Adaptive Classification Method in Machine Learning for Detecting and Filtering Fake Content

by

Roll: 170024

A Thesis Report submitted to the Institute of
Information Technology
in partial fulfillment of the requirements for the degree of
Master of Science

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342
December 13, 2018

DECLARATION

I hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Roll: 170024

CERTIFICATE

This is to certify that the thesis entitled **Adaptive Classification Method in Machine Learning for Detecting and Filtering Fake Content** has been prepared and submitted by **Roll: 170024** in partial fulfillment of the requirement for the degree of Master of Science (M.Sc.) in Information Technology on December 13, 2018.

Supervisor

Accepted and approved in partial fulfillment of the requirement for the degree Master of Science (M.Sc.) in Information Technology.

K M Akkas Ali
Chairman

Jesmin Akhter
Member

Dr. Md. Abu Yousuf
Member

Prof. Dr. Md. Mahbub Alam
Member (External)

ACKNOWLEDGEMENTS

I feel pleased to have the opportunity of expressing my heartfelt thanks and gratitude to those who all rendered their cooperation in making this report.

This thesis is performed under my supervisor from Institute of Information Technology (IIT), Jahangirnagar University, Savar, Dhaka. During the work, my supervisor has supplied us a number of books, journals, and materials related to the present investigation. Without that help, kind support and generous time spans has been given, I could not perform the thesis work successfully in due time. First and foremost, I wish to acknowledge our profound and sincere gratitude to my supervisor for guidance, valuable suggestions, encouragement and cordial cooperation.

Moreover, I would also like to thank the other faculty members of IIT who have helped me directly or indirectly by providing their valuable support in completing this work.

I express my gratitude to all other sources from where I have found help. I am indebted to those who have helped me directly or indirectly in completing this work.

Last but not least, I would like to thank all the staff of IIT, Jahangirnagar University and our friends who have helped me by giving their encouragement and cooperation throughout the work.

December 2018.

ABSTRACT

In our modern era where internet is ubiquitous, everyone relies on various online resources for news. Along with the increase in use of social media platforms like Facebook, Twitter etc. news spread rapidly among millions of users within a very short span of time. The spread of fake news has far reaching consequences like creation of biased opinions to sway election outcomes for the benefit of certain candidates. Moreover, spammers use appealing news headlines to generate revenue using advertisements via click-baits. In this project, we aim to perform a six different classification techniques of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning. We investigate and compare these techniques with the parameter of accuracy, precision, recall and F-measure using different numbers of training data. Then we find the best classifier method based on accuracy and F-measure. We also discuss related research areas, open problems, and future research directions for fake news detection.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

TF	Term Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
SVM	Support Vector Machine
PPDM	Privacy Preserving Data Mining
ML	Machine Learning
NLP	Natural Language Processing
CNN	Convolutional Neural Network
KNN	K Nearest Neighbor
BoW	Bag-of-word
t-SNE	t-distributed Stochastic Neighbor Embedding
RBF	Radial Basis Function
RC	Red Circle
GS	Green Square
BS	Blue Star
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
P	Precision
R	Recall
A	Accuracy

CHAPTER I

Introduction

1.1 Overview

In recent times, Online content has been playing a significant role for users in their daily life with their decisions and opinions. Users can review any online market like e-commerce to help with gaining other users getting good products. Recently opinion spam does not only exist in product reviews and customers feedback, in fact fake news and misleading articles is another type of opinion spam. Social media like Facebook, Google Plus, Twitter and other social media outlet are one of the biggest source of spreading fake news and rumors [5] [6] [7].

The detection of false content has recently sparked growing interest among the general public and researchers as the circulation of erroneous information increases online, particularly in media such as social media, news blogs and online newspapers. For instance, a recent report found that Facebook referrals accounted for 50% of the total traffic to fake news sites and 20% total traffic to reputable websites. Since the majority of U.S. adults 62% gets news on social media [8].

Although the problem of false news is not a new problem, it is believed that the detection of false news is a complex task given that humans tend to believe misleading information and lack of control over the dissemination of false content [9]. False news has been receiving more attention in recent years. It is difficult for humans to detect false news. It can be argued that the only way a person can manually identify false news is to have a vast knowledge of the subject covered. Even with knowledge,

it is considerably difficult to identify successfully if the information in the article is real or false. The open nature of the web and social networks, in addition to the recent advance in information technology, simplifies the process of creating and disseminating false news. While it is easier to understand and track the intent and impact of false reviews, the intent and impact of creating propaganda by spreading false news cannot easily be measured or understood. For example, it is clear that the fake review affects the product owner, the customer and the online stores; On the other hand, it is not easy to identify the entities affected by the false news. This is because the identification of these entities requires measuring the spread of news, which has proven to be complex and requires many resources [10]. Trend Micro, a cyber security company, analyzed hundreds of fake news service providers around the world. They reported that it is easy to buy one of those services. In fact, according to the report, it is much cheaper for politicians and political parties to use those services to manipulate the results of elections and the opinions of people on certain issues [11] [12]. It is believed that the detection of false news is a complex task and much more difficult than the detection of false product reviews, since they are easily spread using social networks and word of mouth.

Fake content is flourishing on the Internet [13]. The motivations for creating false content are diverse, for example:

- many spam emails and comments from spam blogs are completed with random texts to avoid being detected by conventional methods, such as hashing;
- many spam Websites was designed to automatically generate thousands of interconnected web pages on a selected topic, in order to reach the top of the search engine response lists [14];
- many generators of false friends are available to improve the specialty of social networks [15].

False news has been around for decades and is not a new concept. However, the beginning of the era of social media, which may approach the beginning of the 20th

century, has aggravated the generation and circulation of false news in many aspects. False news can be explained simply as a piece of an article that is usually written for economic, personal or political purposes. The detection of these false news articles is possible through the use of several NLP techniques, machine learning and artificial intelligence [16].

1.2 Motivation

The amount of fake information is increasing and spreads to more and more topics, but overall, the more technical and complex a topic is, the harder it is to produce false claims and information for it. The fakes produced changes just as the normal news changes, and is often based on the same topics. For example, during the United States presidential election in 2016, massive amounts of political news were published and spread, and therefore the amount of politically loaded fakes were also increasing [17]. Fake news is increasing, but the fight against it is also increasing, and the overall awareness about it and how to spot it as well. Tools are needed as they evolve, both to minimize, but also to combat it. This change in definition and usage of fake news in the public eye is something that might lead to the misconception about what fake news actually are. Because of this, the researchers working with fake news wants to change the wording of this kind of misinformation, as it entails so much more than just news, and it is not only news that can be fake. A more definite wording is needed to be able to discuss the different aspects better. When looking at how people are accessing news, we can see that social media are one of the most prominent news sources in the world. Combining this with the fact that false information is spreading fastest on said social media sites, it is vital that the issue is taken seriously and either stopped or mitigated. It is spreading the fastest in topics and industries where there is a lot of emotion and points of view, where readers have a slight tendency to believe information that solidifies their existing views, even though they can be based on false information. That's why we are doing such research to identify the fake news and the best methodology to find out and prevent such corruption.

1.3 The challenge of detecting fake content

The information revolution has dramatically disrupted the news industry. Where we used to rely on media organizations to distribute news and other information, we now get our content from multiple sources too numerous to name. While this has given us access to a wider range of information, it has more than a few unintended consequences. Media organizations have always been the guardians of the content. Regulated, as they are, by codes of ethical conduct and laws that impose penalties for publishing false content, most media organizations have safeguards that guarantee the accuracy of the content they disseminate. This is the reason why, although for some time it has been possible to document images with impunity, we seldom see fake photographs in the news, as journalists must take care to ensure the accuracy of the news they report [18].

All this is changing as we speak. Internet distribution has ensured that much of the content that reaches us does so without having first passed through traditional guardians. Thanks to the cascade of information, we no longer have the bandwidth to verify the news we receive; It is much easier to trust what we are told to verify, even if the information we receive contradicts what we believe is true. As a result, we believe without doubt much of what is transmitted to us through social networks. As more and more people share information recklessly, falsehoods now receive the same level of seriousness as the facts. At the same time, digital spoofing has improved to the point where it has been possible to generate totally credible digital fakes in virtually any medium. Technologies such as generative confrontation networks (GAN) use two neural networks, one to generate a false image and the other to evaluate whether the first neural network managed to create an adequate image. Working iterative and together, GAN technologies are capable of producing digital counterfeits that are virtually impossible to detect.

As a result, it is now possible for digital manipulators to put words in the mouths of public figures and generate, on the fly, video sequences that seem completely genuine when in fact they are totally invented. This technology is called deep fakes and,

although currently the porn industry uses it mainly to generate false celebrity videos, it is not hard to imagine how these techniques, in the hands of the unscrupulous, could be used for extortion and defamation. And the false propaganda.

Both phenomena are about to combine with a devastating effect. We have already begun to see an increase in the frequency of cascades of information that avoid the guardians of the truth and the effect it is having on our credibility. At the same time, new technologies capable of generating fiction that can not be distinguished from facts are becoming viable enough to be widely implemented. Governments around the world are grappling with this problem by doing away with the platforms through which content is disseminated. However, it is not clear to me what these platforms are expected to do. The really good deep fakes are indistinguishable from the truth and it will be practically impossible for platforms, on their own, to distinguish truth from falsehood. In theory, it should be possible to use the same neural network technologies that created deep fakes in the first place to develop forensic techniques that can detect fake content. However, most experts agree that it is easier said than done. All that is needed to evade detection is that the false technologies are trained exactly on what the new forensic techniques are detecting to be able, very quickly, to learn to evade these measures.

If our reality is capable of being distorted so easily, perhaps the answer lies in finding a non-repudiate way of establishing the truth. One approach could be to create immutable life records that prove, beyond the shadow of a doubt, what really happened in a person's life minute by minute. It should be possible to achieve this by using technologies that are already around us: a combination of wearable, block chain, cloud computing, remote sensors, etc. When combining entries from multiple sources in a tamper-proof format, it should be possible to establish a record. Of life that can rebut false news. To make all this more efficient, the immutable life record can be made accessible through the APIs (application program interfaces) so that social services and other platforms that disseminate content can dynamically verify the content that leads to the true record of the life of a certain person. When the content matches the life record, the platforms can continue transporting it. When

we do not, we can prevent that content from showing a false image of a person's life. The problem with this approach is that it involves a considerable sacrifice of privacy. Complete records of life, if committed, could have a devastating effect on a person's personal life. That said, given the alternative, this could be a bargain that public personalities, who otherwise have a lot to lose, may feel it's worth doing.

1.4 Our Proposed System and contributions

We present in this paper an N-gram and tf-idf feature extraction based approach to detect fake content which consist of data preprocessing using stop words removal, tokenization, stemming, lemmatization and create a model with six machine classifier methods. We study, investigate and compare six machine classifier methods namely, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), Naive Bayes (NB) and Long Short-Term Memory (LSTM). We find the best classifier methods using the parameters of accuracy, precision, recall and F-measure using different numbers of training data. In this paper, our contribution is to find the best classifier methods based on accuracy and F-measure.

1.5 Problem Statement

Some approaches work on the text for detecting fake content, but there is no clear classify model for detecting fake content. Careful features selection is very important for training of machine learning model. All the detection techniques are capable of detecting spam up to some extent but if we rely on single method, spammers will nd the way to bypass it. So we have to use the adaptive methods of all possible techniques to detect phishing content. In the other hand, all the techniques are not for the best solution for detecting fake content. So we need to specify that which method or technique is suitable for that particular spam and then we need to find that method.

1.6 Objectives

The main objectives of this paper is to detect fake content with adaptive machine learning methods in web as the spammers find a new bypass way to attack spam in web and also finding the best classifier in this context.

1.7 Socio-Economic Advantages

News frightens us because scary stories sell. Then we get isolated and fearful and we rely more on the news because were afraid and so we need to know whats happening out there in the scary world. The more paranoid we get; the more money the news makes. Its all just shock, entertainment, or gossip with a pretense of being important and creating a well-informed citizenry. So, fake news is being viral in our society through Facebook, News Papers, Twitter etc. If we can detect fake news we may protect our society from the scary stories. False information is dangerous because of its ability to affect public opinion and electoral discourse. The advantages of detecting fake news are

- to reduce frightening people from the scary stories
- to reduce ambiguity from the false news and minimize harassment people as well
- to avoid unwanted content and also safe time
- to secure any account from malicious bots
- some fake news is affected directly to a persons real life; it will be very helpful if he/she know that news was fake.

1.8 Assumption and Limitation

The challenge for fake news detection comes with the democratization of news sources, and how easy modern technology makes sharing news articles in the age of

social media. But every news is not true, there are some news those are false. There is also some fake news, machine cannot detect them. Machine can detect that fake news that machine is trained. Otherwise it cannot detect. This is the huge limitations of detecting spam or fake content or related content in machine learning language. There are some other limitations given below

- Machine cannot detect 100% accuracy of fake content; it can give near 100% accuracy of fake content.
- Sometimes the accuracy may be shown the wired number like 50%, then we cannot take any decision whether it is fake or not.
- As the internet is changing rapidly and adding some new feature day by day, so it is very difficult to detect any new fake content that is not trained by the machine.

1.9 Organization of Thesis

Rest of the thesis is organized as follows

Chapter 2: Literature Review This chapter provides the researches which are the summary of related works.

Chapter 3: Proposed Approach and Model This chapter provides which methods and models we use in our simulation and describes short description and why we use these methods and models.

Chapter 4: Experiment and Evaluation Result This chapter provides the readers how we simulate our system and the final result.

Chapter 5: Conclusion & Future Work The concluding remarks about this research are presented in this chapter.

Finally, we add some references and coding summary in appendix section.

CHAPTER II

Literature Review

2.1 Introduction

Many researchers interest in the field of fake content or spam detection in machine learning. Many research and a lot of work has been done for detecting fake content [19] [20] [21]. Recently researchers proposed different model to detect fake content and the accuracy for detecting fake content was good. They proposed one or two feature extraction methods and one or two classifier. If we get better accuracy for detecting fake content, we need to focus on feature extraction methods. Then we need to find the best classifier model that provide the best accuracy result. In this paper, we are focusing on feature extraction and then finding the best classifier for detecting and filtering the fake content.

2.2 Related Work

The paper presented by Thomas Lavergne, Tanguy Urvoy and Francois Yvon describe the effectiveness of n-gram language models as fake content detectors is a consequence of their ability to capture short-range semantic and syntactic relations between words: fake contents generated by word stuffing or second order models fail to respect these relations. They introduced two approaches to detect fake content.

The approach of the language model, which yields fairly good results, and a new technique, using relative degrees of entropy, which resulted in improved results against advanced generators such as the Marcovian text generators [22]. They have shown that it is possible to detect generated texts that are considered natural enough that they cannot be detected using standardized method tests, yet vary greatly with one another so that they cannot be identified in plagiarism detection schemes. These methods were validated using a domain-independent model based on Google's n-gram, which resulted in the discovery of a highly false content detector. The techniques presented in this paper seem to bridge a gap between plagiarism detection schemes, and stylistics detection systems. As such, they might become part of standard anti spam tool kits.

Benjamin Riedel, Isabelle Augenstein, Georgios P.Spithourakis, Sebastian Riedel describes baseline for the fake news challenge stance detection task [23]. They have only focused on two simple bag of words like term frequency (TF) and term frequency-inverse document frequency (TF-IDF). They have specified some news articles and divide the articles into two parts, one body another one is the headline. They conclude that, although their system works satisfactorily in general, this can be attributed mainly to the almost perfect classification of the cases in "related" and "unrelated" body / title pairs (accuracy: 96.55%) and the values more or less predetermined discuss' classification of 'related' instances. The performance of your system with respect to the "agreement" label is, at best, average, while the accuracy of the system in the "disagree" test examples is clearly quite poor. The disappointing performance is remarkable since these two labels are possibly the most interesting in the FNC-1 task and the most relevant for the higher objective of automating the posture evaluation process.

In [24], they use language signaling techniques and a network analysis approach to design a fake primary news detector that provides high accuracy in terms of classi-

fication functions. They propose a hybrid system whose characteristics are included such as multi layered language processing and the addition of network behavior. In [25], they propose a method of detecting deceptive evidence online using a logistic regression class based on POS tags extracted from a set of real and deceptive texts, up to a 72% accuracy that can be further improved by performing cross-sectional analysis of classification models and reducing the vector size of input properties.

In [26] Rubin et al. it suggests that there are at least three different subtasks in the detection of false news: a) fabrication, b) deception and c) satire detection.

