil, P., Dwivedi, Y. K., & Nerur, S. (2018). Advances in Social Media Research: Past, Present and Future. *Information Systems Frontiers*, 20(3), 531-558.
- Kataria, M., & Shah, S. M. (2015). Extended Comprehensive Sentiment Analysis for Informal Opinion Text. *International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181*, 4(05), 481-483.
- Kham, N. N. (2019). Lexicon Based Emotion Analysis on Twitter Data. *Published in International Journal of Trend in Scientific Research and Development (ijtsrd)*, ISSN: 2456-6470, 3(5), 1008-1012. Retrieved from <https://www.ijtsrd.com/papers/ijtsrd26566.pdf>
- Khan, M. T., Durrani, M., Ali, A., Inayat, I., Khalid, S., & Khan, K. H. (2016). Sentiment Analysis and the Complex Natural Language. *Complex Adaptive Systems Modeling*, 4(2),
- Kim, A. J., & Johnson, K. K. (2016). Power of Consumers Using Social Media: Examining the Influences of Brand-Related User-Generated Content on Facebook. *Computers in Human Behavior*, 58,
- Kim, Y., & Chung, M. (2019). An Approach to Hyperparameter Optimization for the Objective Function in Machine Learning. *Electronics*, 8(11), 1267.
- Konez, P., & Paralic, J. (2011). An Approach to Feature Selection for Sentiment Analysis. *INES 2011 15th International Conference on Intelligent Engineering Systems. June 23–25, 2011, Poprad, Slovakia.*

- Korde, V., & Mahender, C. N. (2012). Text Classification and Classifiers: A Survey. *International Journal of Artificial Intelligence & Applications (IJAIA)*, 3(2), 85 – 99).
- Korjus, K., Hebart, M. N., & Vicente, R. (2016). An Efficient Data Partitioning to Improve Classification Performance While Keeping Parameters Interpretable. *PLoS ONE*, 11(8), e0161788.
- Krishna, K. M. (2019, July 8). One-Hot-Encoding, Multicollinearity and the Dummy Variable Trap – This Article Discusses about the Dummy Variable Trap Stemming from the Multicollinearity Problem. Retrieved from: <https://towardsdatascience.com/one-hot-encoding-multicollinearity-and-the-dummy-variable-trap-b5840be3c41a>.
- Kumar, R. (2011). Research Methodology 3rd Edition: A step-by-step guide for beginners.
- Kusumawati, R., D'arofah, A., & Pramana, P. A. (2019). Comparison Performance of Naive Bayes Classifier and Support Vector Machine Algorithm for Twitter's Classification of Tokopedia Services. In *Journal of Physics: Conference Series* 1320(1), 012016.
- Kwon, O., & Sim, J. M. (2013). Effects of data set features on the performances of classification algorithms. *Expert Systems with Applications*, 40(5), 1847-1857
- Li, X., Peng, Q., Sun, Z., Chai, L., & Wang, Y. (2017). Predicting social emotions from readers' perspective. *IEEE Transactions on Affective Computing*, 10(2), 255-264.
- Liu, A. Y., & Martin, C. E. (2011). Smoothing Multinomial Naïve Bayes in the Presence of Imbalance. *Machine Learning and Data Mining In Pattern Recognition. 7th International Conference, MLDM 2011, New York, NY, USA, August 30 – September 3, 2011. Proceedings*, (pp.46-59).

- Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Conference Paper: Synthesis Lectures on Human Language Technologies 5.1 (2012)*: (pp. 1-167).
- Lo, S. L., Cambria, E., Chiong, R., & Cornforth, D. (2017). Multilingual Sentiment Analysis: from Formal to Informal and Scarce Resource Languages. *Artificial Intelligence Review*, 48(4), 499-527.
- Maniruzzaman, M., Rahman, M., Ahammed, B., & Abedin, M. (2020). Classification and Prediction of Diabetes Disease Using Machine Learning Paradigm. *Health information science and systems*, 8(1), 1-14.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. *Cambridge*: Cambridge University Press.
- Mayer, Z. (2019, December 12). A Brief Introduction to caretEnsemble - caretEnsemble-intro. pp. Retrieved from Contributed Packages - The R Project for Statistical Computing. Retrieved from <https://cran.r-project.org/web/packages/caretEnsemble/vignettes/caretEnsemble-intro.html>.
- Mazumder, S. (2021, June 21). 5 Techniques to Handle Imbalanced Data For a Classification Problem. *Analytis Vidhya*. Retrieved from: <https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/>.
- Mehta, P., Pandya, S., & Kotecha, K. (2021). Harvesting Social Media Sentiment Analysis to Enhance Stock Market Prediction using Deep Learning. *PeerJ Computer Science*, 7, e476.
- Melville, P. (2003). *Creating Diverse Ensemble Classifiers*. The University of Texas at Austin.

