Distinguishing Fake and Real News of Twitter Data with the help of Machine Learning Techniques

by

Aanan Shah

A thesis submitted in partial fulfillment

of the requirements for the degree of

MSc Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Distinguishing Fake and Real News of Twitter Data with the help of Machine Learning Techniques	
Name of Candidate Nom du candidat	Shah, Aanan	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance March 30, 2021

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Oumar Gueye
(Committee member/Membre du comité)

Dr. Aniket Mahanti
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. Lace Marie Brogden
Madame Lace Marie Brogden
Acting Dean, Faculty of Graduate Studies
Doyenne intérimaire, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Aanan Shah**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Abstract

News articles have an influence on people's belief and views about various circumstances. In this regard, some news publishers with political or ideological bias try to spread news which are distorted or totally wrong. This thesis intends to develop a machine learning model that identifies fake news and original news by taking aid from natural language processing. Natural language processing was used to preprocess the text. Some general features like, number of words, sentences, stopwords, non-alphabetic words, verbs, nouns, and adjectives were identified. The stopwords and hyperlinks were removed to clean the text data. In the preprocessing step after cleaning the data and removing the stopwords, the position of each word was concatenated with the word itself. This procedure helps in distinguishing between a word as a noun, a pronoun, an adjective or a verb in the sentences. After preprocessing, feature extraction methods were used for converting the text of news to analyzable data. The frequency of the words in each article was used for filtering out the non-informative words. Three feature extraction methods were used in this study namely, count vectorizer, Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer and word2vec embedding. It was observed that the results obtained by TF-IDF feature extraction method were superior compared with the other two methods. After feature extraction, various machine learning models were used for training the model namely, Naive Bayes, Logistic Regression, Random Forest, K-nearest neighbors (KNN) and Support Vector Machine (SVM). The Recurrent Neural Network (RNN) was also used as a deep learning model. The model was successfully tested on two datasets. On the first dataset, SVM achieved an accuracy of 98.5% and RNN achieved an accuracy of 98.03% which is much improvement over the best results of Agarwalla et al., 2019 (83.16 % accuracy). On the second dataset, SVM achieved an

accuracy of 97.76%, RNN achieved 97.1% and Logistic Regression achieved 97.50% which is an improvement over the best results of Vijayraghavan et al. 2020 (94.88% accuracy).

Acknowledgements

The thesis is the outcome of the hard work and support of different people involved directly and indirectly throughout, so I cannot keep myself away from acknowledging them.

First and most of all, I would like to thank my almighty God for successfully completing this thesis.

Secondly, I would like to especially acknowledge my thesis supervisor, Dr. Kalpdrum Passi, for his supervision, expertise, comments and suggestions, assistance, and patience throughout researching and writing of this thesis. His constant support, belief and inspiration played a vital role in bringing this thesis to the edge.

I would like to thank my committee members for their generous and helpful suggestions and encouragement.

I would like to praise my loving parents and family members, who have continuously supported and inspired me throughout my study, research, and thesis writing. Also, to my fiancé, thank you for being around and for never-ending motivations throughout my study.

Lastly, I am much thankful to all my friends and everyone else who have not been mentioned in this thesis work but contributed in one or many ways to complete this thesis. Thank you all.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The era of computing has observed a rise in the information sharing among the people. Before the era of the Internet and mobile communications the information exchange among people was limited by the postal and other services that were time consuming. The mass communication through the Internet and mobile communication has surprised the world with its great facilities of providing fast information and brought people closer to each other through text, voice and video communication. There is rapid availability of latest news over social media in addition to the television medium. The authenticity and reliability of the news is many times compromised and it influences the people positively or negatively depending on the news they receive from television and social media. The credibility and dependability of the news through different media is questionable in many instances.

Today, the daily information exchange throughout the world is highly dependent on the mass media. The entire local, national, and international news are part of the daily mass communication media, and people depend highly on the news that they listen to or see on their television, computers, or mobile screens.

We also have social media platforms to interact with the people where millions of news varieties are uploaded daily and exchanged. These media sites include Facebook, Twitter, WhatsApp, and various others. Despite shrinking the distances between places and circulating the news rapidly

all around, the social information exchange mediums are somewhat unpredictable and unreliable at spreading the news to the community. When a person becomes a part of the social media community, there are different posts and information available for them to see and absorb content from them. Because millions of posts are available at the social media sites, sometimes some users might make a mistake to decide if the news from the post they have seen is real or fake. Above all, this fake news circulation strategy not only focuses on the business growth stuff for product selling purposes but also targets celebrities, politicians, and different famous personalities. This task aims to spread rumors about the targets among the community and manipulate the facts. The situation gets worse when people blindly start to share the news even further. It not only exaggerates the situation but also results in stressful and unwelcoming outcomes.

Along with advancements in the media sector and social networking platforms, the emergence of Artificial Intelligence (AI) in the computing and research sector has revolutionized the world with its enormous applications into various fields of life [1]. These applications find their way into Engineering, Computer sciences, biomedical sciences, business, and many other fields. The winter of AI has taken a humungous turn and has evolved into a field of never-ending growth possibilities. This has become possible with the advancements in electronic gadgets through the Very Large-Scale Integration (VLSI) technology and the development of high-resolution cameras leading toward greatly improved imaging. The sub-fields of AI, such as machine learning and deep learning, have provided substantial benefits to the designers and the Engineers to dig deep into the fields and research plenty of subjects dealing with real life. Figure 1.1. shows the contrasting fields where AI has proved highly promising in the development and research sectors.