- Serious Fabrications exposed fraudulent journalistic writings, discussed in Compton & Benedetti (2015) or Shingler, are ideal for a corpus of false news. It takes a long time and it is tedious to collect original copies of invented news without covering; and the resulting bodies may have a limited size for NLP. The tabloids and the tabloids present a wide spectrum of new and unverified titles that use flashy headlines, exaggerations, scandal evasions or sensationalism to increase traffic or profits. The tabloids specifically emphasize such topics as sensational crime stories, astrology, gossip columns about celebrities and junk food news (Tabloides, 2015). Yellow journalism is an adequate source of false news in cases of obvious or overt falsification, fabrication or exaggeration, and may require investigation.

- Deception is another type of fabrication or deliberate counterfeiting in the mainstream or social media. Attempts to deceive audiences are disguised as news, and can be picked up and erroneously validated by traditional news media.

- They distinguish news made serious of humorous. If readers are aware of the humorous intention, they may no longer be predisposed to take the information at face value. Technology can identify humor and prominently display source sources (eg, The Onion) to alert users, especially in de-contextualized news

aggregators / platforms. News satire sites, news parodies, also known as fake news (eg, The Onion and CBC's This is That) are a specific genre with numerous sites presenting news "in a format typical of conventional journalism but that they depend to a large extent on irony and deadpan humor to emulate a genuine news source, which imitates credible news sources and stories, and which often reaches a wide distribution" (News Satire, 2015).

Nhauo Davuth and Sung-Ryul Kim [27] describes the classifications of malicious domain names using support vector machine and bi-gram method. Its purpose is to identify malicious domain names as soon as they appear and help mitigate many Internet threats. They analyzed the DNS based on domain names using the SVM classifier. Through their experiment, the results showed that the characteristics extracted by bi-gram performed much better than a single alphanumeric character; however, the degree of precision is still limited. To solve this problem, they set a threshold value to eliminate common characteristics that have low frequency. And then they discovered that it worked very well with a precision higher than 88.14%. And they can also get a better result, if they combine many domain names in one instance. However, if they are compared with other algorithms such as Nave Bayesian or C5.0, they found that SVM gave an effective result and is suitable for classifying the examples of two classes with a high dimensional space.

Wang et al. presented LIAR [21], a new set of data that can be used for the automatic detection of false news. Thought LIAR is considerably larger in size, unlike other data sets, this set of data does not contain complete articles, contains 12800 hand-labeled short statements. Compared with the previous data sets, LIAR is an order of a greater magnitude, which allows the development of statistical and computational approaches for the detection of false news. The short and authentic statements of LIAR in various contexts with various speakers also make possible the investigation into the development of a fake news detector of wide coverage.

They show that by combining metadata with text, significant improvements can be made to detect fake fine-grain news. Given the detailed analysis report and links to source documents in this data set, it is also possible to explore the task of automatic data verification based on knowledge in the future. Its corpus can also be used for the classification of postures, the mining of arguments, the modeling of topics, the detection of rumors and the political research of NLP.

In [28], Giuseppe Ateniese and Luigi V. Mancini introduced a novel approach to extract meaningful data from ML classifiers using a meta-classifier. While previous work investigated the privacy concerns of a single database record, its focus is on statistical information strictly related to the training samples used during the learning phase. They showed that several ML classifiers suffer a new kind of information leakage that is not captured by models that preserve privacy, such as PPDM or differential privacy.

2.3 Fake content detection analysis

There are some open issues in fake content detection. Fake content detection in web is newly emerging area so the research directions should be the data mining perspectives. Fake content in the web may have 4 types of open issues- [29]

1. Fake content detection based on data
2. Fake content detection based on feature
3. Fake content detection based on model
4. Fake content detection based on application

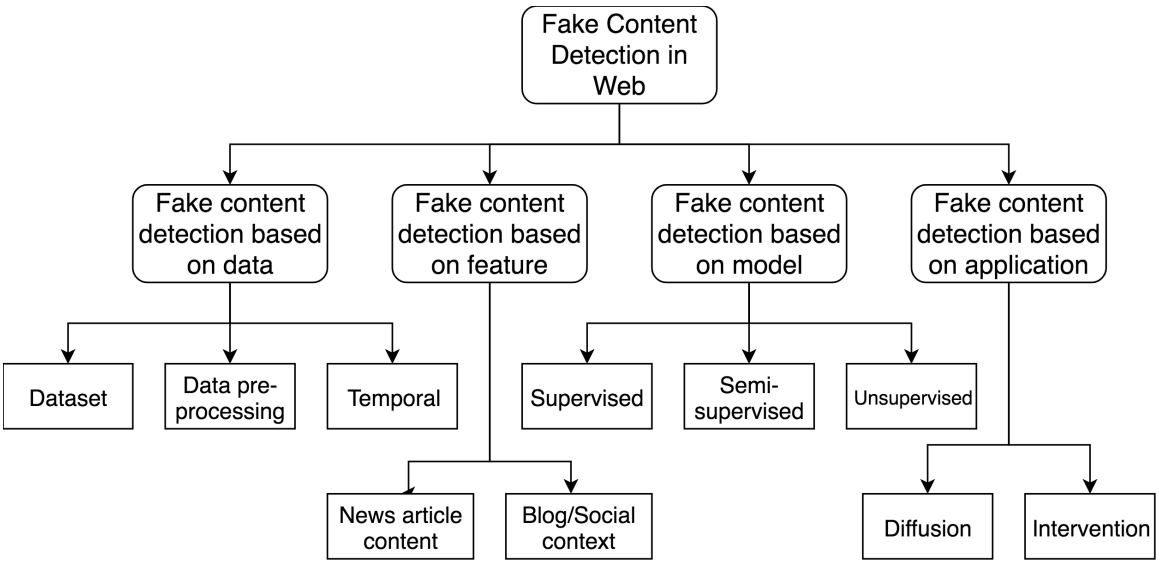

Figure 2.1: Open issues for fake content detection in web

2.3.1 Fake content detection based on data

Fake content detection based on data research focuses on different types of data characteristics, such as: data set, data preprocessing and temporal. From the perspective of the data set, we demonstrate that there is no baseline data set that includes resources to extract all the relevant characteristics. One promising direction is to create a large-scale false news reference data set, which researchers can use to facilitate further research in this area. From a temporal perspective, the dissemination of false news on social networks demonstrates unique temporal patterns different from real news. In this line, an interesting problem is to make an early detection of false news, whose objective is to provide early alerts of false news during the diffusion process. From a data preprocessing perspective, the data can be processed by several methods such as TF, TFIDF.

2.3.2 Fake content detection based on features

Fake content research based on features aims to determine effective features to detect false news from multiple data sources. We have shown that there are two main

sources of data: news content and social context / blog. From a news content perspective, we introduced linguistic and visual-based techniques to extract text information features. Note that features based on linguistics have been studied extensively for general NLP tasks, such as the classification and grouping of texts, and specific applications such as the identification of authors and the detection of deception, but the underlying characteristics of false news. They have not fully understood. Therefore, it becomes much more challenging and important to differentiate real and false visual content, and more advanced visual characteristics are needed for this research. From a blog / social context perspective, we introduced functions based on the user, the publication and the network. Existing user-based functions focus primarily on the general user profiles, rather than differentiating account types separately and extracting user-specific functions. Functions based on publications can be represented using other techniques, such as convolution neural networks (CNN), to better capture people's opinions and reactions to false news.

2.3.3 Fake content detection based on model

Fake content detection based on model opens the door to the construction of more effective and practical models for the detection of false news. The approaches mentioned above focus on extracting several characteristics, incorporating these features in supervised classification models, such as Naive Bayes, decision tree, logistic regression, k nearest neighbor (KNN) and support vector machines (SVM), and then selecting the classifier that does the best.

In addition, most existing approaches are supervised, which requires a set of truthful background data from previously recorded false news to train a model. However, obtaining a set of reliable false news data requires a lot of time and work, since the process often requires experts to conduct a careful analysis of the claims and additional evidence, context and reports from authoritative sources. Therefore, it is

also important to consider scenarios in which limited or unlabeled false news is not available in which semi-supervised or unsupervised models can be applied. While models created by supervised classification methods may be more accurate, given a set of truly well-fielded data for training, unsupervised models may be more practical because unlabeled datasets are easier to obtain.

2.3.4 Fake content detection based on application

Application-based fake content research encompasses research that is directed to other areas beyond the detection of fake news. We propose two main directions in this sense: dissemination of false news and intervention of false news. The spread of false news characterizes the dissemination routes and false news patterns on social networking sites. Some initial research has shown that true information and misinformation follow different patterns when propagated on online social networks. Likewise, the dissemination of false news on social networks demonstrates its own characteristics that require further investigation, such as social dimensions, the life cycle, the identification of the spreader, etc.

2.4 Chapter Summary

Many researchers proposed different models in their research [21] [30] [31] but majority of the researchers use only one or two machine classifier methods in their simulation. But there is a hidden problem that researchers use the classifier method may not the proper method. So we use six machine classifier methods and find the best method.

CHAPTER III

Proposed Approach and Model

3.1 Introduction

Data preprocessing, feature selection and classifier are crucial for classification in machine learning. Analyzing the key properties of a dataset with the feature extraction and model best suited for that data to lead to better results. The accuracy is better if we use the feature extraction method properly. Classical machine learning classifier requires numerical values to represent observations of each class. From the data sets in natural language processing tasks (NLP) are usually plain text, as is the case with this research study, there must be a conscientious choice about how to accurately represent the text numerically. Here we use some natural language processing (NLP) techniques to produce vectorized representations of text documents. We use the NLP techniques in the raw data for preparing the data in the numerical representation. In this thesis, we use the techniques like stop word removal, tokenization, stemming, lemmatization, word embedding (word2vec), sentiment analysis, term frequency-inverse document frequency (TF-IDF). This chapter indicates the preprocessing techniques and feature extraction techniques to refine the textual data, as well as a brief synopsis on the field of sentiment analysis to motivate the idea that sentiment scores can be important features for this study since they provide insight about the motivation and purpose of a piece of text. Finally, this chapter formally

introduces the classifier models used in this research: support vector machines, Naive Bayes, Logistic Regression, k-nearest neighbors, Decision Tree, and neural networks.

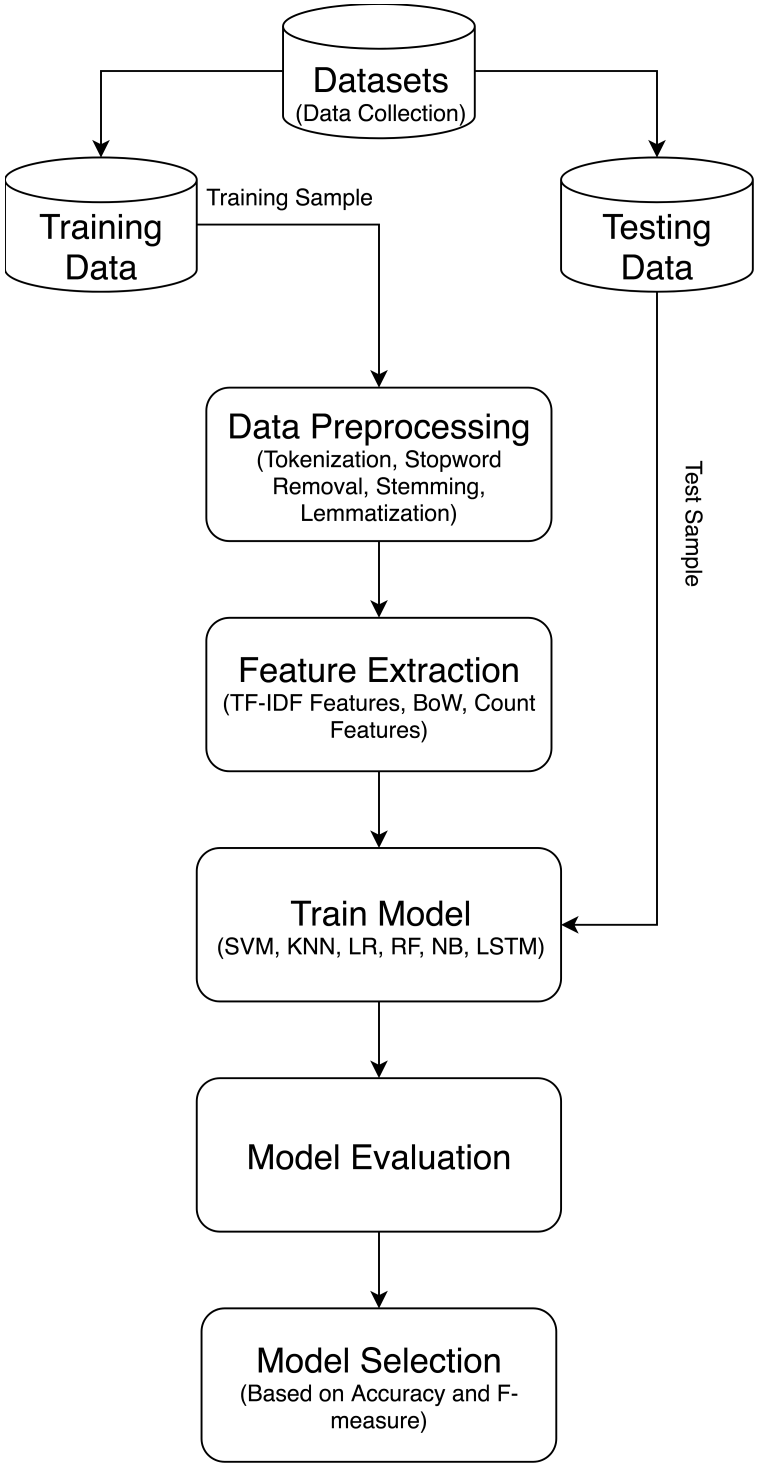

Figure 3.1: Work Flow Diagram

3.2 Selecting Different Content

The number of selected articles for this study pretty high. We use different types of content here to create a dataset. The idea was to gather several articles with news source both inside the news sources and in real life. We wanted to avoid retrieving very different articles, with respect to their content, which would ease the task of determining the authorship. In this way, the conditions found in cyberbullying situations, with respect to the participants and conditions, are sufficiently replicated. In order to select the articles, we checked that it was regular news, the number of articles of the related news, irregular news like the news published at once in a year or more time, and the relation between other selected articles (we wanted the selected articles to be connected). The final dataset was homogeneous, regarding writing style, but was also diverse, regarding the real articles behind the news sources.

3.3 Data Preprocessing

Real-world data is usually messy. One of the first steps before performing any data analysis is to clean or refine the data by making the data structured and correct, and removing any discernible noise. The preprocessing steps performed in this study fall into the category because the text is being converted into a convenient and standard form. The very first step in text processing is to tokenize the data, or separate the words. Though common practice is to simply use whitespace and punctuation to delimit words, compound words such as proper nouns (e.g., White House) can lose their meaning when broken up. To overcome this information loss, named entity recognizers can be used to prevent splitting up these tokens.

To standardize the data, the tokens may then be converted to their roots so dierent tenses of words can be linked together. However, this task, called lemmatization, requires that the words in each sentence are first tagged with their part-of-speech to

determine the root word. Since part-of-speech tagging is sometimes too computationally intensive for large documents, a simpler approach called stemming is often used in its place. Stemming aims to remove the suxes of each word to get the root. However, the eectiveness of the stemmer is implementation dependent. For example, the Porter stemmer aims to remove suxes using pattern matching, potentially producing incoherent words or semantically incorrect words (e.g., the stem of ties is ti, and the stem of operator is operate). Finally, to refine the text of the document, articles, pronouns, prepositions, and other uninformative words are sometimes filtered out before the core analysis is performed. So these uninformative words are called stop words, generally help in info recovery and document resemblance tasks.

3.3.1 Stop Word Removal

Stop word removal is one of the frequently used preprocessing steps through different NLP applications. The idea is simply removing the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally categorized as stop words. These words have no meaning in some of the NLP tasks like information recovery and classification, which means these words are not very discriminative. In contrast, in some NLP applications stop word removal will have very little impact. The stop word list for the given language is a well hand-curated list of words that occur most commonly across corpuses. While the stop word lists for most languages are offered online, these are also ways to repeatedly generate the stop word list for the given corpus. Simply, to build a stop word list is based on word's document frequency (Number of documents the word presents), where the words present across the corpus can be treated as stop words. Many research has been done to get the ideal list of stop words for some specific corpus.

Stop words are unimportant words in a language that will produce noise when used as features in text classification. These are words that are usually used a lot in

sentences to help connect the thought or to help in the structure of the sentence. Articles, prepositions and conjunctions and some pronouns are measured empty words. We eliminate common words like, to, about, an, are, at, be, by, for, from, how, in, is, of, on, or, that, the, this, this, was, what, when, where, who, will, etc. Those words were removed from each document and the managed documents were stored and moved on to the next step.