- Mishra, A. (2020, June 24). Decoding Support Vector Machines. *Intuitively Understand how Support Vector Machines work*. Retrieved from: <https://towardsdatascience.com/decoding-support-vector-machines-5b81d2f7b76f>.
- Misra, R. (2019, May 1). Support Vector Machines — Soft Margin Formulation and Kernel Trick. *Learn Some of the Advanced Concepts that Make Support Vector Machine a Powerful Linear Classifier*. Retrieved from: <https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe>.
- Mohammad, S. M. (2017). Challenges in sentiment Analysis. *In A practical guide to sentiment analysis (pp. 61-83)*. Springer, Cham.
- Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29, 436-465
- Moudřík, J., & Neruda, R. (2015). Evolving Non-linear Stacking Ensembles for Prediction of Go Player Attributes. *2015 IEEE Symposium Series on Computational Intelligence*, 1673–1680.
- Naikoo, A. A., Thakur, S. S., Guroo, T. A., & Lone, A. A. (2018). Development of Society Under the Modern Technology- A Review. *Scholedge International Journal of Business Policy & Governance*. 05(1), 1-8..
- Namugera, F., Wesonga, R., & Jehopio, P. (2019). Text Mining and Determinants of Sentiments: Twitter Social Media Usage by traditional Media Houses in Uganda. *Comput Soc Netw* 6(3).
- Nasteski, V. (2017). An Overview of the Supervised Machine Learning Methods.

- Neethu, M. S., & Rajasree, R. (2013). Sentiment Analysis in Twitter Using Machine Learning Techniques. *In 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) (pp. 1-5)*. IEEE.
- Ngo, L. (2020). *Introduction to Factors in R. Factors Play a Crucial Role in Data Analysis. Learn how to create, subset, and compare them. Towards Data Science*. Retrieved from <https://towardsdatascience.com/introduction-to-factors-in-r-dd752f732c94>.
- Nguyen, T. H., & Shirai, K. (2015). Topic Modelling Based Sentiment Analysis on Social Media for Stock Market Prediction. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, (pp. 1354–1364.). Beijing, China, July 26-31, 2015. .
- Nguyen, V. H., Nguyen, H. T., Duong, H. N., & Snasel, V. (2016). *Recent Development in Intelligent Information and Database Systems. Studies in Computational Intelligence 642*,.
- Nielsen, F. A. (2011). A new: Evaluation of a Word List for Sentiment Analysis in Microblogs. *In CEUR Workshop Proceedings, 718* (pp. 93–98.). <http://arxiv.org/abs/1103.2903>.
- Osisanwo, F. Y., Akinsola, J. E., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised Machine Learning Algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3), 128-138.
- Oza, N. C., & Tumer, K. (2008). Classifier Ensembles: Select Real-World Applications. *Information Fusion*, 9(1)

- Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends® in information retrieval*, 2(1–2), 1-135.
- Patil, G., Galande, V., Kekani, V., & Dange, K. (2014). Sentiment Analysis Using Support Vector Machine. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(1)
- Pawar, A. B., Jawale, M. A., & Kyatanavar, D. N. (2016). Fundamentals of Sentiment Analysis: Concepts and Methodology. In *Sentiment Analysis and Ontology Engineering*. Springer, Cham. (pp. 25-48).
- Pratiwi, A. I. (2018). Adiwijaya, On the Feature Selection and Classification Based on Information Gain for Document Sentiment Analysis,. *Appl. Comput. Intell. Soft Comput*, 2018.
- Quinlan, J. (1996). Bagging, Boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*, (pp. 725-730).
- RamaKrishna, B. V., Rao, B. B., Rao, K. G., & Chandan, K. (2015). An Enumerative Framework for Extraction of Bag-Of-Words from Legal Documents. *Asian Journal of Computer Science And Information Technology*, 62-66.
- Rana, T. A., & Cheah, Y. N. (2016). Aspect Extraction in Sentiment Analysis: Comparative Analysis and Survey. *Artificial Intelligence Review*, 46(4), 459-483.
- Rashid, A., Anwer, N., Iqbal, M., & Sher, M. (2013). A Survey Paper: Areas, Techniques and Challenges of Opinion Mining. *IJCSI International Journal of Computer Science Issues*, 10(6)
- R Core Team (2017). R: A Language and Environment for Statistical Computing [Computer software manual]. Vienna, Austria. Retrieved from <https://www.R-project.org/>.