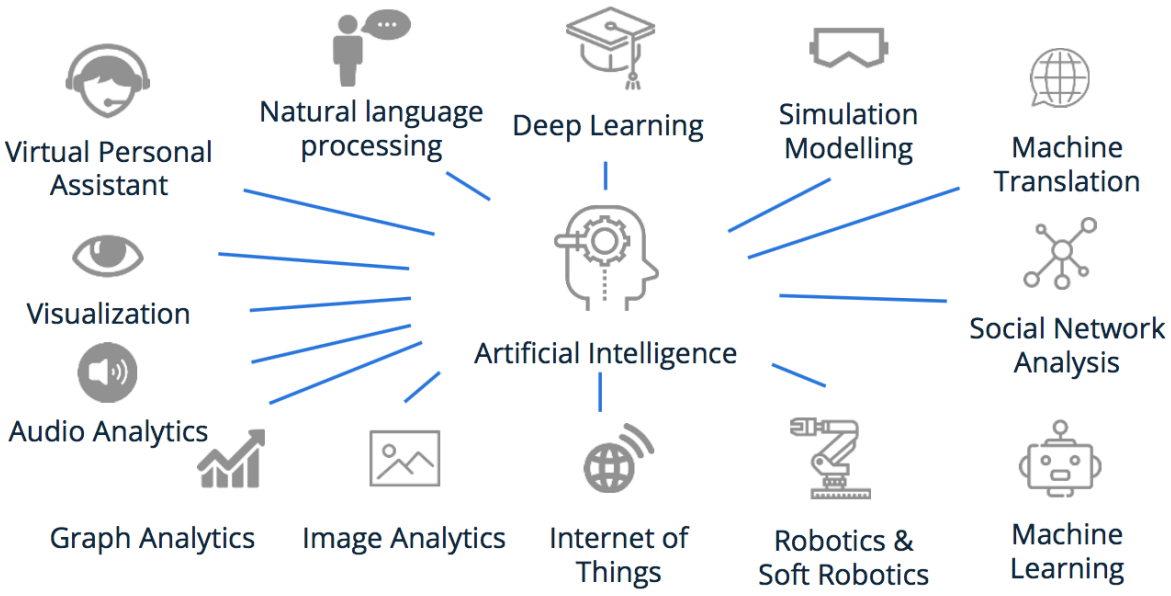


Figure 1.1. *Application areas of Artificial Intelligence (AI)[2]*

It can be observed that there is not a particular field in which AI has its applications. There is a vast majority of such fields, including medical, modeling and simulation, social networking analysis, language processing, graph, audio or video analytics, robotics, or visualization [2]. AI's primary advantage is that it does not require complex modeling and design instead relies on simple equations, but the quality of the data for training and testing the predictive models for the required application must be highly accurate for good results.

Undoubtedly, the field is continuously emerging and has great significance in solving real-world queries. The interlinking of AI with Computer Vision provides rapid growth in the technology and research sectors where extracting information from images is of utmost importance. There are thousands of image datasets present over the internet which are utilized and tested for various AI-based applications development.

The fake news detection also has become more interesting with the involvement of AI. The problem of identifying news as a real or a fake belongs to the class of Natural Language Processing (NLP) [3] under the AI domain, which deals with interpreting the natural languages such as English, Chinese or any other by the software. The NLP algorithms are used to train the computer to read and decode the human language and extract valuable information out of it. These algorithms are machine learning and deep learning-based solutions to the applications of NLP.

All the application areas in AI go through a complete modeling process to detect the outcomes and provide predictions. The AI-based applications' major workflow is discussed in detail further to provide insights into the field at a general abstract level.

1.2 Artificial Intelligence (AI) Workflow

Artificial Intelligence has two main categories: Artificial Narrow Intelligence (ANI) and Artificial General Intelligence (AGI). ANI is more realistic, and much work is being done in this field in various domains. ANI deals with accomplishing specific tasks rather than general ones that perform all kinds of tasks. This general intelligence comes under AGI, a highly complex field, and is not much realizable. It is predicted that it may take years for AGI to evolve.

ANI has dealt with many applications such as smart speakers, self-driving cars, web researching, farming and agriculture, robotics, and numerous others. Despite different fields of applications and completely different frameworks of all these fields, various steps are commonly involved in all AI-based applications. These steps' insights vary as per the task's requirements under consideration, but all these steps' primary purpose remains the same.

In AI and machine learning, the machine is trained to learn the parameters from the data or the targeted attributes. This learning can be supervised or unsupervised depending on the data and the targeted model. Supervised learning is a way in which a machine learns the mapping from input to output or say A to B. It means that the provided data in supervised learning is labeled. These types of models are called as a "Teacher" in the machine learning domain. Most machine learning algorithms uses supervised learning. The aim is to find the approximation function, which best maps the input to the correct output whenever any new data is provided to the model. A threshold is set to stop the learning/ training when performance reaches the desired limit.

In the case of unsupervised learning algorithms, the provided data is unlabeled, and the task of the model is to extract valuable insights into the data. It filters out the data's underlying dynamics and structure or finds the distribution of the data. This helps to further research on the data and learn more insightful parameters. There is another important area in machine learning called reinforcement learning. Reinforcement learning sequentially deals with the agents and reward-based learning approach in which the learning is done to maximize the reward. Here the reinforcement agent has the responsibility of decision making. It aligns a strategy and decides what operations should be performed on the given data to perform the job. In this way, various solutions could be obtained, and the best suited could be finalized based on the maximum reward. The examples of supervised learning may include the following.

- Spam Filtering
- Speech Recognition
- Language/Machine Translation
- Online Advertising

Table 1.1. *Supervised Vs. Unsupervised Machine Learning Algorithm*

	Supervised Learning	Unsupervised Learning
Continuous Data	Regression (Linear, Logistic, Polynomial) Decision Trees Random Forest	Clustering and Dimensionality Reduction Space Vector Decomposition (SVD) Principle Component Analysis (PCA) K-Means Clustering
Discrete or Categorical Data	Classification K-Nearest Neighbors (KNN's) Decision Trees Logistic Regression Naïve Bayes Support Vector Machines (SVM)	Association Analysis Apriori FP-Growth Hidden Markov Models (HMM) Clustering

Similarly, unsupervised learning applications may include,

- Pattern recognition
- Anomaly detection
- Feature extraction
- Item categorization
- Clustering customers
- Identity management
- Similar item recommendations
- Performance monitoring
- Visual Recognition

Table 1.1 provides the grouping with model examples for supervised and unsupervised learning methods for continuous and discrete data types. The continuous data may involve speeches, radar signals, etc. and discrete data. Examples involve images, clustered data, etc. It could be observed in Figure 1.2. that the steps seem like a human brain working, which means that we can create an analogy between the AI workflow and a human's process to undergo any task. We shall be discussing each step separately with its significance and details.

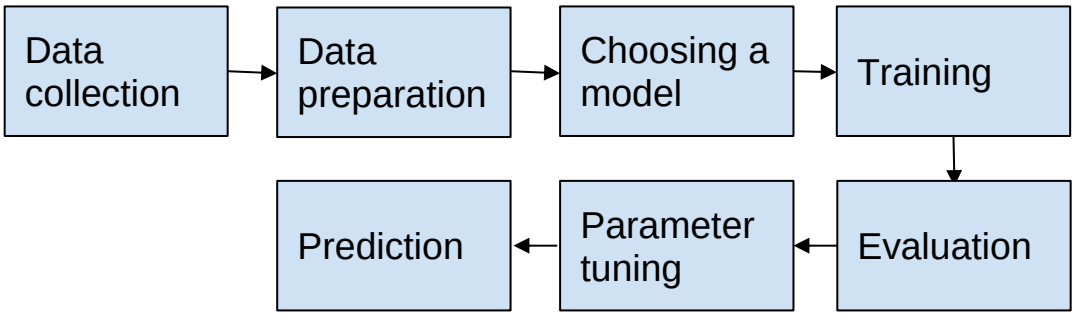


Figure 1.2. *Steps involved in Artificial Intelligence (AI) based task.*

1.3 Data Collection

The first step involved in AI or machine learning-based solutions is the collection of data. This is the most critical step and the most important one as it can make or break the rest of the task purpose [4]. When a human wants to get some tasks done, he first makes up his mind and collects the necessary information related to the task. The more refined information a person receives, the better is the further interlinking strategy to produce good results. Similarly, the data collection stage is of evaluative importance as the rest of the system's accuracy is completely based on the data's quality. If the data is irrelevant, diminutive, or too vague, it will not work at its best for the designed application [5].