3.3.2 Tokenization

Tokenization is the act of breaking up a sequence of strings into portions such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be single words, phrases or even whole sentences. While tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining. Tokenization relies on simple heuristics in order to distinct tokens by following a few steps:

- Tokens or words are separated by whitespace, punctuation marks or line breaks
- White space or punctuation marks may or may not be included depending on the need
- All characters within contiguous strings are part of the token. Tokens can be made up of all alpha characters, alphanumeric characters or numeric characters only.

Tokens themselves can also be separators. For example, in most programming languages, identifiers can be placed together with arithmetic operators without white spaces. Although it seems that this would appear as a single word or token, the grammar of the language actually considers the mathematical operator (a token) as a separator, so even when multiple tokens are bunched up together, they can still be separated via the mathematical operator. We use tokenization to break a stream

of text up into words, phrases, symbols, or other meaningful elements. The list of tokens becomes input for further processing such as stemming, lemmatization etc.

3.3.3 Stemming

Stemming, in literal terms, is the process of cutting down the branches of a tree to its stem. So successfully, with the use of some basic rules, any token can be cut down to its stem. Stemming is more of a crude rule-based method by which we want to club together different variations of the token. For example, the word eat will have variations like eating, eaten, eats, and so on. In few applications, as it does not make sense to distinguish between eat and eaten, we typically use stemming to club both grammatical variances to the root of the word. While stemming is used most of the time for its simplicity, there are cases of complex language or complex NLP tasks where it's necessary to use lemmatization instead. By using stemming to make classification quicker and efficient. Furthermore, we use Porter stemmer, which is the most commonly used stemming algorithms due to its accuracy.

3.3.4 Lemmatization

Lemmatization is the process of converting a word to its base form. The dissimilarity between stemming and lemmatization is, lemmatization reflects the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors. Lemmatization is a more robust and methodical way of combining grammatical variations to the root of a word. Such as, lemmatization would correctly identify the base form of caring to care, whereas, stemming would cutoff the ing part and convert it to car. We use stemming to make classification quicker and efficient.

3.4 Feature Extraction

3.4.1 Bag of Words

A bag-of-words model, or BoW for short, is a method of mining features from text for use in modeling, such as with machine learning algorithms. The method is very simple and flexible, and used in a myriad of ways for extracting features from documents. A bag-of-words is a demonstration of text that describes the occurrence of words within a file. It includes two things:

1. A vocabulary of known words

2. A measure of the existence of known words

It is named a bag of words, because any data about the order or structure of words in the document is rejected. The model is only concerned with when known words occur in the document, not where in the document.

3.4.2 N-grams

An n-gram is simply an order of tokens. In the background of computational linguistics, these tokens are typically words, though they can be characters or subsets of characters. The n simply refers to the number of tokens. N-grams of texts are widely used in text mining and natural language processing tasks. They are fundamentally a set of co-occurring words within a given window and when calculating the n-grams we typically move one word forward (although we can move X words forward in more advanced scenarios). Such as, for the sentence "I live in dhaka". If N=2 (known as bigrams), then the ngrams would be:

- I live

- live in

- in Dhaka

So we have 3 n-grams in this case. Notice that we moved from the->live to live->in to in, etc, essentially moving one word forward to generate the next bigram. If N=3, the n-grams would be:

- I live in
- live in dhaka

So we have 2 n-grams in this case. When N=1, this is stated to as unigrams and this is basically the separate words in a sentence. When N=2, this is named bigrams and when N=3 this is named trigrams. When N>3 this is usually mentioned to as four grams or five grams and so on. In this thesis, we use unigrams, bigrams and trigrams model. Table shows the word level unigrams, bigrams and trigrams for the sentence I live in dhaka.

Table 3.1: Word level Unigrams, Bigrams and Trigrams

Unigrams	
Token Value	**Token Sequence**
I	1
live	2
in	3
dhaka	4
Bigrams	
I live	1
live in	2
in dhaka	3
Trigrams	
I live in	1
live in dhaka	2

3.4.3 TF-IDF

Tf-idf means frequency-inverse document frequency, and the tf-idf weight is a weight frequently used in information recovery and text mining. This weight is a numerical measure used to estimate how significant a word is to a document in a collection or corpus. The significance increases consistently to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Dissimilarities of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. One of the easiest ranking functions is calculated by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model. Tf-idf can be effectively used for stop-words filtering in many subject fields including text summarization and classification. Naturally, the tf-idf weight is collected by

two terms: the first computes the normalized Term Frequency (TF). The number of times a word seems in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), calculated as the logarithm of the number of the documents in the corpus divided by the number of documents where the exact term appears.

TF: Term Frequency, it measures how repeatedly a term occurs in an article. Every document is different in length so, it is possible that a term would look much more times in long documents than smaller ones. So, the word frequency is often divided by the article length (the total number of terms in the article) as a way of normalization:

$$TF(t) = \frac{\textit{Number of times term t appears in a article}}{\textit{Total number of terms in the article}} \tag{3.1}$$

IDF: Inverse Document Frequency, it measures importance of a term. While calculating TF, all terms are measured equally important. Still it is known that certain terms, for example "is", "of", and "that", may appear a lot of times but have less importance. Therefore we need to weigh down the frequent terms while scale up the rare ones, by calculating the following:

$$IDF(t) = \log_e \frac{\textit{Total number of terms in the article}}{\textit{Number of times term t appears in a article}} \tag{3.2}$$

As soon as preprocessing is done, the left over text must be transformed into real valued vectors so that the text can be used by a model. One method to produce numerical values for words in an article is to represent each word by its term frequency-inverse document frequency score. The term frequency-inverse document frequency (tf-idf) of a word is used to quantify the importance of a word in a corpus based on how frequently the word shows up in a article and how many other articles also contain the word.

$$TF - IDF(t) = \frac{Number\ of\ times\ term\ t}{Total\ number\ of\ terms} * \log_e \frac{Total\ number\ of\ terms}{Number\ of\ times\ term\ t} \quad (3.3)$$

A basic tf-idf scoring function is available in the above Equation. The first term represents the term frequency (tf) of the word t, which is the ratio of the number of times term t appears in a document and total number of terms in the document. The 2nd or another term is the inverse document frequency (idf) which assists to boost rarer, more useful words, and diminish the impact of commonly used non-informative words, like articles and pronouns. Since the idf is computed by taking the logarithm of the total number of documents divided by the number of documents with term t in it with the word oset by 1 to avoid 0 denominators, words that appear in almost all the documents will have a idf close to 0. In contrast, words that appear in only select documents will have higher idf values, thereby growing their tf-idf weights. For example, consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousands of these. Then, the inverse document frequency (i.e., idf) is calculated as log (10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.

3.4.4 Word Embedding

One more way to numerically represent a text is to use a word embedding model to transform each word into a real-valued vector. The word embedding of a word is the high-dimensional vector that is the outcome of mapping the word with a parameterized function that was developed using a large corpus that is representative of the source language. One of the primary benefits of using word embedding is that similar words will tend to map to the same region. Word embedding is one of the

most popular demonstration of document vocabulary. It is capable of taking context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

Embedding Layer: An embedding layer, for lack of an improved name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification. It needs that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is stated as part of the model, such as 50, 100, or 300 dimensions. The initialization of vector starts with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way with the Back propagation algorithm. The one-hot encoded words are plotted to the word vectors. If a multilayer Perception model is used, at that time the word vectors are concatenated before being fed as input to the model. If a recurrent neural network is used, now each word may be taken as one input in a sequence.

Consider the following related sentences: Have a good day and Have a great day. They hardly have different meaning. If we make an exhaustive vocabulary (lets call it V), it would have V = Have, a, good, great, day. So now, let us create a one-hot encoded vector for each of these words in V. Length of this one-hot encoded vector would be equal to the size of V (=5). We would have a vector of zeros except for the element in the index that represents the corresponding word in the vocabulary. That specific element would be one. The explanation of the encodings is given below. Have = [1,0,0,0,0] ‘; a = [0,1,0,0,0] ‘; good = [0,0,1,0,0] ‘; large = [0,0,0,1,0] ‘; day = [0,0,0,0,1] ‘(‘ represents transposition) If we try to visualize these encodings, we can think of a space of 5 dimensions, where each word occupies one of the dimensions and has nothing to do with the rest (there is no projection in the other dimensions). This means that ”good” and ”great” are as different as ”day” and ”have”, which is false.

Our goal is for words with a similar context to occupy close spatial positions. Mathematically, the cosine of the angle between these vectors must be close to 1, that is, angle close to 0.

Here comes the idea of creating distributed representations. Intuitively, we introduce a certain dependence of a word on other words. The words in the context of this word would obtain a greater proportion of this dependence. In an active encoding representation, all words are independent of each other as mentioned above.

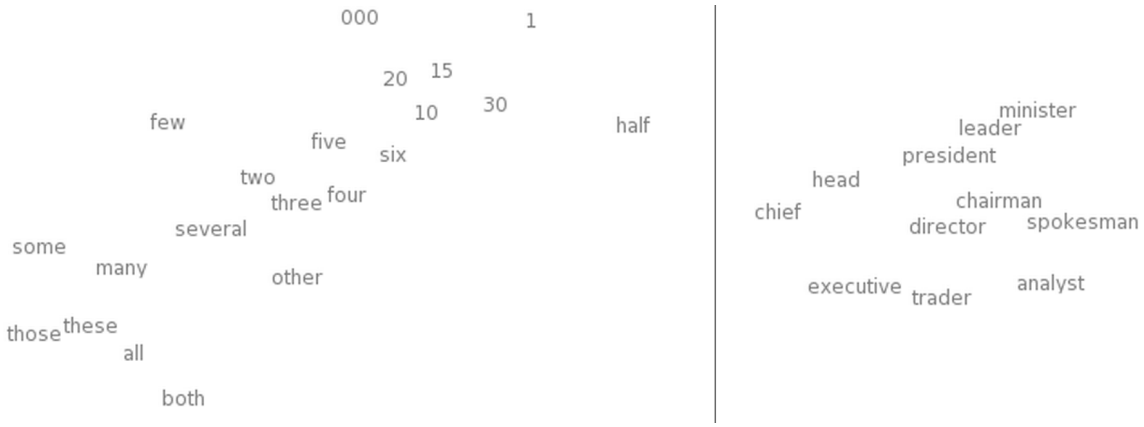

Figure 3.2: t-SNE visualization of word embeddings [1] Left: the number region in a word embedding model Right: the occupations region

t-distributed stochastic neighbor embedding (t-SNE): It is a probabilistic technique designed for the reduction of dimensionality and can be used to visualize high-dimensional data sets such as word embedding. The t-SNE algorithm converts the similarities between the data points into joint probabilities and tries to minimize the divergence between these joint probabilities and the high-dimensional data. Due to the reduction in dimensionality, mainly for visualization purposes, the axes and their units in the t-SNE graphics have no real meaning; however, the t-SNE still shows significantly which of the points are closest together in the space of the original feature. An example of the benefit is shown in Figure 3.2, which shows enlarged views in the number and work regions of a visual t-SNE constructed by word embed-

ding. The word integration model used in this study is the GloVe model that was trained by a team of Stanford researchers who use global statistics of word-to-word occurrence. This model of vector representation proves to be highly effective when grouping similar or related words in higher dimensions. In addition, the model was designed in such a way that vector differences between the two-words word embedding capture "the meaning specified by the juxtaposition of two words" [42]. For example, the conceptual difference between "short", "shorter" and "shorter" is similar to the difference between "slow", "slower" and "slower". By using the GloVe word-incorporation vectors of the underlying words, this conceptual difference is reflected in the vector differences in the t-SNE graph of Figure 3.3.

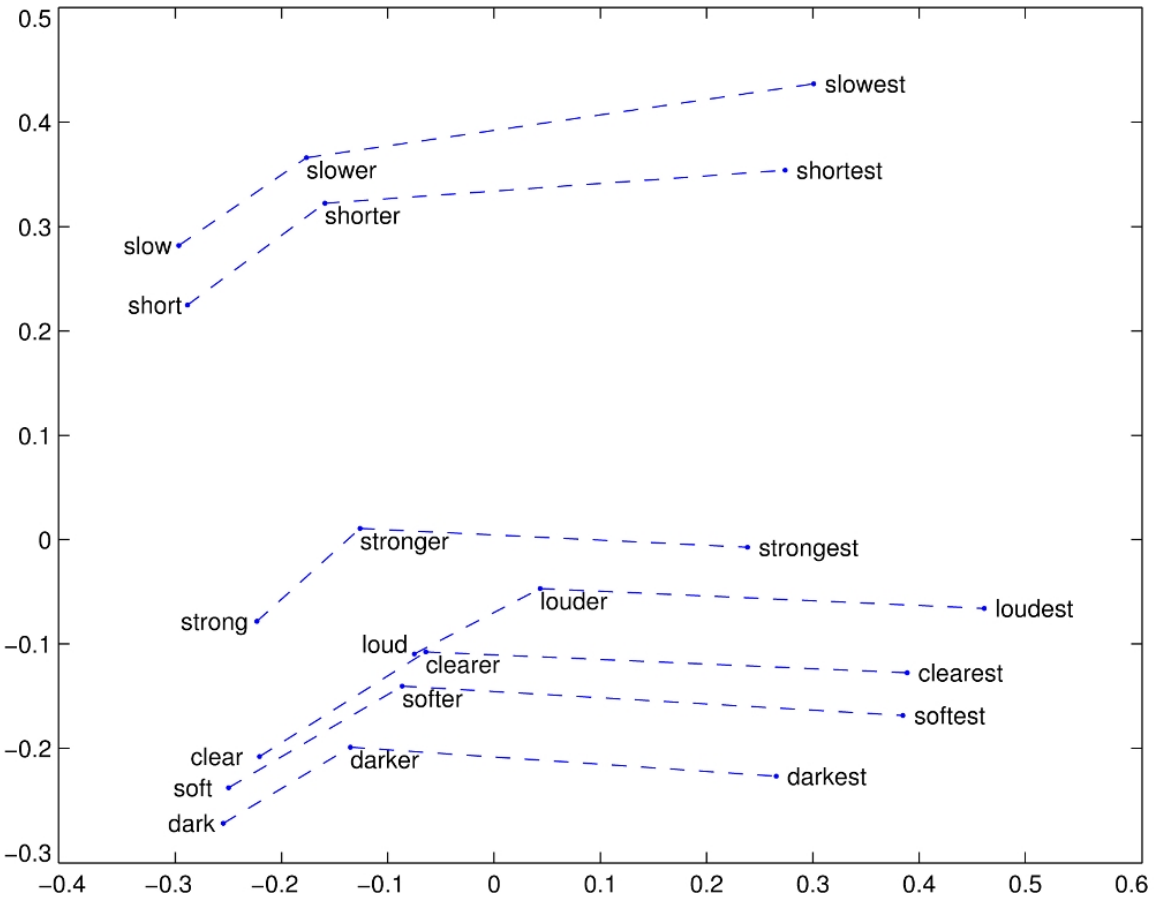

Figure 3.3: t-SNE visual of GloVe word embeddings for select comparative and superlative words [2]

3.5 Classification Models

3.5.1 Support Vector Machine (SVM)

Introduction: Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges. However, it is mainly used in classification problems. SVM is a discriminative classifier formally defined by a separating hyperplane. In other words, given the labeled training data (supervised learning), the algorithm generates an optimal hyperplane that categorizes new examples. In the two-dimensional space, this hyperplane is a line that divides a plane into two parts, where in each class it lies on each side. In this algorithm, we plot each data element as a point in n-dimensional space (where n is the number of entities we have) with the value of each entity as the value of a particular coordinate. Then, we perform the classification finding the hypo plain that differentiates the two classes very well.

How does it work: Suppose we give us a graph of two kinds of labels in the graph as shown in the Figure 3.4. Can we decide on a separation line for the classes? We could have found something similar to the following image Figure 3.5. Separate the two classes enough. Any point that is to the left of the line falls in the orange circle class and, to the right, falls in the blue square class. Class separation, that's what SVM does. Find a line / hyperplane (in the multidimensional space that separates the outs classes). In short, we will discuss why we write the multidimensional space.