- R Core Team (2022). R: A language and Environment for Statistical Computing. *R Foundation for Statistical Computing, Vienna, Austria*. Retrieved from <https://www.R-project.org/>.
- Roberts, M. E., Stewart, B. M., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S. K., & Rand, D. G. (2014). Structural Topic Models for Open-Ended Survey Responses. *American Journal of Political Science*, 58(4), 1064–1082.
- Rocca, J. (2019, April 22). Ensemble Methods: Bagging, Boosting and Stacking. *Understanding the Key Concepts of Ensemble Learning*. Retrieved from: <https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205>.
- Rodriguez, J. (2019, March 29). The Three Pillars of Robust Machine Learning: Specification Testing, Robust Training and Formal Verification. Retrieved from: <https://www.linkedin.com/pulse/three-pillars-robust-machine-learning-specification-formal-rodriguez>.
- Rokach, L. (2005). Ensemble Methods for Classifiers. In: *Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook*. Springer, Boston, MA:
- Rokach, L. (2010). Ensemble-Based Classifiers. *Artif Intell Rev (2010) 33: 1*, 1–39.
- RStudio-Community. (2018, February 14). About the Machine Learning and Modeling category. Retrieved from: RStudio Community: Retrieved from <https://community.rstudio.com/c/ml/15>
- Sabri, B., & Saad, S. (2016). Arabic Sentiment Analysis with Optimal Combination of Features Selection and Machine. *Research Journal of Applied Sciences, Engineering and Technology*, 13(5), 386-393.

- Sagi, O., & Rokach, L. (2018). Ensemble Learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4), e1249.
- Saraee, M., & Bagheri, A. (2013). Feature Selection Methods in Persian Sentiment Analysis. *International Conference on Application of Natural Language to Information Systems. NLDB 2013: Natural Language Processing and Information Systems*.
- Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN COMPUT. SCI.* 2, 160 (2021). Retrieved from:
- Saugata, P. (2018). Ensemble Learning. Bagging, Boosting, Stacking and Cascading Classifiers in Machine Learning using SKLEARN and MLEXTEND Libraries. Retrieved from: <https://medium.com/@saugata.paul1010/ensemble-learning-bagging-boosting-stacking-and-cascading-classifiers-in-machine-learning-9c66cb271674>.
- Schapire, R. E. (1990). The Strength of Weak Learnability. *Machine Learning*, 5, 197-227.
- Schmidt, D., & Heckendorf, C. (2017, November 21). Guide to the N-gram Package. Fast n-gram Tokenization, Version 3.0.4. pp. Retrieved from: <http://cran.nexr.com/>: <http://cran.nexr.com/web/packages/ngram/index.html>
- Schrauwen, S. (2010). Machine Learning Approaches to Sentiment Analysis using the Dutch Netlog Corpus. *Computational Linguistics and Psycholinguistics Technical Report series*, CTRS-001, July 2010.
- Seijo-Pardo, B., Porto-Díaz, I., Bolón-Canedo, V., & Alonso-Betanzos, A. (2017). Ensemble Feature Selection: Homogeneous and Heterogeneous Approaches. *Knowledge-Based Systems*, 118, 124-139.

- Sethi, A. (2020, April 6). Supervised Learning vs. Unsupervised Learning – A *Quick Guide for Beginners*. *Analytis Vidhya*. Retrieved from: <https://www.analyticsvidhya.com/blog/2020/04/supervised-learning-unsupervised-learning/>.
- Shah, C. (2017). Social Media and Social Networking. In: Social Information Seeking. *The Information Retrieval Series, vol 38*. Springer, Cham.
- Sharma, N. (2019, January 13). Importance of Distance Metrics in Machine Learning Modelling. Retrieved from: <https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d>.
- Shirbhate, A. G., & Deshmukh, S. N. (2016). Feature Extraction for Sentiment Classification on Twitter Data. *International Journal of Science and Research (IJSR)*, 5(2), 2183 – 2189
- Silge, J., & Robinson, D. (2019, May 16). *Text Mining with R. A Tidy Approach*, pp. Retrieved from <https://www.tidytextmining.com/sentiment.html>.
- Silge, J., & Robinson, D. (2020). *Text mining with R: A tidy approach*. O'Reilly Media, Inc. Retrieved from: <https://www.tidytextmining.com/>.
- Singh, S. (2018). Understanding the Bias-Variance Trade-off. Retrieved from: <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>.
- Soong, H. C., Jalil, N. B., Ayyasamy, R. K., & Akbar, R. (2019). The essential of Sentiment Analysis and Opinion Mining in Social Media: Introduction and Survey of the Recent Approaches and Techniques. In *2019 IEEE 9th symposium on computer applications & industrial electronics (ISCAIE)*. IEEE. pp. 272-277.