In the problem of fake news detection, this data collection could be based on the news posted on social media sites daily. Millions of such news stories could be collected, and a database could be generated. These datasets are already collected and outsourced for the researchers, and repositories are already created for easy access to help the research community. One of the well-known dataset sites is the Kaggle [6].

1.3.1 Data Preparation

The data preparation stage involves the preprocessing of the data to transform it in a way that is understandable by the machine algorithm. This data acquisition stage has its importance as the output data from this stage is directly fed to the model for training. The purpose of this stage is to refine the data and remove unwanted information from the data. In other words, the shortcomings during the data collection stages are removed at this stage [7].

Think of a process a human goes through after collecting the data, 'filtering.' He removes the unwanted information and focuses only on that part of the essential information to help process the task. Similarly, to make the process more relevant in machine learning applications, the

collected data is refined, and unwanted information is filtered out [8]. Various steps are performed on the data at this preprocessing stage. These steps may involve,

- Data Quality Assessment
- Feature Aggregation
- Feature Sampling
- Dimensionality Reduction
- Feature Encoding

We can use various strategies for feature extractions for our problem of fake news detection, including Count Vectorizer, TF-IDF vectorizer, or word2vect embeddings. This preprocessing is essential in the NLP application to prepare the data in machine-readable formation. Other processing may include removing stop words, stemming, phrasing, lemmatizing, semantics, or tokenization. Not all these steps need to be performed on all kinds of data. It depends on the requirement and the collected data type. Respectively, the operation is performed, and the data is made entirely ready for further processing. The news data is preprocessed by applying various techniques, and much refinement is done at this stage to gather good results.

1.4 Choosing a Model

A model in machine learning represents what has been learned by the algorithm. It comprises of the learned parameters after running the model on the training dataset. This sub-module in the process flow of AI applications deals with selecting the most suitable statistical machine learning predictive model out of the available ones that seems best for the intended problem. It is a real challenge for the applied machine learning engineers to compare the models, their limitations,

and the best choice for modeling a particular problem. The model's selection could be based on multiple levels of abstractions, depending on the application [9]. First, we select the hyperparameters at which we want to structure our modeling results. These parameters are specified before the training. This stage is basically termed as algorithm selection based on the hyperparameters. These metrics become fundamental for selecting the right model for the application. More specifically, the model selection could be based on the following points:

- Does the model meet the requirements of the problem to be solved?
- Does the intended model is sufficiently skillful in terms of time and resources?
- Does the model perform better as compared to the naïve models available?

There may involve other metrics to choose the model in addition to the discussed ones. Every model has its own way of extracting the results. The models vary in the order of feature engineering over the provided data and are all different in dealing with the provided information.

There are numerous models available for implementing machine learning tasks. The correct model's deployment is necessary and is based on the target that one wants to predict. When detecting fake news, the data is discrete as well as categorical, and to learn through this type of data using supervised learning, the following modeling schemes could be utilized and implemented.

- Support Vector Machines (SVM)
- Logistic Regression
- Naïve Bayes Classifier
- Decision Trees
- Random Forest

- K-Nearest Neighbors (KNNs)
- Recurrent Neural Networks (RNN)

Each of these models has its significance and strategy to tackle the provided data and make predictions. These models differ from one another in their weight assigning scheme, data holding strategies, rules and numbers, and the underlying data structures. Further, we will briefly discuss each of these classification models for machine learning applications in their general forms. These could be specifically manipulated for the respective applications, but the algorithm's basic general strategy remains the same.

1.5 Model Training

The machine learning and AI specify to teach a machine to learn itself by providing real-world data samples to estimate the model's performance on the real parameters. These data samples are termed as training features for the machine. The machine is required to learn from these training features, and after the training is complete, it should be capable of predicting the correct results for future data, which is entirely new for the machine and unseen. The available data is divided intelligently into sub-data groups called as training data, validation data, and testing data within a particular ratio to carry out this whole process.

The model training, evaluation, and parameter tuning are all done in a feedback manner and are all linked as specified in Figure 1.3. As discussed in section C, different models are taken and trained for fake news detection and then evaluated and tested to compare their accuracy.

1.5.1 Model Evaluation and Parameter Tuning

After the model's training is done, and the weights are decided based on the training, we perform the model evaluation or say validation. The validation data group is used to analyze the trained

model performance. Repeated application of performance evaluation at the validation dataset is performed, and the best performing model is picked out from the trained model. Also, the trained model gets refined and tuned this way, and the prediction becomes unprejudiced later. This process is done by looping through the feedback path and gets a break when the best model for the data is achieved.

If the model starts to fit too well to the training data, it becomes biased. Another significant advantage of using the validation strategy is that it prevents the model from overfitting the training data. This reduces the chance of biased estimations at the prediction stage. Therefore, it is beneficial to have large training data, which ultimately reduces the overfitting and increases the validation score, which results in a good performance on the generalized test dataset.