Handle complex datasets: The support vector machine (SVM) classifier is a high-performance machine learning algorithm that is based on the relatively simple concept of dividing data into different regions. For example, in the binary classification, the SVM seek to maximize the distance between the data points of the opposite classes and the dividing decision limit. The decision limit, also known as the hyperplane, is formulated as a linear combination of weights in each dimension of the

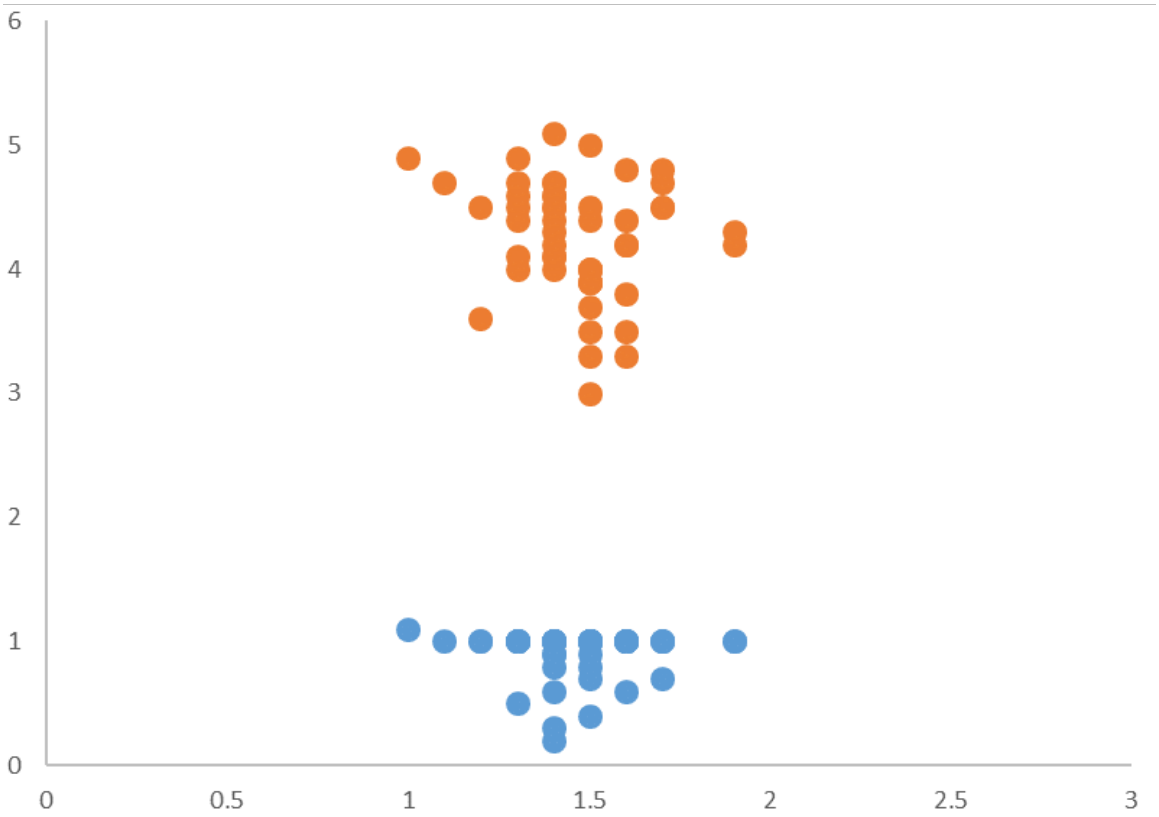

Figure 3.4: We can separate the classes by drawing a line between the orange circles and blue circles

input, as shown in the equation 3.4. w is the vector of weights applied to the input x, a support vector, whose length corresponds to the dimensionality of x, and b is the bias, a constant offset.

$$w^T x + b = 0 \tag{3.4}$$

The data points closest to the hyperplane are called support vectors, and the shortest distance between these points and the limit is called margin. By maximizing the margin, the probability of classification errors due to noise is reduced, assuming that the test data points come from the same distribution as the training data. One clear advantage of using SVM is the low memory cost: only these support vectors must be kept in memory; the other data points, which are furthest from the hyperplane,

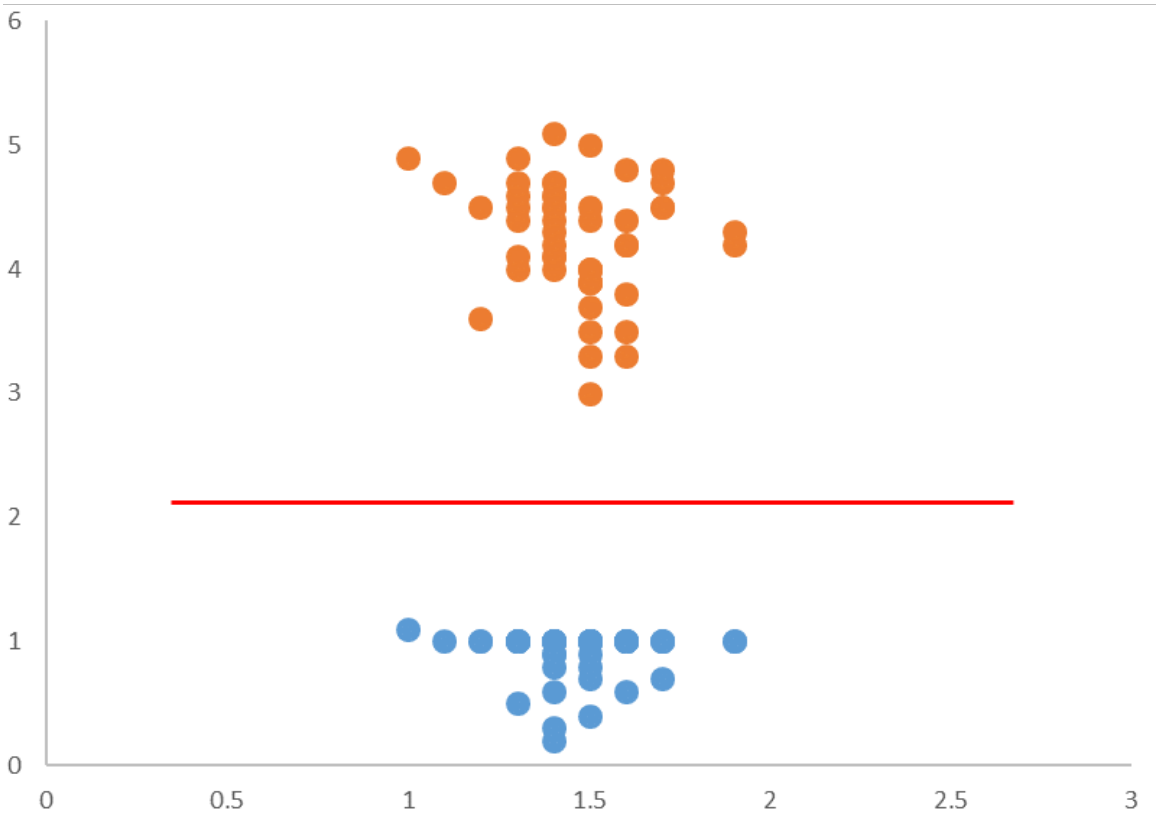

Figure 3.5: Sample cut to divide into two classes

should no longer be considered (remember that the KNN algorithm requires that each input be compared with each training observation to determine the most likely class) [32]. However, in most practical applications, the data cannot be separated directly into 2 well-defined regions, especially for complex data sets with much overlap in the space of the original feature. In these cases, it may be possible to allocate the space of the characteristic in a higher dimension where the classes are more easily separable. However, this assignment may be too complex and computationally intensive to apply to large training sets. This computational complexity can be completely avoided by using symmetric kernel functions in the pair of vectors, resulting in a similarity score that is exactly equivalent to the product of points of their inputs when assigned to a higher dimensional space without having to map explicitly the vectors calculating their point product, as shown in equation 3.5 [33].

$$K(x, x_0) = G(x)TG(x_0) = G(x_0)TG(x) = K(x_0, x) \tag{3.5}$$

One example use-case for this higher-order mapping is when the classes data points are radially dependent. As shown in Figure 3.6, the data can be mapped to R3 using the mapping function in equation 3.7.

$$z^2 = x^2 + y^2 \tag{3.6}$$

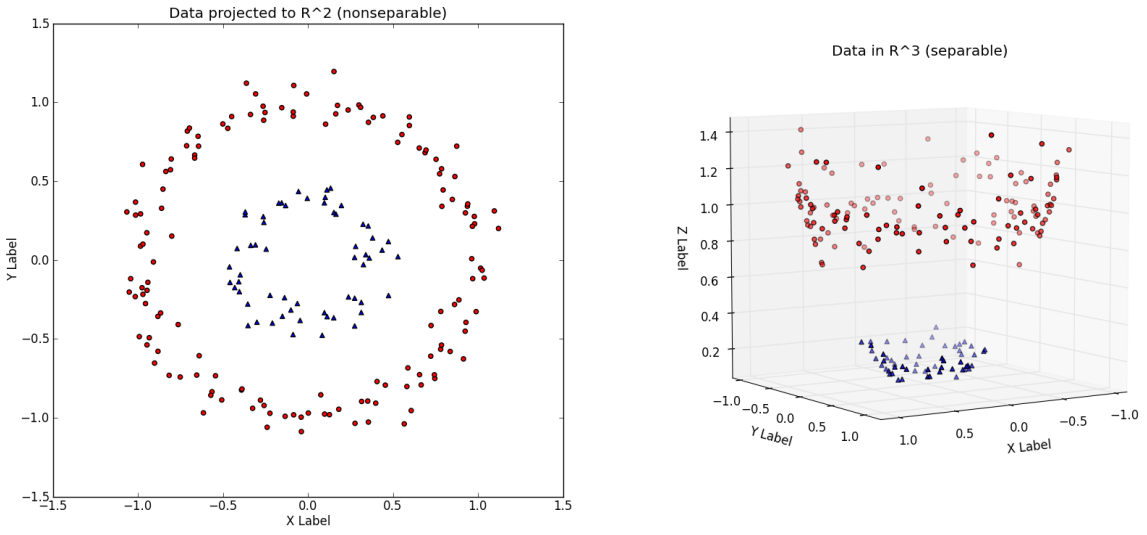

Figure 3.6: Visualizing 2-dimensional observations from two radially dependent distributions [3] Left: 2-D plot of the dataset where x is the X-Label and y is the Y-Label Right: Dataset is mapped into R3 where z is the radius from the origin

3.5.2 Naive Bayes

Introduction: The Naive Bayesian classifier is based on Bayes' theorem with assumptions of independence among the predictors. A naive Bayes model is easy to build, without an estimation of complicated iterative parameters, which makes it particularly useful for very large data sets. Despite its simplicity, the naive Bayes

classifier often works surprisingly well and is widely used because it often exceeds the most sophisticated classification methods. A naive Bayes classifier uses probability theory to classify the data. Bayes 'naive classification algorithms make use of Bayes' theorem. The key idea of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced. What makes a naive Bayes classifier naive is its assumption that all the attributes of a data point under consideration are independent of each other. A classifier that classifies fruits in apples and oranges would know that the apples are red, round and have a certain size, but they would not assume all these things at once. Oranges are also round, after all.

A Naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms that make use of statistical independence. These algorithms are relatively easy to write and execute more efficiently than the more complex Bayes algorithms. The most popular application is spam filters. A spam filter examines the email messages for certain keywords and places them in a spam folder if they match. Despite the name, the more data we get, the more accurate a naive Bayes classifier becomes, like when a user marks emails in an inbox for spam. Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian environment. It can also be denoted using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection.

Algorithm: The Bayes theorem provides a way to calculate the posterior probability, $P(A|B)$, from P(A), P(B) and $P(B|A)$. The Naive Bayes classifier assumes that the effect of the value of a predictor (B) on a given class (A) is independent of the values of other predictors. This assumption is called conditional class independence.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.7}$$

- $P(A|B)$ is the posterior probability of class (objective) given predictor (attribute).
- P(A) is the previous class probability.
- $P(B|A)$ is the probability that is the predictive probability of a given class.
- P(B) is the previous probability of a predictor.

In the ZeroR model there is no predictor, in the OneR model we try to find the best predictor, naive Bayesian includes all the predictors that use the Bayes rule and the assumptions of independence among the predictors.

Classification: Now that we have a way of estimating the probability that a given data point falls into a certain class, we need to be able to use this to produce classifications. Naive Bayes handles this in a very simple way; simply choose the c_i that has the highest probability given the characteristics of the data point. This is known as the Maximum A Posteriori decision rule. This is because, referring to our formulation of the Bayes rule, we only use the terms $P(B|A)$ and P(A), which are the probability and the previous terms, respectively. If we only used $P(B|A)$, the probability, we would be using a maximum Probability decision rule.

Multinomial: We use here the basic multinomial model using Naive bayes. It is used for discrete counts. Such as, we have a text classification problem. Here we can consider Bernoulli's essays, which is one step further and instead of "word that appears in the document", we have "count how often the word appears in the document", and it can be considered as number of times that the result number x_i on the n tests is observed.

3.5.3 K-Nearest Neighbors (KNN)

Introduction: The nearest k-neighbor algorithm (KNN) is a classification algorithm based on decision boundaries that classifies an entry to the largest class of

its k nearer neighbors in space [34]. The hyper adjustable parameters for this algorithm include k, the number of neighbors used to classify any given input and the distance metric used to determine the nearest neighbors. It is believed that the least robust KNN is k = 1 because this model runs the risk of becoming very dependent on noise or atypical values. The KNN variants can weigh the votes of the training set observations with their cosine similarity in relation to their input. The advantages of KNN as a classification algorithm for practical applications are numerous. As a non-parametric learning algorithm, KNN does not require assumptions about the underlying distribution of the input data. For applications in natural language processing, specific classification of news items such as genuine or malicious political authorship, there is not enough evidence to assign the text of the articles to a closed probability probabilistic distribution. In addition, since KNN is an algorithm based on instances, the only "learning" required is to load the observations into memory, which significantly reduces training time [69]. As with any machine learning algorithm, KNN also has some drawbacks. Input testing with KNN requires that the entire training set be in memory, or be available through a high-performance data store, such as a database, so that the vectorized inputs can be compared to each point of training to determine the nearest k neighbors. However, because our data set is not too large, this limitation is not a burden for our task. Another disadvantage of KNN is the cost of the test time: for each individual entry, each training point should be checked to see if it is one of the nearest k neighbors, and if so, use its label for classification. As a result, even this limitation does not seriously affect the performance of the tests due to the size of the data set; therefore, a trained KNN model has high performance and can be used for real-time classification [35].

Algorithm: Lets take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS):

Here, we intend to find out the class of the blue star (BS). BS can be RC or GS

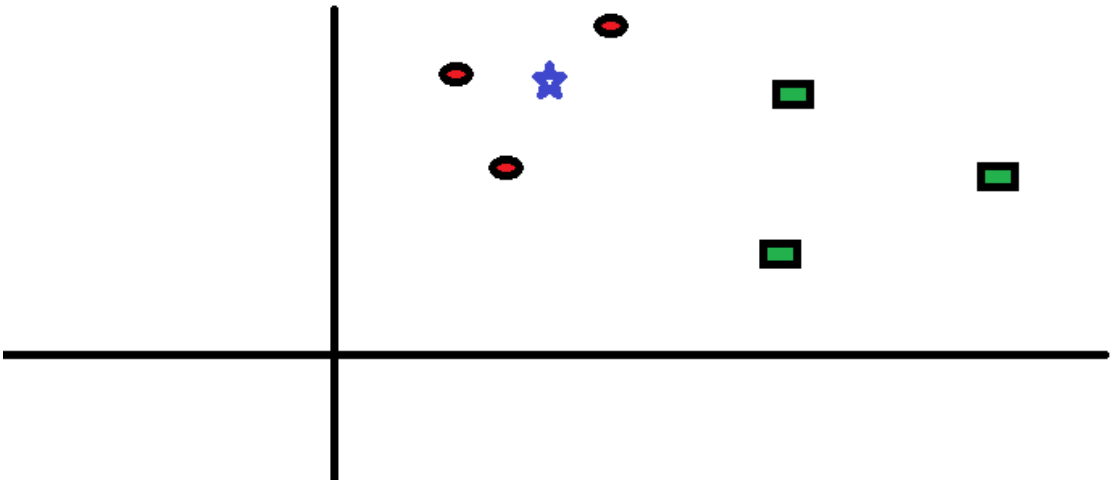

Figure 3.7: Spread of red circles (RC) and green squares (GS)

and nothing else. The "K" is the algorithm KNN is the closest neighbor we want to vote on. Let's say that K = 3. Therefore, we will now make a circle with BS as a center so large that it only includes three data points in the plane. See the following diagram for more details:

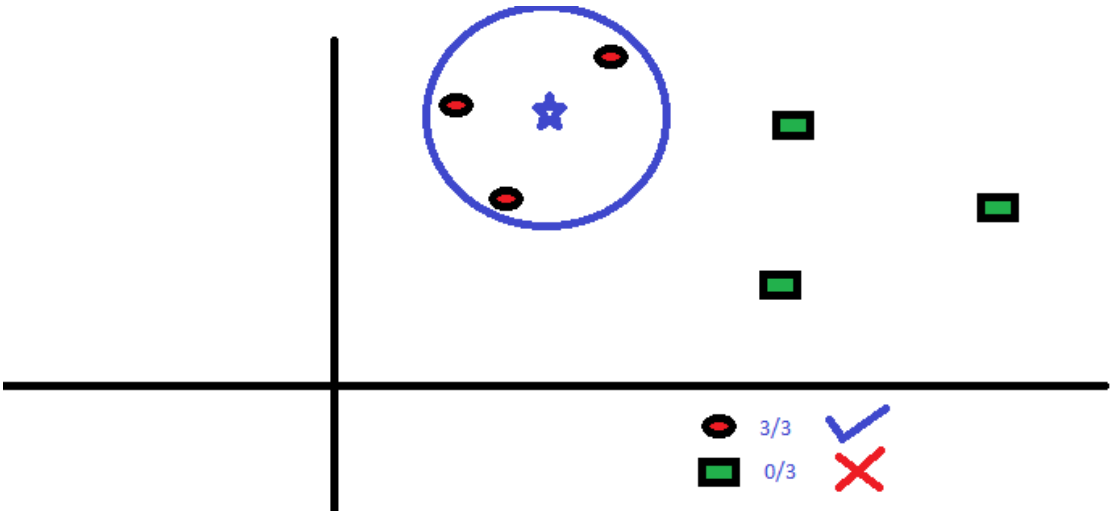

Figure 3.8: The three points closest to BS are all RC

The three points closest to BS are all RC. Therefore, with a good level of confidence we can say that the BS must belong to the RC class. Here, the selection became very

obvious when the three votes of the nearest neighbor went to RC. The choice of the parameter K is very critical in this algorithm. Below we will understand what are the factors to consider in order to conclude the best K.