- Strickland, J., & Chandler, N. (2017). How Twitter Works. *HowStuffWorks.com.*, pp. Retrieved from: <https://computer.howstuffworks.com/internet/social-networking/networks/twitter.htm>> .
- Tang, J., Alelyani, S., & Liu, H. (2015). *Data Classification: Algorithms and Applications.* pp. 498-500: Data Mining and Knowledge Discovery Series, CRC Press (2015).
- Tejwani, R. (2014). Sentiment Analysis: A survey. *arXiv preprint arXiv:1405.2584.*
- Tellez, E. S., Miranda-Jiménez, S., Graff, M., Moctezuma, D., Siordia, O. S., & Villaseñor, E. A. (2017). A Case Study of Spanish Text Transformations for Twitter Sentiment Analysis. *Expert Systems with Applications, 81*, 457-471.
- Thelwall, M., Buckley, K., & Paltoglou, G. (2011). Sentiment in twitter events. *Journal of the American Society for Information Science and Technology 62*(2), 406-418.
- Tripathy, A., Anand, A., & Rath, S. (2017). Document-Level Sentiment Classification using Hybrid Machine Learning Approach. *Knowledge and Information Systems.* 53.
- Tripathi, H. (2019, September 24). What Is Balanced And Imbalanced Dataset? – *Techniques To Convert Imbalanced Dataset Into Balanced Dataset. Analytis Vidhya.* Retrieved from: <https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5>.
- Trivedi, S. K., & Dey, S. (2013). Interplay Between Probabilistic Classifiers and Boosting Algorithms for Detecting Complex Unsolicited Emails. *Journal of Advances in Computer Networks, 1*(2), 132-136.

- Tuwe, L. (2015). On Effectively Creating Ensembles of Classifiers: *Studies on Creation Strategies, Diversity and Predicting with Confidence*. Stockholm University, Ph.D. thesis.
- Undhad, P. R., & Bhalodiya, D. J. (2017). Text Classification and Classifiers: A Comparative Study. *IJEDR* 5(2), 2043 – 2047.
- Viegas, F., Gonçalves, M. A., Martins, W., & Rocha, L. (2015). Parallel lazy semi-naive bayes strategies for effective and efficient document classification. *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, (pp. 1071-1080).
- Vijayarani, S., & Janani, R. (2016). Text Mining: Open Source Tokenization Tool – An Analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, 3(1)
- Vilares, D., Alonso, M. A., & Gómez-Rodríguez, C. (2017). Supervised Sentiment Analysis in Multilingual Environments. *Information Processing & Management*, 53(3), 595-607.
- Viola, P., & Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2), 137-154.
- Vosoughi, S., Zhou, H., & Roy, D. (2016). Enhanced Twitter Sentiment Classification Using Contextual Information. *arXiv preprint arXiv:1605.05195*.
- Wankhade, M., Rao, A., & Kulkarni, C. (2022). A Survey on Sentiment Analysis Methods, Applications, and Challenges. *Artif Intell Rev* (2022), Welbers, K., Atteveldt, W. V., & Benoit, K. (2017). Text Analysis in R. *11*(4) , 245–265.
- Wolpert, D. H. (1992). Stacked generalization. *Neural networks* 5(2), 241-259.

- Wolpert, D., & Macready, W. (1997). No Free Lunch Theorems for Optimization. *Evolutionary Computation, IEEE Transactions on on Evolutionary Computation* 1(1), 67-82.
- Workshop, r. T. (n.d). tif: Text Interchange Format. Retrieved from: <https://github.com/ropensci/tif>.
- Yadav, N., Kudale, O., Gupta, S., Rao, A., & Shitole, A. (2020). Twitter Sentiment Analysis using Machine Learning for Product Evaluation. *In 2020 international conference on inventive computation technologies (ICICT) IEEE*, (pp.181-185) .
- Yadollahi, A., Shahraki, A. G., & Zaiane, O. R. (2017). Current State of Text Sentiment Analysis from Opinion to Emotion Mining. *ACM Computing Surveys (CSUR)*, 50(2), 1-33.
- Yang, D., Zhang, D., Yu, Z., Yu, Z., & Zeghlache, D. (2014). SESAME: Mining User Digital Footprints for Fine-Grained Preference-Aware Social Media Search. *November 2014 ACM Transactions on Internet Technology* 14(1),
- Zhang, G. (2018, November 10). What is the kernel trick? Why is it important? Retrieved from: <https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d>.
- Zhang, L., & Liu, B. (2017). Sentiment Analysis and Opinion Mining. *In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA*
- Zhao, R., & Mao, K. (2017). Fuzzy Bag-Of-Words Model for Document Representation. *IEEE transactions on fuzzy systems*, 26(2), 794-804.