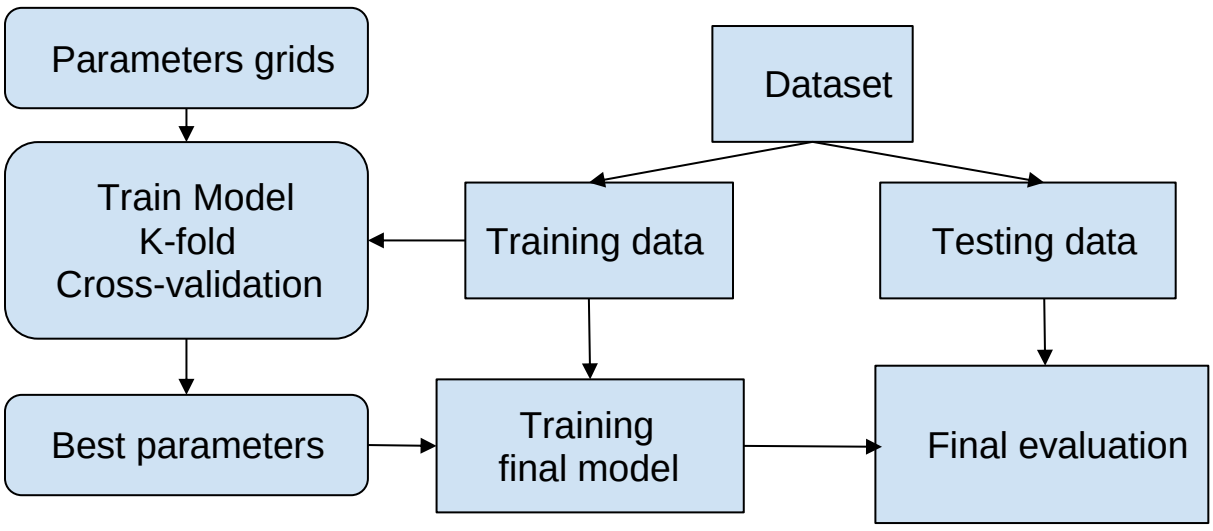


Figure 1.3. *Training, Testing and Validation*

1.5.2 Prediction

When the data is split into train, test, and validation groups, the test set is kept aside for later use in predictions at the final stage. The performance of the trained and validated model is evaluated

using the test data. Ideally, it should be used only once on the model; otherwise, the model becomes biased and provides a very high performance, which is the wrong prediction. All these steps are necessary for any machine learning based predictive modeling and analysis for detecting false news. The model out of the discussed ones is selected, and further training and testing are done based on that.

1.6 Motivation

Nowadays, social media is increasingly becoming important for dissemination and consumption of news as it is readily available at virtually no cost within a short time. However, it also has a dark side of allowing the spread of false news, negative comments, aggressive and duplicate content, etc., which is causing negative impacts on both individuals and the community by manipulating their minds. The leading reason is that fake news can be easily published at a fast speed without much cost than traditional news media such as newspapers and TV. Sometimes it might be fake because of any opposition, by intention or any reason. Many people are getting affected by it, as after they read it, they try to spread it because they have no idea whether the news is 100% original or not, which directly or indirectly hurts individuals, consumers, or politicians, which is not suitable for the world.

Considering the above points motivated us to use technology to understand whether the news is fake or not and if it is fake, how much it is fake and how much it is true. After researching, we discovered that some researchers [11] have already done this task and have also got the results using the twitter dataset, but the results are still not as good as possible. This work on fake news discovery was undertaken to improve the results on the same dataset and other datasets by

implementing various machine learning methods and techniques to increase accuracy. When the accuracy is high, people can easily trust the authenticity of news, which saves the community from fake news.

1.7 Study Objectives

To attain media attention and create propaganda, "Fake news" is circulated, and it results in bad impacts of the leading personalities and manipulates the correct information and the events. It also results in exaggeration among the community due to false reporting of the contents. Today, there are many social media sites such as Twitter, Facebook, etc. where such manipulators are actively working day and night to spread fake news about the people and spread the wrong content.

Such motivations of spreading wrong information and fake news could be demolished by utilizing advanced technologies for their detections. To perform such a task, schemes such as Artificial Intelligence (AI), Machine Learning (ML), and/or Deep Learning (DL) could be adopted. It must also be noted that it is difficult for some people to identify the news as a real or a fake one for the machine. Like the human brain, the machine needs to gather all the information, summarize it, extract the hyperparameters, and compare them with the trained model parameters to report it as a fake one. It also involved NLP's application to recognize the semantics used in the news phrasing and many other aspects for more robust detection.

This study has a motivation to help social media sites identify fake news on their platforms. It helps the community get the right information and protects the targeted people's ethical rights to provide them a secure communication medium with the public. The study has been influenced by

the work done in [36] and has undertaken some additional techniques to develop a fake news detection mechanism by utilizing deep learning and the NLP approach.

Chapter 2

Literature Review

Recent literature is discussed in this chapter on the distinguishing proof of fake content utilizing electronic broadcasting, acknowledging bots through online broadcasting, and alluring messages on express races using web broadcasting. Most of the current review has been focused on orchestrating web news and online broadcasting posts. In simpler words, Fake News is a report that is deliberately sham and could hoodwink readers. This tight definition is essential as it can take out the vulnerability between Fake News and other related thoughts, e.g., manufactures and farces [10]. Here we review recent research in the detection of fake news using a machine learning approaches.

Agarwalla et al. [11] studied to examine how mass media influences the general public's life and how it happens. Even though it is a testing job to choose the correct quality and measure of information, there are scarcely any principles for AI on massive data that should be followed. The dataset was taken from kaggle.com in this venture. The dataset dimension is 4008 rows and 4 columns. "URLs", "Function", "Body" and "Name" are the names of the columns. It is seen from the dataset that 2136 false news storeys and 1872 genuine news stories are available. In their method, each word was considered individually (1-gram) in the feature matrix. The

maximum accuracy of 83.1% was achieved by Naïve Bayes classifier with Lidstone smoothing on the specified preparation set. However, 74% accuracy was achieved in previous models with Naïve Bayes (without Lidstone smoothing) [11]. In those models Logistic Regression was used where the learning rate (α) was the critical boundary. The learning rate between 5 and 12 offered the same mixing point, and an approximation of 10 was therefore used. The model brought about exceptionally low accuracy 65.8%. SVM was also used for classification, which achieved an accuracy of 81.6%.

Looijenga [12] investigates how during the 2012 Dutch parliamentary election campaign, fake messages were used on Twitter. It examines the performance on a Twitter dataset of 8 guided Machine Learning classifiers. The authors claim that with an F1-Score of 88%, the Decision Tree performs the best on the used dataset. Out of 613,033 tweets 328,897 were identified to be real and 284,136 were identified fake tweets. The bogus substance's clear highlights and characteristics were found and gathered into six distinct groups through a subjective substance investigation of bogus tweets sent throughout the political campaign. They have used an existing Twitter dataset for this analysis. The archive of information contains tweets written about the election campaign on September 12, 2012. Looijenga [12] investigated how, in the 2012 Dutch elections, online citizens persuaded individual voters. To collect relevant hashtags, they used the logic of the snowball examining technique. To record messages on Twitter, a hashtag composed with a '#' image is used. A classifier has been used on Twitter to identify potentially counterfeit tweets. Supervised Machine Learning algorithms were applied to build the model. For example, the tweets were haphazardly picked and named by the scientist. Initially, 150 genuine messages were received. These messages were used to analyze the impact of individual voters on Twitter. For their analysis, the dataset was cleaned.

A further 150 tweets have been compiled from the corpus. These messages were being sent by bots or as being fake. Using the Camisani-Calzolari rule set [21], these messages were labeled as bogus. The greater the probability that the tweet was bogus, the more norms of this set of rules were deemed invalid. A dataset of Twitter messages about the 2012 Dutch election campaign was grouped. On these ordered tweets, a subjective substance inquiry was carried out. the analysis on 613,033 tweets were done. It was mentioned that the Decision tree classifier performed best in the detection of false and true messages. They have found a weighted F-score of 88% using decision trees algorithm. Another good performance was found by using linear support vector machine algorithms in which they got an F-score of 86% for distinguishing false messages from true ones.