Choosing the factor K: In K means algorithm, for each test data point, we would be seeing the K training data points closer and we would take the most frequent classes and assign that class to the test data. Therefore, K represents the number of training data points that are near the test data point that we will use to find the class.

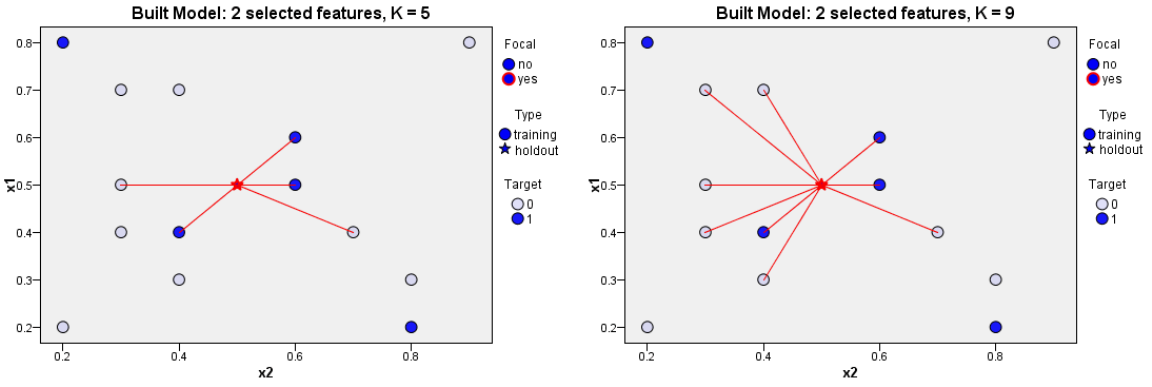

Figure 3.9: Left: The factor K = 5; Right: The factor K = 9

3.5.4 Logistic Regression

Logistic regression, falls under supervised machine learning. Solve classification problems (to make predictions or make decisions based on past data). It is used to predict binary results for a given set of independent variables. The result of the dependent variable is discrete. The result or output of Logistic Regression is a sigmoid curve (or more popularly known as S curve). Where the value on the x-axis (independent variable) would define the dependent variable on the y-axis. In logistic regression there are only 2 possible outcomes. 0 and 1. That happens, or it does not happen. We use a threshold value to facilitate our prediction. If the value of

y corresponding to the x-axis (probability) is smaller than the threshold value, the result is taken as 0. If it is greater than the value, the result is taken as 1. It is an extension of the linear regression where the dependent variable is categorical and does not continue. Predict the probability of the outcome variable. Logistic regression can be binomial or multinomial. In binomial or binary logistic regression, the result can have only two possible types of values (for example, "Yes" or "No", "Success" or "Error"). Multinomial logistics refers to cases in which the result can have three or more possible types of values (for example, "good" versus "very good" versus "better"). In general, the result is coded as "0" and "1" in the binary logistic regression.

Representation of Logistic regression:

1. **Logistic Response Function:**

$$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+.....+\beta_k x_k)}} \tag{3.8}$$

   - Positive values are predictive of class 1
   - Negative values are predictive of class 0

   The coefficients, or values, are selected to maximize the probability of predicting a high probability for the observations that really belong to class 1 and to predict a low probability for the observations that really belong to class 0. The output of this function always is between 0 and 1

2. **Odds Ratio:**

   Odds: P(y=1)/P(y=0)

   The odds ratio for a variable in logistic regression represents how the odds change with a 1 unit increase in that variable holding all other variables constant.

- Odds >1 if y = 1 is more likely
- Odds <1 if y = 0 is more likely

3. The Logit

$$odds = e^{\beta_0+\beta_1 x_1+\beta_2 x_2+.....+\beta_k x_k} \tag{3.9}$$

$$\log(odds) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k \tag{3.10}$$

This is called the Logit and looks like linear regression The bigger the Logit is, the bigger is P(y = 1).

3.5.5 Random Forest

Random Forest is a supervised learning algorithm. As we can see in his name, he creates a forest and does it in some random way. The "forest" that he builds is a set of decision trees, most of the times trained with the "bagging" method. The general idea of the bagging technique is that a combination of learning models increases the overall result.To put it in simple words: Random forest creates multiple decision trees and combines them to obtain a more accurate and stable prediction.

A great advantage of random forests is that they can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about the forest at random in the classification, since classification is sometimes considered the basic component of machine learning. Below we can see what a forest would look like at random with two trees:

Random Forest has almost the same hyper parameters as a decision tree or bag sorter. Fortunately, we do not have to combine a decision tree with a bagging sorter and we can easily use the sorting class of Random Forest. As I said, with Random Forest, we can also deal with the tasks of Regression using the Random Forest. Random Forest adds additional randomness to the model, whereas the trees are growing. Instead of looking for the most important feature when dividing a node, look for the

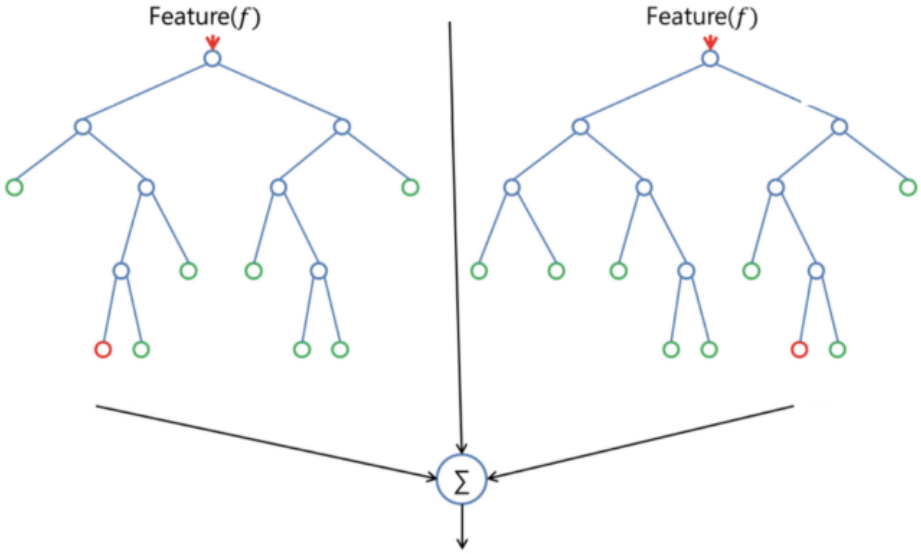

Figure 3.10: Random Forest with two trees

best feature among a random subset of features. This results in a wide diversity that generally results in an improved model.

Therefore, in Random Forest, the algorithm for dividing a node only takes into account a random subset of the characteristics. It can even make trees more random, by using additional random thresholds for each function instead of looking for the best possible thresholds (as a normal decision tree does).

How the system is trained with Random Forest model: Here is how such a system is trained; for a certain number of trees T:

1. Show N random cases with replacement to create a subset of the data. The subset should be about 66% of the total set.

2. In each node:

- For some number, m the predictor variables are selected at random from all the predictor variables.
- The predictive variable that provides the best division, according to some objective function, is used to make a binary division in that node.
- At the next node, choose another variable m randomly from all the predictor variables and do the same.

Depending on the value of m, there are three slightly different systems:

- Selection of random divisor: m = 1
- Breiman packer: m = total number of predictor variables
- Random forest: m <<number of predictor variables. Brieman suggests three possible values for m: 1/2 $\sqrt{m}$, $\sqrt{m}$ and $2\sqrt{m}$ [36]

When a new entry is entered into the system, it runs in all the trees. The result can be an average or a weighted average of all the terminal nodes that are reached, or, in the case of categorical variables, a majority with the right to vote.

- With a large number of predictors, the set of eligible predictors will be quite different from one node to another.
- The higher the correlation between trees, the higher the error rate of random forests, so that a pressure in the model is that the trees are as less correlated as possible.
- As m decreases, both the correlation between trees and the strength of individual trees decrease. So some optimal value of m must be discovered.

3.5.6 Long Short-Term Memory (LSTM)

LSTM neural networks, which represent long-term long-term memory, are a particular type of recurrent neural networks that received much attention recently within the machine learning community. In a simple way, LSTM networks have some internal contextual status cells that act as long-term or short-term memory cells. The output of the LSTM network is modulated by the state of these cells. This is a very important property when we need the prediction of the neural network to depend on the historical context of the inputs, instead of just the last input.

As a simple example, consider that we want to predict the next number of the following sequence: 6 ->7 ->8 ->?. We would like the next exit to be 9 (x + 1). However, if we provide this sequence: 2 ->4 ->8 ->?, We would like to obtain 16 (2x). Although in both cases, the last current entry was number 8, the result of the prediction must be different (when we take into account the contextual information of the previous values and not only the last one).

LSTMs are RNNs with special memory units (also known as cells) that can maintain information selectively over a long period of time [37]. Instead of consisting of a single repeating layer, as in the case of an RNN, the vanilla LSTM has a memory unit composed of four special layers: three sigmoid layers and one tanh layer [38]. All the sigmoidal layers produce values between 0 and 1, and the tanh layer puts values between -1 and 1. Together, these layers help the cell forget, remember, update and produce information.

As shown in the LSTM in Figure 3.11 and Figure 3.12, the first layer is a sigmoid layer whose input is the concatenation of the output from the previous state $(h_t - 1)$ and the current input (x_t). In LSTMs, the sigmoidal layers that bind with dot multiplication operators act as gates that let information pass when their outputs are non-zero values. The first sigmoid layer uses $(h_t - 1)$ and (x_t) and forms a gate in the incoming value stream of the previous cell state (the upper left corner of each cell

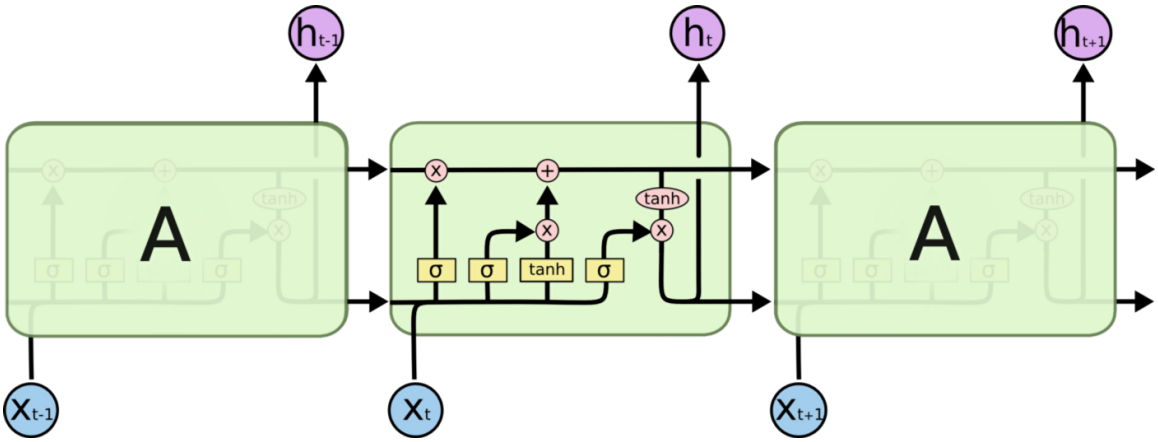

Figure 3.11: Unrolled RNN and LSTM chains: RNN with a single layer in each state [4]

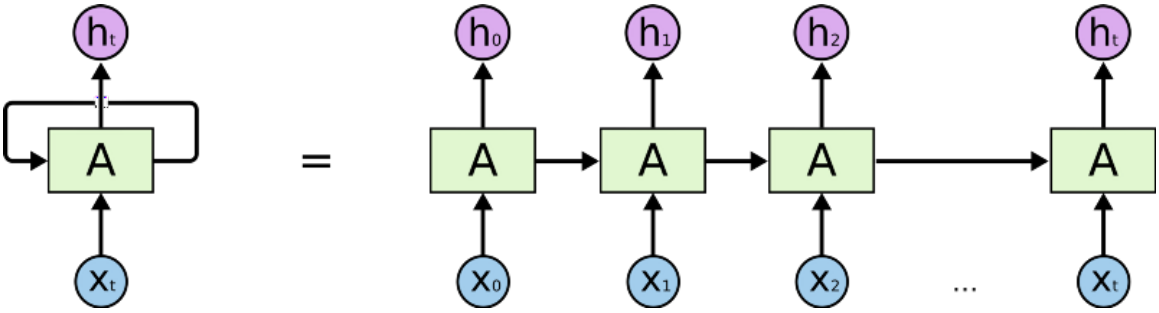

Figure 3.12: Unrolled RNN and LSTM chains: LSTM with one memory unit in each state [4]

in Figure 3.11 and Figure 3.12. This gate, often called the "forgotten gate", generates a number between 0 and 1 for each value in the previous state. Therefore, if the door generates 0 for a particular value, that value is completely forgotten. The next stage of the cell decides what part of the new information (h_{t-1} and x_t) will be retained in the state. First, a $\tan(h)$ layer is used to calculate an updated value for each new information bit. Subsequently, these updated values are filtered through another door to extract what is considered relevant by the LSTM. Finally, the filtered information is added with any information from the previous state that has not been forgotten to form the new cell state.

The last stage of the cell calculates the output for the current state (h_t). The first

step in this stage is to map the state of the cell in the acceptable output space. In Figure 3.11 and Figure 3.12, the activation function of $\tan(h)$ is applied to the state of the cell to push each value between -1 and 1. The transformed state then passes through the final sigmoid gate; therefore, the output consists of only the parts of the cell state that LSTM considers appropriate. For example, in a machine translation configuration, the LSTM can show if the subject is singular or plural, so the LSTM knows how to conjugate the next verb if in fact it is the next entry. There are other LSTM configurations, and each has its own advantages in certain configurations. However, a recent study shows that most of these variations do not work significantly better than vanilla LSTM. The LSTM model used in this thesis is a variant of the vanilla LSTM model that has a configurable number of memory units [39]. Note that the LSTM presented in this section had only one memory unit and, therefore, produced a single output. Adding additional memory units increases the dimensionality of the output in each state. Therefore, if the LSTM contains 50 memory units in each layer, the LSTM will eventually produce a 50-dimensional output vector after processing the entire input sequence. Since the challenge presented in this thesis is a binary classification task, the outputs of each LSTM are passed through another layer of sigmoid activation to produce a prediction.

CHAPTER IV

Experiments and Evaluation Results

4.1 Datasets

4.1.1 Data Collection

In our experiment, the raw data we use comes from different types of real and false sources. We collect different data by scrapping the web. We divide them into training datasets and test datasets. We collect three types of data such as real data, false data and test data.

Real data: The field of false news detection is a relatively new area of research. Therefore, few public data sets are available. In this work, we mainly use a new set of data collected by our team through the compilation of a public news article. We also tested our model on the Horne and Adali data set [40], which is accessible to the public. We use some real data that is collected from the reliable news source. The article that is compiled from the reliable news source, we verify that news in the different source because we justify the real news or not. Most of the data is collected from online that available for public data. We begin by collecting a set of legitimate news data that belong to six different domains (sports, business, entertainment, politics, technology and education). To ensure the accuracy of the news, we conducted a manual verification of the facts in the news content, which included the verification

of the news source and the information of cross-references between various sources.