- Zheng, L., Wang, H., & Gao, S. (2018). Sentimental Feature Selection for Sentiment Analysis of Chinese Online Reviews. *International Journal of Machine Learning and Cybernetics*, 9(1), 75-84.
- Zhou, Z.-H. (2012). *Ensemble Methods - Foundations and Algorithms*. Chapman & Hall/CRC - Machine Learning & Pattern Recognition Series.
- Zhu, Y., Xie, C., Wang, G. J., & Yan, X. G. (2017). Comparison of Individual, Ensemble and Integrated Ensemble Machine Learning Methods to Predict China's SME Credit Risk in Supply Chain Finance. *Neural Computing and Applications*, 28(1), 41-50.
- Zin, H. M., Mustapha, N., Murad, M. A., & Sharef, N. M. (2017). The Effects of Pre-Processing Strategies in Sentiment Analysis of Online Movie Reviews. *AIP Conference Proceedings 1891, 020089 (2017)*::

APPENDICES

Appendix I: Data Mining and Sentiment Analysis Experiment Sheet

These codes were variously retrieved from RStudio-Community (2018) and Github (2019).

DATA MINING IN R-STUDIO

```
rawtweets = searchTwitter("OnePlus", n=2000, since = "2019-12-01", lang = "en")  
length.rawtweets <- length(rawtweets)  
length.rawtweets  
rawtweets.df <- lapply(rawtweets, function(t) t$toDataFrame())  
write.csv(rawtweets.df, "tweets.csv")
```

SENTIMENT ANALYSIS

```
txt = sapply(rawtweets, function(x) x$getText())  
txt1 = gsub("RT|via)((?:\\b\\w*@\\w+)+)", " ", txt)  
txt2 = gsub("http[^:blank:]", " ", txt1)  
txt3 = gsub("@\\w+", " ", txt2)  
txt4 = gsub("[[:punct:]]", " ", txt3)  
txt5 = gsub("[^[:alnum:]]", " ", txt4)  
write.csv(txt5, "tweets1.csv")  
txt6 <- Corpus(VectorSource(txt5))  
txt6 <- tm_map(txt6, content_transformer(tolower))  
txt6 <- tm_map(txt6, removeWords, stopwords("english"))  
txt6 <- tm_map(txt6, stripWhitespace)
```

```

txt6 <- tm_map(txt6, stemDocument)

for (i in 1:50) {

  cat(paste("[", i, "] ", sep = ""))

  writeLines(as.character(txt6[[i]]))

}

library(rJava)

library(RWeka)

token_delimtr <- " \\t\\r\\n.!?,;\\\"()"

NgramTokenizer <- function(min, max) {

  result <- function(x){

    RWeka::NGramTokenizer(x, RWeka::Weka_control(min = min, max = max,
delimiters=token_delimtr))

  }

  return(result)

}

textTDM <- TermDocumentMatrix(txt6, control=list(tokenize=NgramTokenizer(min=1,
max=1)))

inspect(textTDM)

textTDM_sparse = removeSparseTerms(textTDM, 0.98)

textTDM_sparse

dtm <- TermDocumentMatrix(txt6)

m <- as.matrix(textTDM_sparse)

v <- sort(rowSums(m),decreasing=TRUE)

```

```

d <- data.frame(word = names(v),freq=v)

head(d, 50)

str(dtm)

features <- sort(rowSums(as.matrix(dtm)), decreasing=TRUE)

barplot(features[1:25], col="sky blue", las=2, main = "Most Frequent Words", ylab =
"Frequencies")

str(features)

summary(features)

length(features)

head(features, 10)

tail(features, 10)

library(tm)

library(SnowballC)

library(wordcloud)

library(RColorBrewer)

pal <- brewer.pal(8, "Dark2")

png("wordcloud_packages.png", width=1000,height=1000)

wordcloud(txt6, min.freq = 2, max.words=100, random.order=FALSE, rot.per=0.35,
          colors=pal)

get_sentiments("afinn")

get_sentiments("bing")

get_sentiments("nrc")

library(ggplot2)