Wynne and Wint [13] proposed a framework that recognizes fake online news stories. They utilized word n-grams and character n-grams in their investigation. Preprocessing was applied to the features and the news stories to reduce the data. The size of the n-gram can have a distinguishing description. An n-gram of a word or character is called unigram; two is bi-gram, three is trigram, and four-gram is where n=4. By analyzing the two tests, Gradient Boosting accomplished the highest precision of 96% when utilizing character trigram and four-gram, and TF-IDF at 10,000 highlights. The more significant part of the examination centers around word n-gram as an element while identifying fake news. Although character n-gram has been utilized in numerous grouping assignments, this is the first run by using character n-grams in identifying Fake news. The authors observe that character n-gram are better than word n-grams utilizing TF and TF-IDF as highlight extraction methods. Curiously, they have noticed that the length of the word n-grams expanded while the classifiers' precision diminished.

Reis et al. [14] have studied fake news detection by employing various feature extraction techniques. They have used multiple features such as language features, lexical features, linguistic features, Psycholinguistic Features, semantic features, and subjectivity features. They have extracted language features from sentences. Using a bag of words (n-gram) and tagging them by finding the words' location in the sentences (categorize the words as a noun, adjective, verb), lexical features showed the number of times a unique word was used. In contrast, linguistic features showed the number of words (word counts). Semantic features gave toxicity scores to the news articles. Subjectivity features were sentiment scores, which categorize the features as positive, neutral, or negative.

Reis et al. [14] also have used other types of features derived from news sources. Bias, credibility and trustworthiness of the news and domain location are other aspects of the news. Bias features show the political bias of the news source. For credibility, they have used the top 500 news of Alexa papers. Then they have calculated the distance metric (dissimilarity) between each news domain with those of Alexa paper. The derived minimum edit distance metric was considered as a credibility measure. Domain location shows IP, latitude/longitude, city, and country of the news domain. The other two environmental features which were used are engagement and temporal pattern. They have taken 12 features from several likes, comments, and shares from Facebook users for attention. They considered them through the time from the beginning of the news release. The temporal pattern showed the rate of comments released from the users at the same time window. All these features were used to identify among genuine and fake news. They have shown by the chi-squared test that among all these features, credibility and location of the domain and engagement are the most discriminative features. For classification methods, they have used Naive Bayes (NB), Random Forest (RF), k-nearest neighbours (KNN),

Support Vector Machines (SVM) and XGboost (XGB). They have shown that among these classifiers for fake news detection dataset, Random forest and XGBoost method performs better compared to other methods with an accuracy of 85% and 86%, respectively.

Ahmed et al. [15] studied fake news detection using a dataset including 12.6K news articles. They used lower casing, punctuation and stop words removal and stemming for preprocessing. The feature extraction used N-gram features from N = 1 to 4, using Term frequency (TF) and term frequency-inverse document frequency features (TF-IDF. For the classification, they used stochastic gradient descent (SGD), linear support vector machine (LSVM), SVM, K-nearest neighbours (KNN) and decision trees (DT). Data was split into training and testing sets with 80% training data and 20% for test data with 5-fold cross-validation. The maximum accuracy of 92% was achieved using LSVM and the best features were unigram TF-IDF features.

Khan et al. [16] used three datasets to develop models for fake news detection. They used traditional and deep learning methods to analyze the datasets. They found that the developed model for counterfeit news detection is not unique for various datasets. In the preprocessing, spelling correction, stop words removal, stemming was done. In the feature extraction, lexical features, sentiment features and TF-IDF features were extracted from three datasets. Another feature that they have used was extracted using the Empath library. This library feature is developed as unsupervised learning, which can categorize the news articles' topic. The topics were organized into groups such as crime news, violence news, pride news, deception news, sympathy news, war news. As traditional classification methods they used support vector machines (SVM), k-nearest neighbours (KNN), logistic regression, decision trees and Naive Bayes. For deep learning methods, they used convolutional neural networks (CNN), long-term short-term memory networks (LSTM), Bi-LSTM, one convolutional layer with one LSTM (C-

LSTM), Hierarchical attention networks (HAN), convolutional hierarchical attention network and character level C-LSTM. Among traditional methods, Naive Bayes performed better than others with 95% accuracy using TF-IDF unigram features in a dataset with 80K news. In two other datasets, the Naive Bayes with TF-IDF bigram features performed better than others with 60% and 90% accuracy in datasets with 7.8K and 12.8K news, respectively. The decision tree had the lowest accuracy than other models (51%, 61% and 67% for datasets with size 7.8K, 12.8K and 80K, respectively). The deep learning methods using Bi-LSTM and C-LSTM had an accuracy of 95% for the 80K dataset, like Naive Bayes' results. For the dataset with a 7.8K news article, the accuracy of deep learning methods for Conv-HAN was 59% and close to Naive Bayes results for bigram features. The accuracy of the data with 12.8K was found to be a bit higher in deep learning; the value of 95% accuracy was found in character level C-LSTM, which is higher than traditional models' maximum accuracy, which was found to be 90% for Naive Bayes classifier. They claim that on a dataset with less than 100k news posts, Naive Bayes with n-gram can accomplish a comparable result to neural network-based models. The performance of LSTM-based models depends heavily on the length of the dataset and the details provided in a news report. They have mentioned that with enough information in the dataset, the models based on LSTM are more likely to resolve the overfitting problem. Also, advanced models such as C-LSTM, Conv-HAN and C-LSTM character levels have shown high promise in fake news detection that needs more attention to these models. Finally, they have conducted a topic-based study that reveals the challenge of correctly identifying deceptive news relevant to politics, health, and science [16].

Thota et al. [17] implemented fake news detection using deep learning algorithms. They outperform current model architectures by 2.5% and achieve a test result accuracy of 94.21%

using a precisely calibrated TF-IDF-Dense Neural Network (DNN) algorithm. Those models worked reasonably well when the headline positions and news articles are unrelated, approved and discussed. Still, the accuracy of the prediction for the disagreement position is low (44%). BoW-DNN model gave the second best performance. It is crucial to recognize that words such as Word2Vec have consistently provided low precision. They have observed that it could achieve smooth and productive language learning by applying regularization techniques such as Dropout, L2 normalization, preprocessing, and earlier ending. Ultimately, they planned to expand this analysis by performing a similar examination on an entirely different data set, such as Twitter and Facebook. By classifying false news from social networking sites, this study offers a benchmark for expanding the range of solutions for fake news detection. Data from the social network would ensure the discussion of language gaps. They would like to dig deeper to assess this news distribution's impact on readers and establish consistent strategies for faster prediction [17].