Fake Data: The data we use for the malicious example comes from several sources. We obtained the list of articles that are known to have been generated by recent Bot nets or that participated in malicious activities. Our main source of blacklists came from the services. We also collect fake open source articles for researchers to use their theses [41]. It displays the most sophisticated item generator by carefully matching the frequency of appearance of vowels and consonants, as well as concatenating the resulting word. We note that most of these articles have been pronounced but are not in the English language dictionary.

Testing Data: The test data sets (unknown data) were obtained from online available publicly. The use of test data, which were already classified, is appropriate to compare the accuracy of our experiment and easily evaluate our approach. [9] As for the fake news, they were compiled from a fake news data set on kaggle.com. The data set collector collected false news from unreliable websites that Political fact has been working with Facebook to remove. We use 37000 fake news articles from kaggle.com and 41000 truth articles. We decided to focus only on the political news article because they are currently the main target of spammers. News articles from fake and true categories occurred on the same timeline, specifically in 2016. For each item, the following information is available article text, label (fake or truthful), title.

4.1.2 Dataset Construction

The quality of the results produced by the classifier depends to a large extent on the quality of the training set and the basic idea of this classification method is the training information that has labels corresponding to the content that is malicious or legitimate. To use the training data, we have to convert the list of items into a data set. We include about 37,000 malicious articles and 41000 legitimate articles. We had the conservation that when building the list of articles, some articles that are

extracted can be related to malicious activities. In our experiment, we divided the data set into one path with fake and real data. In that data set there are two classes, the first class for malicious labeled by false and the second class for legitimate labeled by real.

4.2 Data Exploration

To determine which features may be the most eective for classification, the following plots are constructed to visualize the relationship between select features of the malicious and credible articles.

4.3 Evaluation Matrix

To measure the performance of supervised machine learning methods, several measures of performance are used, such as recovery, accuracy, false-positive rate, precision, specificity, and measure F [42]. These measures can be easily derived from the confusion matrix of the model. The overall performance of the model is analyzed taking into account its performance in both training and test data. A model that is based on training data by learning each case or particularity in a precise way (adjusting in the best way) may not work well in the test data. The high performance only in the training data is not a good indicator of the overall performance of the model. Excessive adjustment of the model is one of the problems in the exercise of creating models. A good model should be able to generalize well in the test data where the test data is completely different from the training data. True positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are four components of the confusion matrix. The calculation [42] of several parameters is given below:

1. **Accuracy = TP + TN/ (TP+TN+FP+FN)**. It tells how well a binary classification test correctly i.e. what percentage of predictions that are correct.

Accuracy alone is not a good indicator, as it does not tell how well the model is in detecting positives or negatives separately.

2. **Precision = TP/ (TP + FP)**. It determines how many of the positively classified are relevant. It is the percentage of positive predictions correct. A high precision is desirable.

3. **Recall = TP/ (TP + FN)**. It explains how good a test is at detecting the positives. i.e. predicting positive observations as positive. A high recall is desired for a good model. Recall is also known as sensitivity or TP Rate.

4. **F-Measure = 2 * (Precision X Recall)/ (Precision + Recall)**. F-measure is a good indicator as it considers both precision and recall. A high F-measure is desirable.

Rates of precision, recall, measure f and false positives are calculated for both kinds of results (for example, yes and no) and are considered a weighted average when experimenting with each technique. In an experiment, no measure indicates how good a model is. The different measures discussed above have been considered in our experiment to evaluate several machine learning models to classify the fake content.

4.4 Experiments Procedures

We run the above-mentioned machine learning algorithms on the dataset, with the goal of predicting whether the articles are truthful or fake and identify the best classify. The experiments start by studying the impact of the size (n) of n-grams on the performance. We start with unigram (n = 1), then bigram (n = 2), trigram (n = 3). Furthermore, each n value was test combined with a tf-idf features. The experiments are run using cross validation; in each validation round the dataset is

divided into several training data and testing data for several time. The algorithms are used to create learning models, and then the learn models were used to predict the labels assign to the testing data. Then we find the best machine classifier. Experiment results are then presented, analyzed and interpreted. Thus, we decide to collect our dataset so we can require fake and real articles and use different data for different time. In total, we picked 77690 articles from real and fake articles we collected, 36000 fake articles and 41000 real articles.

Then we use the all classifier methods to identify the best classifier. We measure the performance with the parameter of accuracy, precision, recall and F-measure. For every classifier method we get different value of the parameters. The experimental procedure is given below step by step:

4.4.1 Data preprocessing procedure

In this paper, we will talk about the basic steps of text preprocessing. These steps are needed for transferring text from human language to machine-readable format for further processing. We will also discuss text preprocessing tools. After a text is obtained, we start with text normalization. Text normalization includes:

- Tokenization
- Stemming
- Lemmatization
- Stop word removal

We will describe text normalization steps in detail below.

- Tokenization is the process of splitting the given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens. We use tokenization process to split the text into tokens.

- Stop words are the most common words in a language like the, a, on, is, all. These words do not carry important meaning and are usually removed from texts. We remove this word by applying stop words removal process.

- Stemming is a process of reducing words to their word stem, base or root form (for example, booksbook, lookedlook). We use the stemming process to reduce the frequency of word.

- The aim of lemmatization, like stemming, is to reduce inflectional forms to a common base form. As opposed to stemming, lemmatization does not simply chop off inflections. Instead it uses lexical knowledge bases to get the correct base forms of words. We also use the lemmatization to convert the word into common base form.

4.4.2 Feature extraction procedure

Bag of Words (BoW) model is a way of representation of text which specifies occurrence (Eg. counts) of terms in a document. In this model, order and the sequence of words are not considered. Vector Space Model (VSM) is the improved version of BoW where each text document is represented as a vector, and each dimension corresponds to a separate term (word). If a term occurs in the document, then its value becomes non-zero in the vector. In this thesis, we will apply CountVectorizer that converts a collection of text documents to a matrix of term counts.

The problem with counting term frequencies is that the frequently used terms become dominant in the document and begin to represent the document. To solve this problem we will use anoher scoring model, TF x IDF, which stands for Term Frequency x Inverse Document Frequency. Model uses two metrics in its computation: term frequency (tf) and inverse document frequency (idf). To convert our documents to a matrix of TF-IDF features, we will use TfidfVectorizer.

4.4.3 Training procedure

After we have completed our initial data exploration analysis, we now prepare the data and train our models using the three describe methods. The data preparation involves the following steps:

- Set human readable column names on the data frame
- Replace the class data with descriptive label where fake and one marks a record as real
- Cast the class column to data type factor as the caret package complains if labels are real or fake
- Take samples from 1000 and split those into test and training sets randomly with a training/ test ratio of 70

4.4.4 Prediction and evaluation process

Finally, after we have completed the training step for all six models let us have a look how they compare to each other regarding performance. We compare the performance of all six approaches by evaluating the most commonly used indicators: precision (P), recall (R), accuracy (A) and F-measure. All three indicators originate from the confusion matrix of each model. We determine from the results that k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Naive Bayes (NB) and Long Short- Term Memory (LSTM) methods show their own precision, recall, accuracy and F-measure and we find the best classifier from using that parameters.

4.5 Model Construction and Evaluation

Using the training, tuning, and testing sets described in above, we trained and evaluated the KNN, SVM, LG, NB, RF, and LSTM algorithms. The following subsections discuss how each classifier was tuned to obtain the optimal performance, and state the results of each classifier.

4.6 Experiments Results

In our experimentation different indicators of fake content (unsolicited commercial articles) or real content have not been considered. The dataset consists of a total of 7796 articles, out of which 50% instances are real and 50% are fake. The dataset consists of an outcome variable (real or fake), frequency count of various words, and length of sequence of consecutive capital letters. The dataset is high-dimensional and complex in nature where spamers have used different strategies so that it would be difficult to identify a fake content. A good classifier builds model that is capable of generalizing and not overfitting to the training dataset. In this paper, holdout method is adopted where the entire dataset is divided to two mutually exclusive data sets: training and testing. The model is built on training data and then evaluated or tested on test data. It is found in literature that for a classifier. For our experimentation, 500, 1500, 300, 4500 and 6000 datasets observations are considered as training respectively rest of the datasets as test data set have been considered with a random selection procedure. Several different classification methods such as Bayes algorithms (Naive Bayes), SVM, Logistic Regression, K-Nearest Neighbors, Decision tree-based algorithms (Random Forest) are used in our experimentation. Various classification techniques are experimented on the dataset. We find the confusion matrix for different classifier using different size of datasets. Then we find the accuracy, precision, recall, F-measure for each classifier using different size of datasets. The experiments

results are in the following Table 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6 below:

Table 4.1: SVM Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	91	91	91	91
1500	92	92	92	92
3000	93	95	92	93
4500	94	95	93	94
6000	95	95	94	95

Table 4.2: Naive Bayes Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	88	88	87	87
1500	89	88	89	89
3000	89	91	87	89
4500	90	90	90	90
6000	92	92	91	91

Table 4.3: Random Forest Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	81	81	81	81
1500	80	80	80	80
3000	84	81	88	85
4500	85	85	84	85
6000	84	86	81	84

Table 4.4: Logistic Regression Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	88	89	88	88
1500	89	90	89	89
3000	91	93	88	91
4500	92	90	95	92
6000	93	93	93	93

Table 4.5: KNN Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	89	88	90	89
1500	91	92	90	91
3000	92	92	92	92
4500	92	93	92	93
6000	93	94	91	93

Table 4.6: LSTM Model: Classification results for five training data size

Training Data Size	**Metrics (%)**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
500	91	91	91	91
1500	91	91	91	91
3000	93	94	93	93
4500	94	94	94	94
6000	94	94	95	95

4.7 Observed Results

The results observed in terms of evaluation metrics. Scatter Plots for accuracy on a different set of data i.e 500,1500, 3000, 4500, and 6000 are shown below for various model:

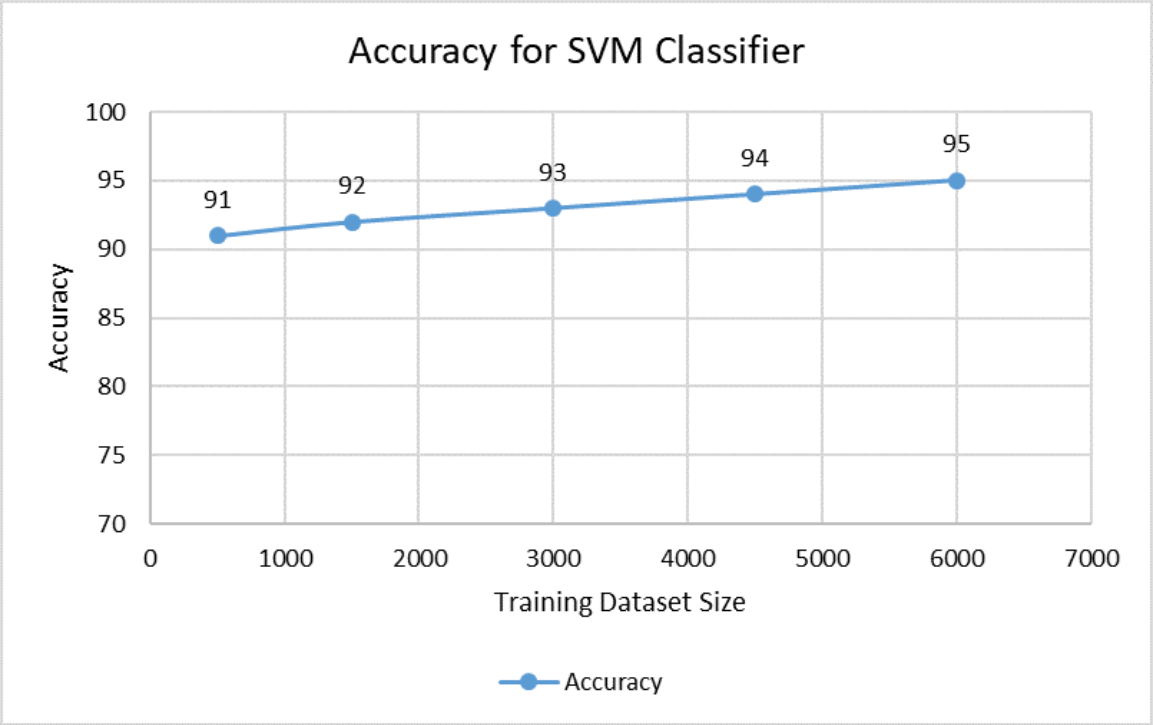

Figure 4.1: Scatter plot for SVM classifier model

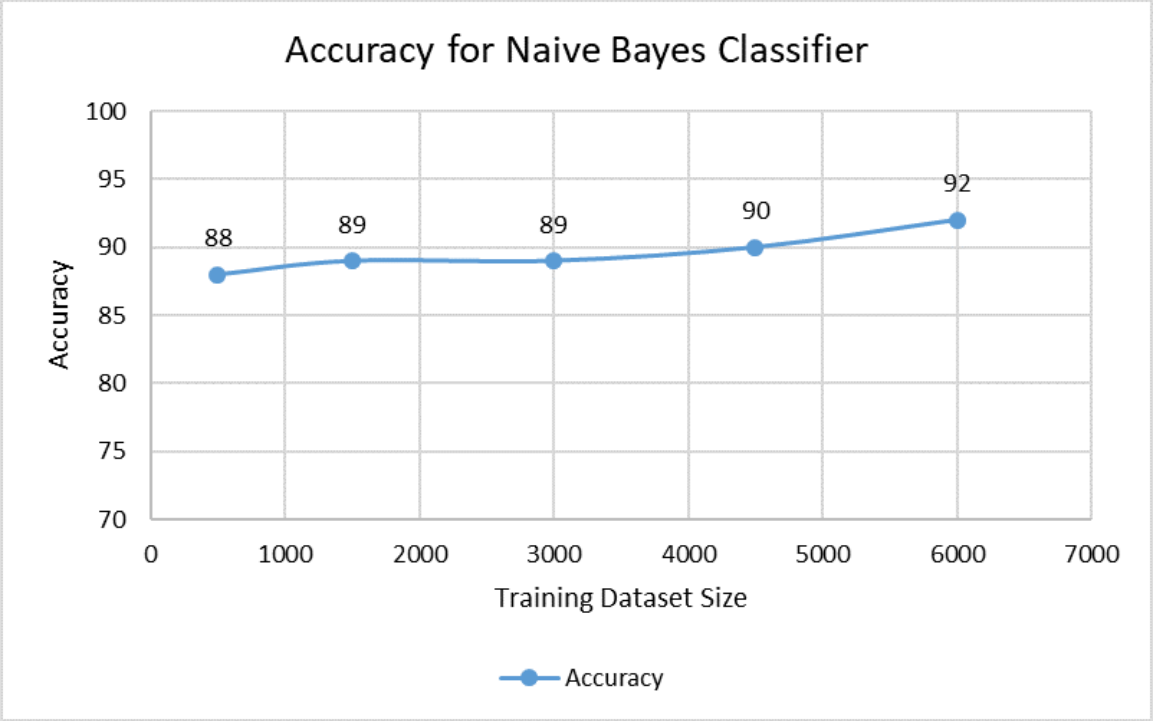

Figure 4.2: Scatter plot for Naive Bayes classifier model

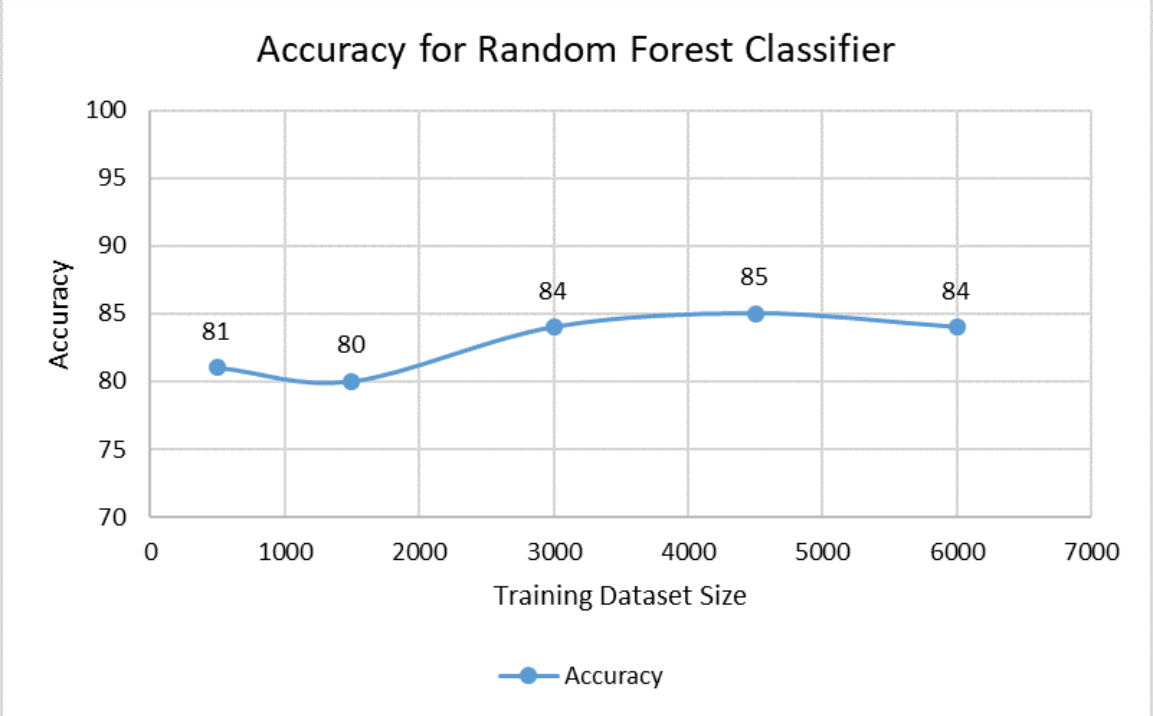

Figure 4.3: Scatter plot for Random Forest classifier model

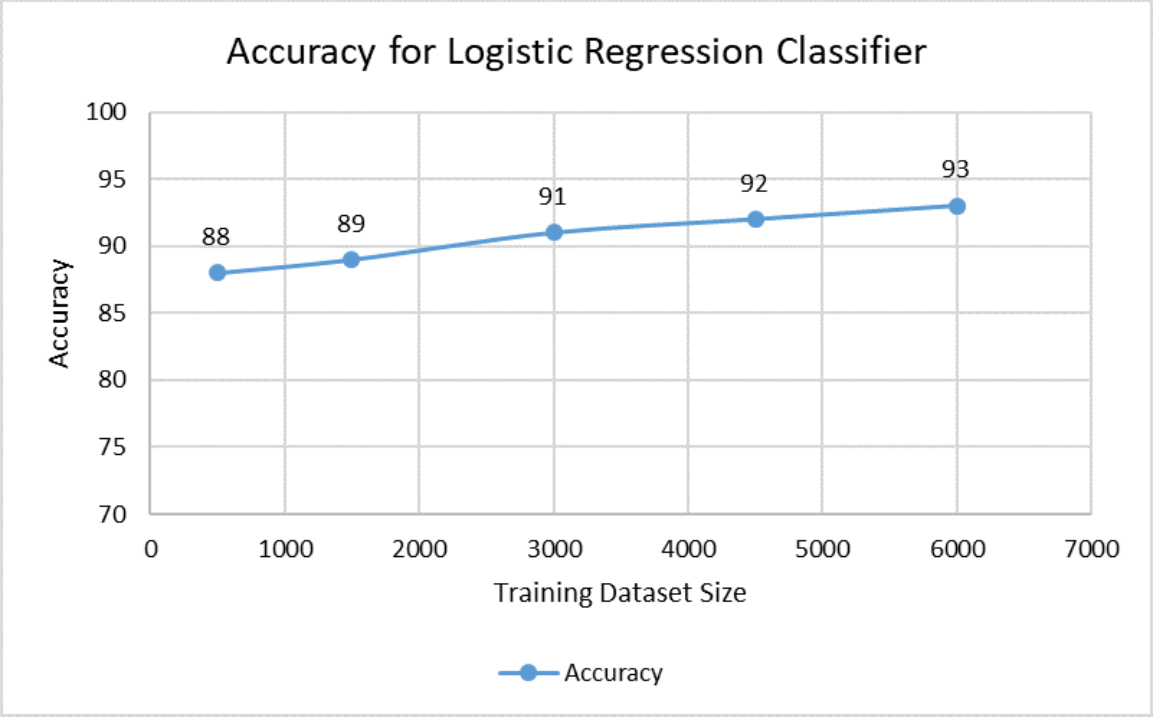

Figure 4.4: Scatter plot for Logistic Regression classifier model

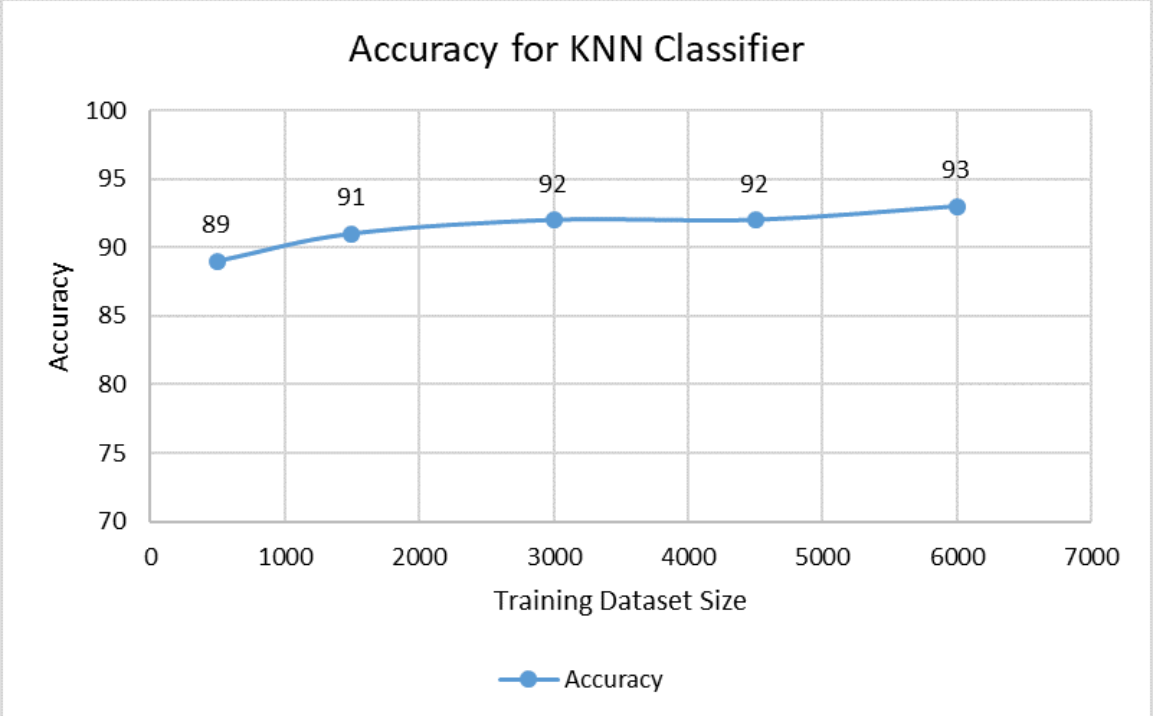

Figure 4.5: Scatter plot for K-Nearest Neighbors (KNN) classifier model

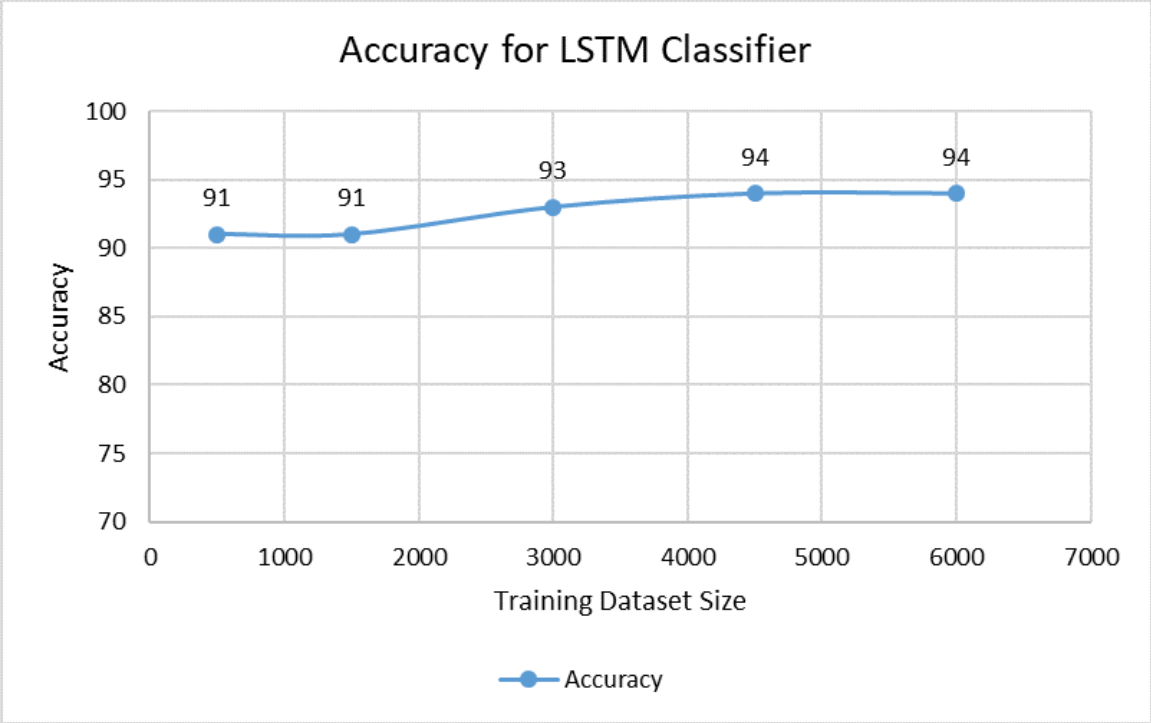

Figure 4.6: Scatter plot for LSTM classifier model

Out of all machine-learning techniques considered in the experiment, SVM found to be the best in terms of accuracy (95%) for the SVM model followed by the model LSTM (Accuracy-94%), KNN (Accuracy-93%), Logistic Regression (Accuracy-93%), Naive Bayes (Accuracy-92%) and Random Forest (Accuracy-84%) Figure 4.7. For the Precision SVM (Precision-95%) found to be the best followed by the model LSTM (Precision-95%), KNN (Precision-94%), Logistic Regression (Precision -93%), Naive Bayes (Precision -92%) and Random Forest (Precision -86%) Figure 4.8. For Recall LSTM (Recall-95%) found to be best and the others are SVM (Recall-94%), Logistic Regression (Recall-93%), Naive Bayes (Recall-91%), KNN (Recall-91%) and Random Forest (Recall-81%). Figure 4.9. For the F-measure SVM (F-measure -95%) and LSTM (F-measure -95%) both found to be the best followed by the model KNN (F-measure-93%), Logistic Regression (F-measure -93%), Naive Bayes (F-measure -91%) and Random Forest (F-measure-84%) Figure 4.10.