```

```

library(syuzhet)

library(tidytext)

library(dplyr)

mysentiment <- get_nrc_sentiment(txt5)

SentimentScores <- data.frame(colSums(mysentiment[,]))

names(SentimentScores) <- "Score"

SentimentScores <- cbind("sentiment" = rownames(SentimentScores), SentimentScores)

rownames(SentimentScores) <- NULL

View(SentimentScores)

write.csv(mysentiment, "mysentiment_apendix.csv")

mysentiment

str(mysentiment)

colnames(mysentiment)

summary(SentimentScores[c("sentiment", "Score")])

library(textdata)

nrc_lexicon <- get_sentiments("nrc")

head(nrc_lexicon)

ggplot(data = SentimentScores, aes(x = sentiment, y = Score)) +
  geom_bar(aes(fill = sentiment), stat = "identity") +
  theme(legend.position = "none") +
  xlab("Sentiment") + ylab("Score") +
  ggtitle("Total Sentiment (Emotion) Score Based on Tweets")

```

```

bing_lexicon <- get_sentiments("bing")

head(bing_lexicon)

get_sentiments("nrc") %>%

  filter(sentiment %in% c("positive",
                        "negative")) %>%

  count(sentiment)

get_sentiments("bing") %>%

  count(sentiment)

library(tidytext)

dtm_tidy <- tidy(dtm)

head(dtm_tidy, 15)

tail(dtm_tidy, 15)

colnames(dtm_tidy)

colnames(dtm_tidy)[colnames(dtm_tidy)=="term"] <- "word"

colnames(dtm_tidy)

nrc_anticipation <- get_sentiments("nrc") %>%

  filter(sentiment == "anticipation")

nrc_anticipation

dtm_tidy %>%

  semi_join(nrc_anticipation) %>%

  count(word, sort = T)

dtm_tidy_sentiments <- dtm_tidy %>% inner_join(get_sentiments("bing"), by =
c(word="word"))

```



```

str(dtm_tidy_sentiments)

summary(dtm_tidy_sentiments)

head(dtm_tidy_sentiments, 15)

tail(dtm_tidy_sentiments, 15)

countclass <- length(which(dtm_tidy_sentiments == 'positive'))

countclass

countclass <- length(which(dtm_tidy_sentiments == 'negative'))

countclass

library(naivebayes)

library(dplyr)

library(ggplot2)

library(psych)

dtm_tidy_sentiments

str(dtm_tidy_sentiments)

dtm_tidy_sentiments$word <- unlist(dtm_tidy_sentiments$word)

dtm_tidy_sentiments$document<- unlist(dtm_tidy_sentiments$document)

dtm_tidy_sentiments$count <- unlist(as.character(dtm_tidy_sentiments$count))

dtm_tidy_sentiments$sentiment<- unlist(dtm_tidy_sentiments$sentiment)

classifiedsentiments.df <- dtm_tidy_sentiments

classifiedsentiments.df

write.csv(classifiedsentiments.df, "classified_sentiments_df.csv")

classifiedwords_df<- read.csv("classified_sentiments_df.csv", stringsAsFactors =
FALSE)

```

```
glimpse(classifiedwords_df)

set.seed(1)

classifiedwords_df <- classifiedwords_df[sample(nrow(classifiedwords_df), )
classifiedwords_df <- classifiedwords_df[sample(nrow(classifiedwords_df), )
glimpse(classifiedwords_df)

head(classifiedwords_df)

tail(classifiedwords_df)

sample_index <- sample(1:nrow(classifiedwords_df), 30, replace = FALSE)

sample_index

classifiedwords_df[sample_index, ]

classifiedwords_df$sentiment <- as.factor(classifiedwords_df$sentiment)

classifiedwords_df$word <- as.factor(classifiedwords_df$word)

glimpse(classifiedwords_df)

str(classifiedwords_df$sentiment)

classifiedwords_df

write.csv(classifiedwords_df, "classifiedwords_df_apendix.csv")

classifiedterms_df[,]

levels(classifiedwords_df$sentiment)

table(classifiedwords_df$sentiment)

barplot(table(classifiedwords_df$sentiment))

proportional_table <- table(classifiedwords_df$sentiment)

proportional_table
```

```
round(prop.table(proportional_table)*100,3)# output in percentage
```