Vijayaraghavan et al. [18] have studied fake news detection. In preprocessing, they have removed stop words, punctuation, digits, special characters, and URL links. Then they have compared the distribution of polarity sentiment of the data before and after preprocessing. The texts were tagged to identify the position of the words. By drawing bar plots, they have shown that pronouns were used more in real news. In contrast, adverbs and adjectives were used more in fake news. In the feature extraction step, they have used word2vec embedding, word count vectorizer and TF-IDF vectorizer. They have considered features derived from unigram and bigram words in their word count vectorizer and TF-IDF vectorizer. Before implementing the classification methods, they have done outlier removal and fine-tuning to get proper tuning parameters for each classification model. In the classification analysis, Artificial Neural Network

(ANN) and long-term short memory networks (LSTM), a special case of recurrent neural networks (RNN), were used as deep learning methods. Other classifiers like support vector machines (SVM), random forest (RF), logistic regression (LR) was used. Among feature extraction methods, the count vectorizer performed best compared with two others. The word2vec embedding was the worst feature extraction method. Performing 3-fold cross-validation, they show that the count vectorizer features could get an accuracy of 94.88% in the long-term short memory model (LSTM). The highest accuracy found by word2vec embedding features was derived in ANN as 93.06%. For TF-IDF features, the maximum accuracy was found in logistic regression as 94.79%.

Jagadeesh and Mathias [19] analyzed the social media data of Twitter. They got the dataset using keywords "Las Vegas shooting" and "Hurricane Harvey that occurred in Houston". They took 250 related tweets from Twitter API. They labelled the tweets manually as fake and real news. From 250 tweets in these two events, 156 tweets were real news and 94 were fake news. They used text blob to do sentiment analysis to separate tweets with negative, neutral, and positive sentiments. They also used regexp and nltk libraries to get # of hashtags, # of question marks in each tweet, # of mentions, # of exclamation marks, # of URLs, polarity (negative, neutral, and positive), # of first-order pronoun, # of 2nd order pronouns and # of 3rd order pronouns. After getting these features, they removed stop words and punctuations and URLs to clean the dataset. Then, Word counts of tweets were derived with some personal information from the Twitter account. The tweets source was also categorized, whether from mobile, news channel, Facebook account, browser etc. After splitting the data into 80% as training and 20% testing, they used 10-fold cross-validation with Random Forest, logistic regression, and decision tree as classifiers.

The maximum accuracy of 74% was achieved with Random Forest, with 58% precision and 62.5% recall.

In this study machine learning model for fake news detection was developed using the dataset from kaggle.com [20] the same dataset which was used by (Agarwalla, et al 2019 [11]). The dataset includes labels for fake and real news. Two categories for fake and real news had close proportions which made the dataset balanced between two categories with 53% fake news and 47% as real news. The results were compared with the obtained results by (Agarwalla et al. 2019) [11]. The accuracy of the SVM model (98%) was much higher compared with the maximum accuracy derived by (Agrawalla et al. 2019) [11] using Naive Bayes classifier with Lidstone smoothing (83%). Since they used only a small number of features (around 100 features) and 1-gram, which considers each feature separately in the document. They also used a threshold of approximately 20% for each selected feature's maximum document frequency. In this research, analysis was performed without using limit on the number of features and using (1,2) gram (1-gram means each word separately, and 2-gram means two consecutive words were considered in the feature extraction) for the threshold of maximum document frequency. The machine learning analysis was also done on another dataset from kaggle.com [22] to check whether the improvement seen in the first dataset was due to overfitting or whether it is an appropriate approach for increasing the classification performance models in fake news detection. The algorithm was found to be useful in improving the model accuracy for fake news detection in the second dataset as well.

Chapter 3

Data and Processing

3.1 Overview of Methodology

In this section, an overview of the whole procedure of detecting fake news is described. Figure 3.1. shows the flow diagram for the steps in the implementation of the methodology.

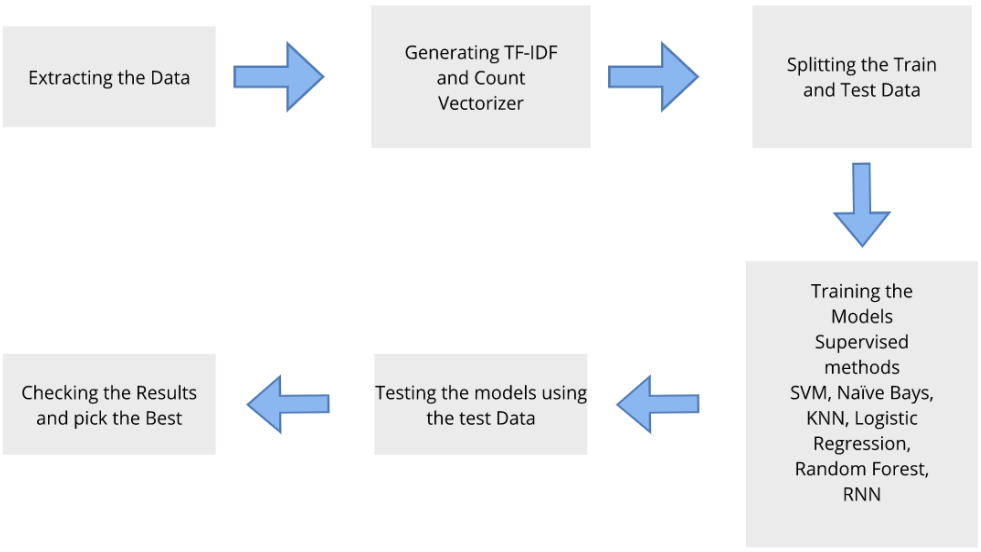


Figure 3.1. *Overview of the methodology*

As shown in Figure 3-1, in the first step the data is extracted for fake news detection from twitter. In the second step, TF-IDF and Count-vectorizer of the data is generated to extract the features from it. In the third step the data is split into 70% to train the model and 30% to test it. In step 4, the classifiers are applied on the test data to predict the fake news. In step 5, the results are analyzed for the best accuracy and the models that provide the best performance is selected.

3.2 Characteristics of the dataset

In this section, the datasets used for fake news analysis are described and the methods and steps of pre-processing the data are described.

In this study, two different types of datasets were used to check the accuracy by training the models and setting the parameters.

3.2.1 First Dataset for fake-news detection

The dataset contains four columns of URL, Headline news, Body of the news and the class label which show whether the article news is real, or it is fake news [20].