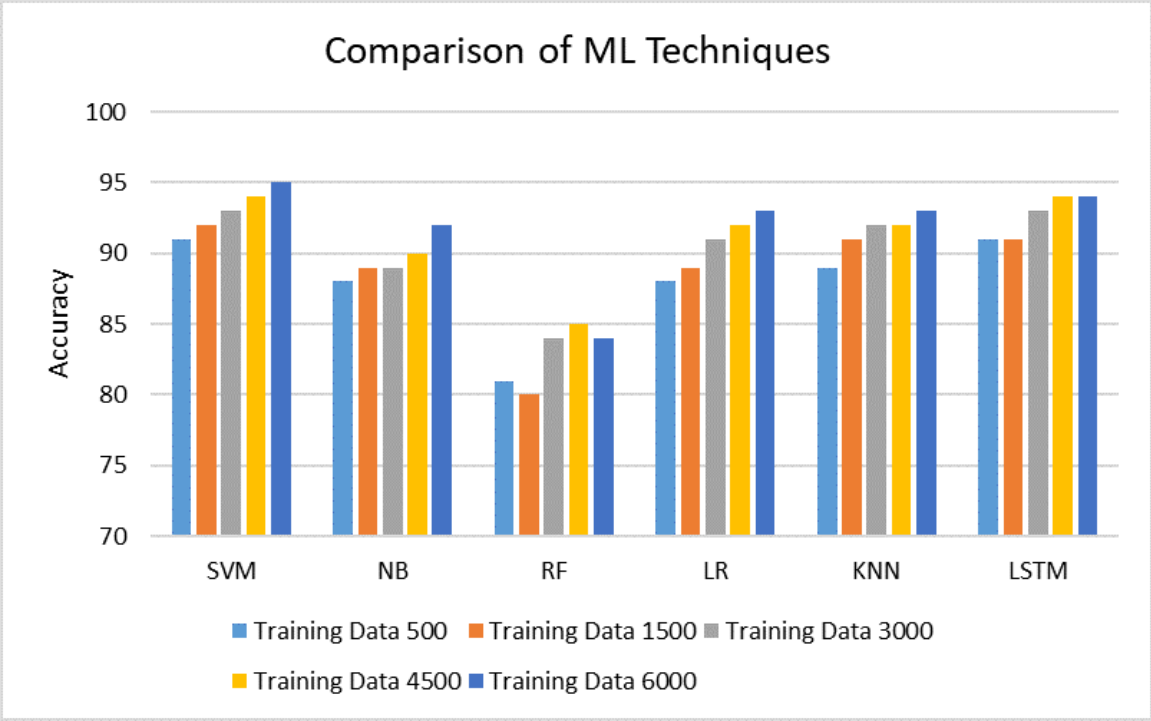

Figure 4.7: Comparison of Machine Learning Techniques: Accuracy

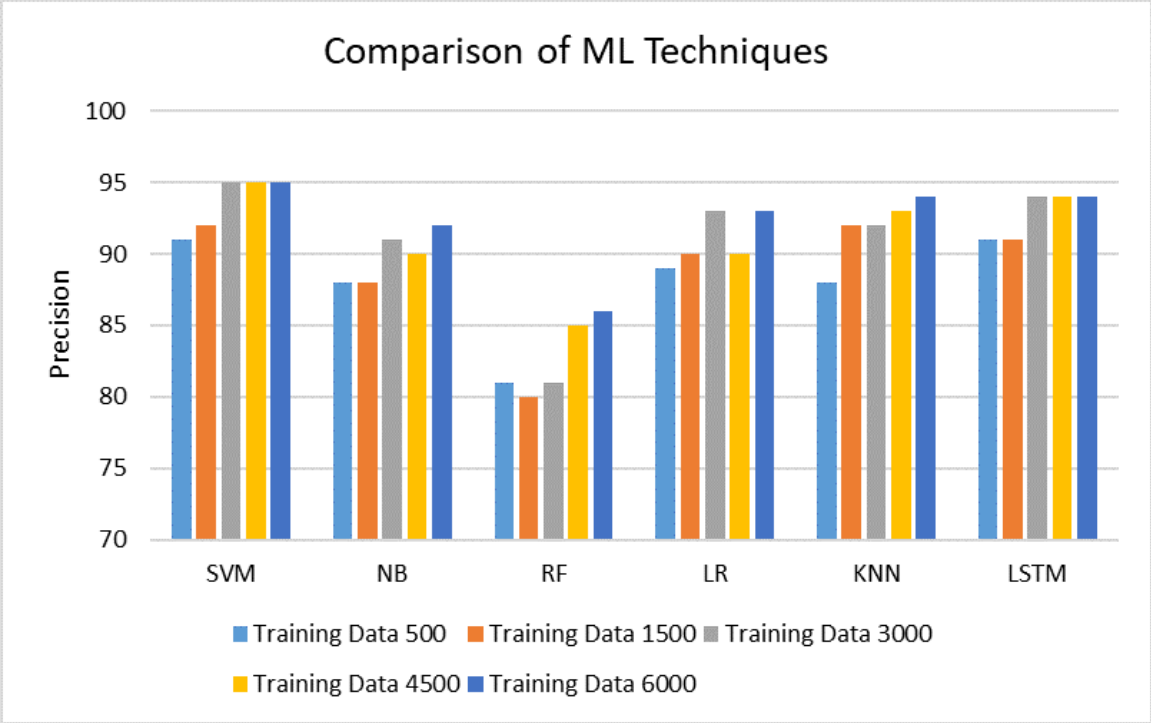

Figure 4.8: Comparison of Machine Learning Techniques: Precision

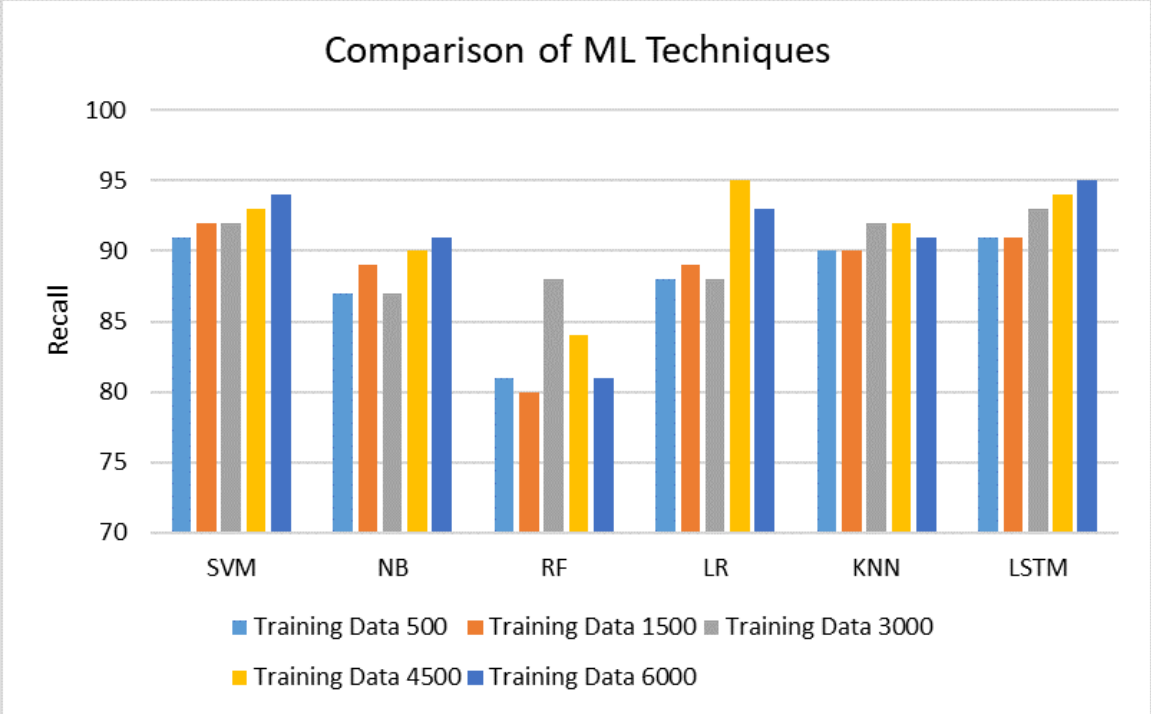

Figure 4.9: Comparison of Machine Learning Techniques: Recall

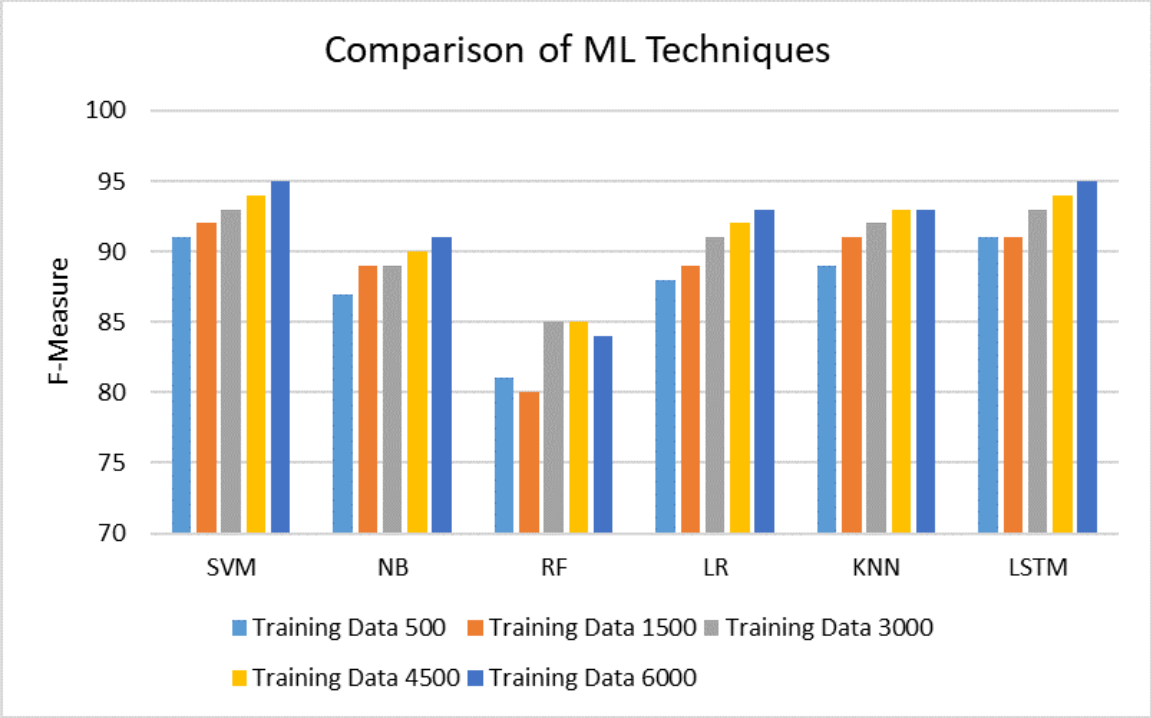

Figure 4.10: Comparison of Machine Learning Techniques: F-Measure

CHAPTER V

Conclusion and Future Work

5.1 Conclusion

Most of the popular fake news stories became more common on Facebook than the most popular mainstream newsletters [43]. A large number of people who read fake news stories have reported that they believe them more than news from mainstream media. Dewey [44] claimed that false news played a major role in 2016 US elections and that they continue to affect people's opinions and decisions.

In this paper, we compare various classification algorithms on fake content dataset through experimentation and it is found that Support Vector Machine (SVM) provides the best result among all the classifiers (95% Accuracy, 95% Precision, 94% Recall and 95% F-measure). SVM model is robust compare to all the category of algorithms considered because of its ability to predict best in spite of the fact that the dataset is noisy and sparse. Also it is observed that for the lowest training dataset size, the result is also over 90% for the SVM model. SVM model has also the best scalability features compare to other algorithms considered as it could able to construct the model efficiently when the dataset considered for experimentation is very large. On the other hand, the interpretability of neural network model is very low compare to random forest algorithm. Due to correlation between variables all other algorithms perform well on the dataset. One of the limitations of this study is that the machine

learning algorithms are tested on one dataset and need to on several public fake datasets.

5.2 Future Work

A complete, production-quality classifier will incorporate many different features beyond the vectors corresponding to the words in the text. For fake news detection, we can add as features the source of the news, including any associated URLs, the topic (e.g., science, politics, sports, etc.), publishing medium (blog, print, social media), country or geographic region of origin, publication year, as well as linguistic features not exploited in this exercise use of capitalization, fraction of words that are proper nouns (using gazetteers), and others.

Besides, we can also aggregate the well-performed classifiers to achieve better accuracy. For example, using bootstrap aggregating for the Neural Network, LSTM and SVM models to get better prediction result. An ambitious work would be to search the news on the Internet and compare the search results with the original news. Since the search result is usually reliable, this method should be more accurate, but also involves natural language understanding because the search results will not be exactly the same as the original news. So we will need to compare the meaning of two contents and decide whether they mean the same thing.

References

[1] Olah, "t-SNE Visualizations of Word Embeddings [Digital image]." http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/img/Turian-WordTSNE.png, 2014. [Online; accessed 15-November-2018].

[2] P. J. . S. R. Manning, C. D., "Comparative-Superlative [Digital image]." https://nlp.stanford.edu/projects/glove/images/comparative$_s uperlative.jpg$, 2014. [$Online$; $accessed 15 - November - 2018$].

[3] E. Kim, "Separable in Higher-Dimension [Digital image]." http://www.eric-kim.net/eric-kim-net/posts/1/imgs/data$_2 d_t o_3 d.png$, 2017. [$Online$; $accessed 15 - November - 2018$].

[4] C. Olah, "The repeating module in a rnn and lstm [digital image]." [Online]. Available from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/img, 2015. Online; accessed 15-November-2018.

[5] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[6] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 35–47, ACM, 2010.

[7] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to

us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*, pp. 551–560, ACM, 2009.

[8] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[9] V. Rubin, N. Conroy, Y. Chen, and S. Cornwell, "Fake news or truth? using satirical cues to detect potentially misleading news," in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pp. 7–17, 2016.

[10] E. Alwood, *Straight news: Gays, lesbians, and the news media*. Columbia University Press New York, 1996.

[11] Y. Chen, N. J. Conroy, and V. L. Rubin, "News in an online world: The need for an automatic crap detector," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[12] C. Condren, "Satire and definition," 2012.

[13] T. Lavergne, T. Urvoy, and F. Yvon, "Detecting fake content with relative entropy scoring.," *PAN*, vol. 8, pp. 27–31, 2008.

[14] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[15] P. Heymann, G. Koutrika, and H. Garcia-Molina, "Fighting spam on social web sites: A survey of approaches and future challenges," *IEEE Internet Computing*, vol. 11, no. 6, 2007.

[16] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter, and H. Al Najada, "Survey of review spam detection using machine learning techniques," *Journal of Big Data*, vol. 2, no. 1, p. 23, 2015.

[17] A. Bessi and E. Ferrara, "Social bots distort the 2016 us presidential election online discussion," 2016.

[18] R. Matthan, "Opinion — The challenge of detecting fake content." https://www.livemint.com/Opinion/rbLvtJE51jwOqE1nyayW8L/Opinion–The-challenge-of-detecting-fake-content.html, 2018. [Online; accessed 20-November-2018].

[19] N. Jindal and B. Liu, "Review spam detection," in *Proceedings of the 16th international conference on World Wide Web*, pp. 1189–1190, ACM, 2007.

[20] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 423–430, ACM, 2007.

[21] W. Y. Wang, "" liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

[22] T. Lavergne, T. Urvoy, and F. Yvon, "Detecting fake content with relative entropy scoring.,"

[23] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel, "A simple but tough-to-beat baseline for the fake news challenge stance detection task," *arXiv preprint arXiv:1707.03264*, 2017.

[24] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," in *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, p. 82, American Society for Information Science, 2015.

[25] L. Zhou, J. K. Burgoon, D. P. Twitchell, T. Qin, and J. F. Nunamaker Jr, "A comparison of classification methods for predicting deception in computer-mediated communication," *Journal of Management Information Systems*, vol. 20, no. 4, pp. 139–166, 2004.

[26] V. L. Rubin, Y. Chen, and N. J. Conroy, "Deception detection for news: three types of fakes," in *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, p. 83, American Society for Information Science, 2015.

[27] N. Davuth and S.-R. Kim, "Classification of malicious domain names using support vector machine and bi-gram method," *International Journal of Security and Its Applications*, vol. 7, no. 1, pp. 51–58, 2013.

[28] G. Ateniese, G. Felici, L. V. Mancini, A. Spognardi, A. Villani, and D. Vitali, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *arXiv preprint arXiv:1306.4447*, 2013.

[29] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[30] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," in *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, p. 82, American Society for Information Science, 2015.

[31] E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro, "Some like it hoax: Automated fake news detection in social networks," *arXiv preprint arXiv:1704.07506*, 2017.

[32] C. D. Manning, P. Raghavan, and H. Schütze, "Support vector machines and machine learning on documents," *Introduction to Information Retrieval*, pp. 319–348, 2008.

[33] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[34] V. V. Raghavan and S. M. Wong, "A critical analysis of vector space model for information retrieval," *Journal of the American Society for information Science*, vol. 37, no. 5, pp. 279–287, 1986.

[35] K. Zakka, "A complete guide to k-nearest-neighbors with applications in python and r," *Retrieved April*, vol. 24, p. 2018, 2016.

[36] J. Ham, Y. Chen, M. M. Crawford, and J. Ghosh, "Investigation of the random forest framework for classification of hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 492–501, 2005.

[37] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

[38] C. Olah, "Understanding lstm networks," 2015.

[39] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[40] B. D. Horne and S. Adali, "This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news," *arXiv preprint arXiv:1703.09398*, 2017.

[41] Kaggle, "Datasets: Fake or real data." https://www.kaggle.com/datasets, 2018. [Online; accessed 15-November-2018].

[42] E.-S. M. El-Alfy and R. E. Abdel-Aal, "Using gmdh-based networks for improved spam detection and email feature analysis," *Applied soft computing*, vol. 11, no. 1, pp. 477–488, 2011.

[43] C. Silverman and J. Singer-Vine, "Most americans who see fake news believe it, new survey says," *BuzzFeed News*, 2016.

[44] C. Dewey, "Facebook has repeatedly trended fake news since firing its human editors," *Washington Post*, vol. 12, 2016.

Appendices

Appendix A

Coding Summary

1.1 Data Preprocessing

```
eng_stemmer = SnowballStemmer('english')
eng_lemma = SnowballStemmer('english')
stopwords = set(nltk.corpus.stopwords.words('english'))
#Stemming
def stem_tokens(tokens, stemmer):
    stemmed = []
    for token in tokens:
        stemmed.append(stemmer.stem(token))
    return stemmed
#Lemmatizing
def lemma_tokens(tokens, lemma):
    lemmatize = []
    for token in tokens:
        lemmatize.append(lemma.wordnet_lemmatizer
            .lemmatize(tokens))
    return lemmatize
```

```
#process the data
def process_data(data,stopword=True,stem=True,lemma=True):
    tokens = [w.lower() for w in data]
    tokens_stemmed = tokens
    tokens_stemmed = stem_tokens(tokens, eng_stemmer)
    tokens_stemmed
    = [w for w in tokens_stemmed if w not in stopwords]
    return tokens_stemmed
    tokens = [w.lower() for w in data]
    tokens_lemmatize = tokens
    tokens_lemmatize = lemma_tokens(tokens, eng_lemma)
    tokens_lemmatize
    = [w for w in tokens_lemmatize if w not in stopwords]
    return tokens_lemmatize
X_train = process_data(X_train)
#creating ngrams
#unigram
def create_unigram(words):
    assert type(words) == list
    return words
#bigram
def create_bigrams(words):
    assert type(words) == list
    skip = 0
    join_str = " "
    Len = len(words)
    if Len > 1:
```

```
        lst = []
        for i in range(Len-1):
            for k in range(1,skip+2):
                if i+k < Len:
                 lst.append(join_str.join([words[i],words[i+k]]))
    else:
        lst = create_unigram(words)
    return lst
#trigrams
def create_trigrams(words):
    assert type(words) == list
    skip == 0
    join_str = "␣"
    Len = len(words)
    if L > 2:
        lst = []
        for i in range(1,skip+2):
            for k1 in range(1, skip+2):
                for k2 in range(1,skip+2):
                    if i+k1 < Len and i+k1+k2 < Len:
                        lst.append(join_str.join([words[i],
                            words[i+k1],words[i+k1+k2]]))
        else:
            lst = create_bigram(words)
    return lst
X_train = create_bigrams(X_train)
```

1.2 Feature Selection

```
#Naive Bayes Classifier
clf = MultinomialNB()
clf.fit(count_train, y_train)
pred = clf.predict(count_test)
score = accuracy_score(y_test, pred)
print("MultinomialNB accuracy:   %0.3f" % score)
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

#Random Forest Classifier
linear_clf = RandomForestClassifier()
linear_clf.fit(count_train, y_train)
pred = linear_clf.predict(count_test)
score = accuracy_score(y_test, pred)
print("RandomForestClassifier accuracy:   %0.3f" % score)
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

#Logistic Regression Classifier
linear_clf = LogisticRegression()
linear_clf.fit(tfidf_train, y_train)
```

```
pred = linear_clf.predict(tfidf_test)
score = accuracy_score(y_test, pred)
print("LogisticRegression accuracy:   %0.3f" % score)
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

#SVM Classifier
linear_clf = svm.LinearSVC()
linear_clf.fit(tfidf_train, y_train)
pred = linear_clf.predict(tfidf_test)
score = accuracy_score(y_test, pred)
print("svm.LinearSVC accuracy:   %0.3f" % score)
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

#KNN Classifier
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(tfidf_train, y_train)
y_pred = knn.predict(count_test)
print("KNN Accuracy:",metrics.accuracy_score(y_test, pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
#LSTM Classifier
linear_clf = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter
linear_clf.fit(tfidf_train, y_train)
pred = linear_clf.predict(tfidf_test)
score = accuracy_score(y_test, pred)
print("LSTM accuracy: %0.3f" % score)
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

1.3 Classifier Model and Final Results

```
def classification(vectorizer, classifier, n=100):
    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1
        = sorted(zip(classifier.coef_[0], feature_names))[:n]
    topn_class2
        = sorted(zip(classifier.coef_[0], feature_names))[-n:]
    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)
    print()
    for coef, feat in reversed(topn_class2):
        print(class_labels[1], coef, feat)
classification(tfidf_vectorizer, linear_clf, n=30)
feature_names = tfidf_vectorizer.get_feature_names()
sorted(zip(clf.coef_[0], feature_names), reverse=True)[:20]
```

```
### Most fake
sorted(zip(clf.coef_[0], feature_names))[:20]
tokens_with_weights
    = sorted(list(zip(feature_names, clf.coef_[0])))
for i in tokens_with_weights:
    print(i)
    break
```