MODEL TRAINING

```
ind <- sample(2, nrow(classifiedwords_df), replace = TRUE, prob = c(0.67, 0.33))
```

```
nbtrain_set <-classifiedwords_df[ind == 1,]
```

```
nbtest_set <- classifiedwords_df[ind == 2,]
```

```
head(nbtrain_set)
```

```
head(nbtest_set)
```

```
nrow(nbtrain_set)
```

```
nbtrain_set
```

```
write.csv(nbtrain_set, "nbtrainset_apendix.csv")
```

```
nb_model <- naiveBayes(sentiment ~ word, data = nbtrain_set)
```

```
nb_model
```

```
summary(nb_model)
```

```
nrow(nbtest_set)
```

```
nbtest_set
```

```
write.csv(nbtest_set, "nbtestset_apendix.csv")
```

```
pred_nb <- predict(nb_model, nbtest_set)
```

```
pred_nb
```

```
confusionMatrix(table(pred_nb, nbtest_set$sentiment))
```

```
pred_nb <- predict(nb_model, nbtest_set, type = "raw")
```

```
pred_nb
```

```
head(pred_nb, 20) # this returns a view of the first 20 rows of the predicted model
```

```

write.csv(pred_nb, "prednbprobabilites_apendix.csv")

head(nbtest_set)

nbtest_set

nb_model2 <- naiveBayes(sentiment ~ word, data = nbtrain_set, laplace = 1)

nb_model2

pred_nb2 <- predict(nb_model2, nbtest_set)

pred_nb2

confusionMatrix(table(pred_nb2, nbtest_set$sentiment))

classifiedwords_df

head(classifiedwords_df)

summary(classifiedwords_df)

classifiedwords_df$sentiment

dummysvMM <- model_matrix(classifiedwords_df, word ~ sentiment )

dummysvMM

summary(dummysvMM)

dummysvMM <- model_matrix(classifiedwords_df, word ~ sentiment-1 )

dummysvMM

write.csv(dummysvMM, "dummysvMM_1.csv")

classifiedwords_df2 <- cbind(classifiedwords_df[,1:5], dummysvMM)

head(classifiedwords_df2)

write.csv(classifiedwords_df2, "dummysvMM_2.csv")

classifiedwords_df

```

```

classifiedwords_df3 <- classifiedwords_df[,c(2,3,4,5)]

classifiedwords_df3

write.csv(classifiedwords_df3, "classifiedwords_df3.csv")

dataset_svm <- read.csv("classifiedwords_df3.csv",header = FALSE)

dataset_svm

dataset_svm$v5[dataset_svm$V5 == "negative" ] <- 0

dataset_svm$v5[dataset_svm$V5 == "positive" ] <- 1

dataset_svm

write.csv(dataset_svm, "dataset_svm.csv")

dataset_svm1 <- dataset_svm[-1,c(2,3,4,6)]

dataset_svm1

str(dataset_svm1)

write.csv(dataset_svm1, "dataset_svm1.csv")

dataset_svm1$v5 = factor(dataset_svm1$v5, levels = c(0, 1))

str(dataset_svm1)

ind <- sample(2, nrow(dataset_svm1), replace = TRUE, prob = c(0.67, 0.33))

trainsv_set <- dataset_svm1[ind == 1,]

testsv_set <- dataset_svm1[ind == 2,]

head(trainsv_set)

head(testsv_set)

str(trainsv_set)

write.csv(trainsv_set, "trainsv_set.csv")

```

```

write.csv(testsv_set, "testsv_set.csv")

set.seed (1)

x=matrix (rnorm (20*2) , ncol =2)

y=c(rep (0,10) , rep (1 ,10) )

x[y==1 ,]= x[y==1,] + 1

plot(x, col = y + 3, pch = 19)

library (e1071)

sv_model = svm(formula = v5 ~ .,
               data = trainsv_set,
               type = 'C-classification',
               kernel = 'linear',
               scale =FALSE)

sv_model

sv_model = svm(formula = v5 ~ .,
               data = trainsv_set,
               type = 'C-classification',
               kernel = 'linear',cost =10,
               scale =FALSE)

sv_model

sv_model$index

sv_model$SV

summary (sv_model )

```

```

sv_model = svm(formula = v5 ~ .,
               data = trainsv_set,
               type = 'C-classification',
               kernel = 'linear',cost =0.1,
               scale =FALSE)

sv_model

sv_model$index

set.seed(1)

tune.out <- tune(svm ,v5 ~ .,
               data = trainsv_set,
               type = 'C-classification',
               kernel = 'linear',
               ranges =list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100) ))

summary(tune.out)

best_model <- tune.out$best.model

summary(best_model)

sv_pred <- predict (best_model ,testsv_set )

sv_pred

table(predict =sv_pred , actual= testsv_set$v5 )

sv_model = svm(formula = v5 ~ .,
               data = trainsv_set,
               type = 'C-classification',

```

```

        kernel = 'linear',cost =.01,
        scale =FALSE)

sv_model

sv_pred=predict (sv_model ,testsv_set )

table(predict =sv_pred , actual= testsv_set$v5 )

sv_model = svm(formula = v5 ~ .,
                data = trainsv_set,
                type = 'C-classification',
                kernel = 'linear',cost =1e5,
                scale =FALSE)

summary(sv_model)

sv_model = svm(formula = v5 ~ .,
                data = trainsv_set,
                type = 'C-classification',
                kernel = 'linear',cost =1,
                scale =FALSE)

summary (sv_model)

rmse

mae

qae

library(parallel)

library(caret)