Table 3.1. *Fake news dataset (First Dataset)*

Column Name	Type
URLs	URL
Headline	String
Body	String
Label	Integer

Supervised machine learning algorithms are used to detect fake news since the dataset labels are known. The dataset for machine learning models is appropriate since the class labels are balanced among both groups of fake and real news. The proportion of label categories (real and fake) is not much different from each other, which does not bias a specific class due to different prior probability for one class label. The frequency of each group of the label is shown in Figure 3.2.

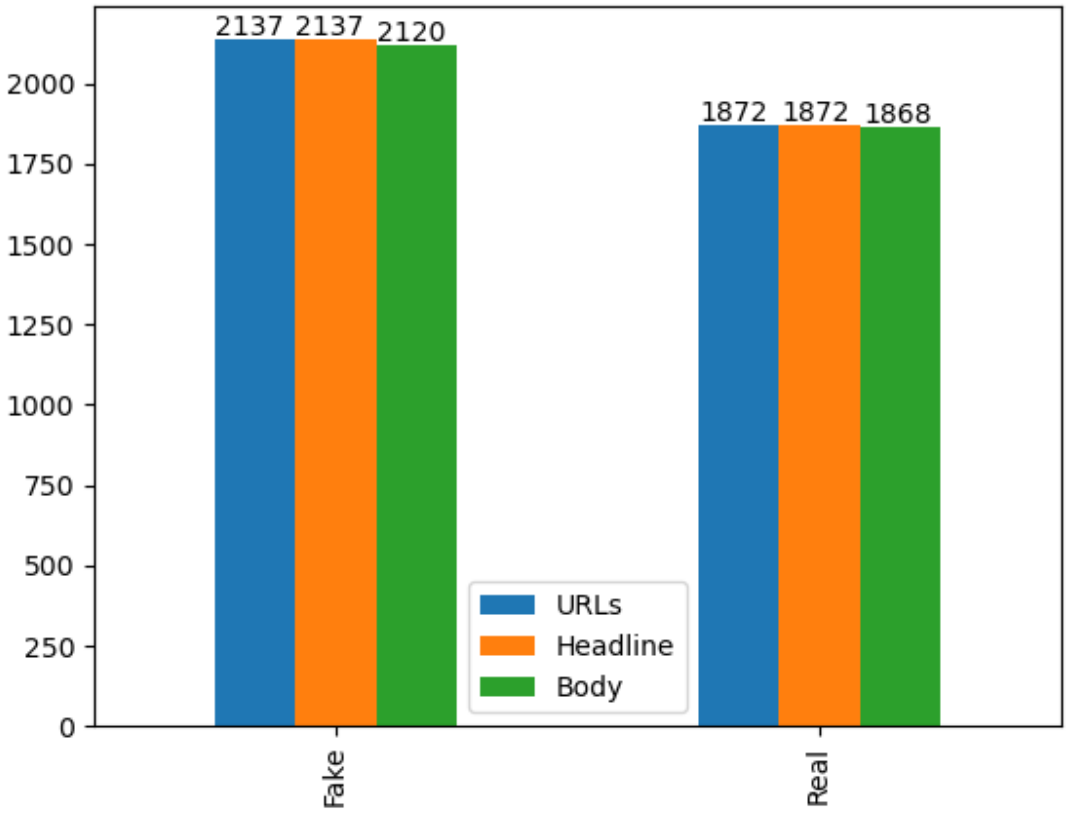


Figure 3.2. *Frequency of fake and real news in the dataset.*

The dataset includes 4009 article news. It could be seen that 2137 (53.3%) news are fake news, and 1872 (46.7%) are real news. The URL and Headline of the news are complete for all 4009 articles. But for Body, there are 17 actual news articles and four fake news articles with no or missing Body. There are 12 unique hostnames in this dataset. These news articles are taken from these 12 news websites: “abcnews”, "before its news", "bleacher report”, "clarivate", "dailybuzzlive", “activist post”, “BBC”, “CNN”, “disclose Tv”, "NYTimes" and "Reuters." There are two articles from api.content-ad.net. The cross-tabulation Table 3-2 for the labels versus these hostnames shows a clear dependence between the hostname and the news type. On

the websites that have real news, all the news is real, and on those that have fake news, all the news is fake in this dataset.

Table 3.2. *Contingency table for news hostname versus label*

Host Name	**Label**	
	Fake news	**Real News**
abcnews.go.com	0	57
beforeitsnews.com	1694	0
bleacherreport.com	0	9
clarivate.com	0	3
dailybuzzlive.com	99	0
www.activistpost.com	271	0
www.bbc.com	0	343
www.cnn.com	0	469
www.disclose.tv	71	0
www.nytimes.com	0	344
www.reuters.com	0	647
api.content-ad.net	2	0

Collection of this data is done somehow, that all the news that was taken, for example from reuters.com, are real news and all the news taken from beforeitsnews.com are fake news. It

seems that data collection was not done randomized. All the fake news was taken from some URLs for collecting the data, and the real news was taken from other URLs. So, the URL column is not informative in this data. Using URL in the analysis will cause overfitting in the training data by increasing the bias, but the prediction on new unseen data will be different, and this is due to bias added in the data collection. Hence in this case study, the column of URL is not used in the analysis.

The other two columns of this dataset are Headline and Body. To separate the sentences of the Headline and Body of the article news, natural language processing is used. The NLTK library includes a punkt statement tokenizer, a pre-trained unsupervised learning model that can tokenize and identify the sentences in the statement, the punctuation, and the position of the words in the sentences. The pre-trained model was trained on English corpus and it can distinguish between the punctuation of each sentence and find the position of the words in the statement.

By using a punkt statement tokenizer on the Body of the articles, the number of sentences in each news article can be counted. The number of words in the headline and Body, the number of nouns, verbs and adjectives used in each news article were counted and are presented in Table 3.2. Table 3.3. shows the information about these parameters and the difference between each class label.

Table 3.3. *Descriptive statistics for some properties of the dataset*

Variable	descriptive statistics	Fake news N = 2120 (53.2%)	Real News N = 1868 (46.8%)

word count headline	Mean	10.36	9.14
	Standard error	0.1031	0.0521
	95% conf int - LB	10.1602	9.0381
	95% conf int - UB	10.5644	9.2424
word count body	Mean	397.72	592.34
	Standard error	10.71	11.99
	95% conf int - LB	376.72	568.81
	95% conf int - UB	418.73	615.87
Sentence count body	Mean	19.22	29.69
	Standard error	0.49	0.68
	95% conf int - LB	18.25	28.36
	95% conf int - UB	20.18	31.03
# of Stopwords	Mean	158.14	228.59
	Standard error	4.35	4.90
	95% conf int - LB	149.60	218.98
	95% conf int - UB	166.67	238.20

# of non-alphabetic	Mean	15.24	31.20
	Standard error	0.53	0.84
	95% conf int - LB	14.19	29.54
	95% conf int - UB	16.29	32.87
# of Nouns	Mean	134.67	202.03
	Standard error	3.53	3.98
	95% conf int - LB	127.74	194.21
	95% conf int - UB	141.60	209.85
# of Verbs	Mean	62.70	98.80
	Standard error	1.68	2.08
	95% conf int - LB	59.41	94.71
	95% conf int - UB	66.00	102.90
# of Adjectives	Mean	187.50	283.56
	Standard error	5.28	6.00
	95% conf int - LB	177.13	271.77
	95% conf int - UB	197.87	295.34

# of URLs link	Mean	0.4943	0.4936
	Standard error	0.0239	0.0288
	95% conf int - LB	0.4476	0.4370
	95% conf int - UB	0.5411	0.5501

LB and UB are lower bound and upper bound of 95% confidence interval.