```



```

library(nnet)

library(e1071)

library(caretEnsemble)

library(parallel)

library(doParallel)

numCores <- detectCores()

classification_error <- function(conf_mat) {

  conf_mat = as.matrix(conf_mat)

  error = 1 - sum(diag(conf_mat)) / sum(conf_mat)

  return (error)

}

library(klaR)

control <- trainControl(method="repeatedcv", number=10, repeats=3,
savePredictions=TRUE, # To save out of fold predictions

                        classProbs=TRUE) # To save the class probabilities of the out of fold
predictions

set.seed(123)

nbmodel <- train(as.factor(sentiment)~ word, data = nbtrain_set, method = "nb",

                trControl=control)

nbmodel

summary(nbmodel)

predicted.classes <- nbmodel %>% predict(nbtest_set)

predicted.classes

```

```

confusionMatrix(predicted.classes, nbtest_set$sentiment )

nb_conf_mat <- table(predicted = predicted.classes, actual = nbtest_set$sentiment )

nb_conf_mat

cat("Classification Error for nbmodel for Sentiment Predictions:",
classification_error(nb_conf_mat), "\n")

print(nbmodel)

write.csv(predicted.classes, "predicted.classes.csv")

mean(predicted.classes == nbtest_set$sentiment)

str(trainsv_set)

head(trainsv_set)

levels(trainsv_set$v5) <- c("negative", "positive")

head(trainsv_set)

head(testsv_set)

levels(testsv_set$v5) <- c("negative", "positive")

head(testsv_set)

str(trainsv_set)

str(trainsv_set)

anyNA(trainsv_set)

sample_n(trainsv_set, 3)

summary(trainsv_set)

head(trainsv_set)

svcontrol <- trainControl(method="repeatedcv", number=10, repeats=3,
savePredictions=TRUE, classProbs=TRUE)

```

```

grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))

set.seed(3233)

svm_Linear_Grid <- train(v5 ~ V2, data = trainsv_set, method = "svmLinear",
                        trControl= svcontrol,
                        preProcess = c("center", "scale"),
                        tuneGrid = grid,
                        tuneLength = 10)

svm_Linear_Grid

plot(svm_Linear_Grid)

test_pred_grid <- predict(svm_Linear_Grid, newdata = testsv_set)

test_pred_grid

write.csv(test_pred_grid, "test_pred_grid.csv")

mean(test_pred_grid == testsv_set$v5)

confusionMatrix(test_pred_grid, testsv_set$v5 )

svm_conf_mat <- table(predicted = test_pred_grid, actual= testsv_set$v5 )

svm_conf_mat

cat("Classification Error for SVM Model Sentiment Predictions:",
classification_error(svm_conf_mat), "\n")

print(svm_Linear_Grid)

library(parallel)

library(caret)

library(nnet)

library(e1071)

```

```

library(caretEnsemble)

enscontrol <- trainControl(method="repeatedcv", number=10, repeats=3,
savePredictions=TRUE, classProbs=TRUE)

algorithmList <- c("nb", "svmLinear")

models <- caretList(v5 ~ V2, data=trainsv_set, trControl=enscontrol,
methodList=algorithmList)

models

ens_pred <- predict(models, newdata = testsv_set)

ens_pred

write.csv(ens_pred, "ens_pred.csv")

xyplot(resamples(models))

results <- resamples(models)

summary(results)

dotplot(results)

modelCor(results)

splom(results)

stackControl=trainControl(
  method="repeatedcv",
  number=10,
  repeats=3,
  savePredictions=TRUE,
  classProbs = TRUE,
  summaryFunction=twoClassSummary

```

```

)

stacked_ensemble <- caretStack(

  models,

  method="glm",

  metric="ROC",

  trControl=stackControl

)

stacked_ensemble

summary(stack_ensemble)

stack_prediction <- predict(stacked_ensemble, newdata = testsv_set)

stack_prediction

write.csv(stack_prediction, "stack_prediction.csv")

mean(stack_prediction == testsv_set$v5)

confusionMatrix(stack_prediction, testsv_set$v5 )

stack_ens_confmat<- table(predicted = stack_prediction, actual= testsv_set$v5 )

stack_ens_confmat

cat("Classification Error for the Stacked Ensemble Model for Sentiment Predictions:",
classification_error(stack_ens_confmat), "\n")

print(stack_ensemble)

```