Stopwords are words that are used more frequently in all statements. These words occur very frequently in the statement and do add much information about the text type. As they are commonly used words in everyday speech and writing, these words are removed in most analyses due to contributing no informative knowledge about the target class. In this study, these words were removed from the Headline and Body of the text in the preprocessing stage of the analysis.

URLs, digits, and string punctuations also do not add information about the target and these strings are considered redundant and they are removed in most of the text classification analysis. URLs, digits, and string punctuations are also removed from the dataset in the preprocessing stage.

Before removing the URLs, stopwords and string punctuations, regular expressions were used to match and find the expressions for the number of URL links presented as reference of the news, the number of stopwords used in the article and the number of non-alphabetic words. The mean value of the number of word count in headlines is lower for real news than fake news. The 95% confidence interval of the mean value for fake news [10.16,10.56] is quite different from the 95% confidence interval of the real news [9.03,9.24]. This shows significantly less word count in

fake news compared with real news. But in the body of the article, it is reversed. The number of word count in fake news [376.72, 418.73] is much less for fake news than real news [568.8, 615.87] in the body of the article news.

Sentences in the body of the article are also less in the fake news compared with real news. The mean value of the number of sentences in the fake news is 19.22, while in real news, the mean value is 29.69. The 95% confidence interval shows distinct ranges of [18.25, 20.18] for fake news and [28.36, 31.03] for real news, which shows that the difference is statistically significant. For the number of nouns, verbs, adjectives in the body of the article, this is also seen that these values are more in the real news compared with fake news. The 95% confidence interval implies that these differences are statistically significant. The number of URL links used in the article's body for fake and real news is not different from each other since they have close mean value, and their confidence interval is not distinct from each other.

3.2.2 Second Dataset for fake-news detection

This dataset was taken from kaggle.com [22]. The columns in this dataset are the title (headline), author of the news, text (body) and the label, which specify whether it is fake news (1) or real news (0). Some of the news has a missing title, author name or body. Table 3.4. shows the columns and their data types.

Table 3.4. *Second Dataset of fake news (Train.csv)*

Column Name	Data Type
Id	Integer
Title	String
Author	String
Text	String
Label	Integer

This data includes 20,800 news articles. From 20,800 articles, 10,387 (49.9%) of them are fake news and 10,413 (50.1%) are real news. The dataset is balanced between two categories of real and fake news. The frequency plot for author name, title and body for fake and real news is shown in Figure 3.3.

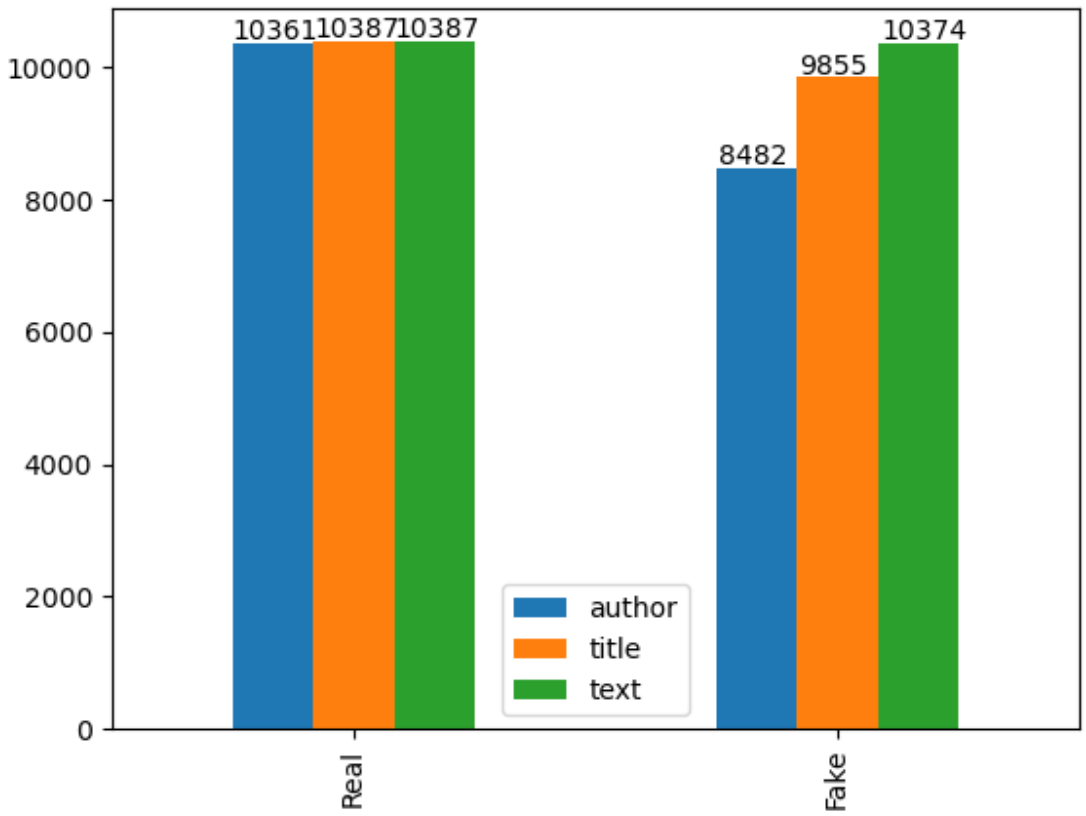


Figure 3.3. *Frequency of non-missing author name, title, and text in the dataset for fake and real news*

For author names there are 1957 missing author names (1931 missing in fake news and 26 missing in real news), no missing item for title and text of real news and there are 558 missing titles for fake news and 39 missing in text of the fake news. These 558 + 39 = 597 news were removed from the dataset. After removing them there are 20,203 news where 10,387 (51.4%) of

them are real news and 9,816 (48.6%) of them are fake news. The dataset is still balanced and could be used for classification analysis.

The title of text of the news is used in the analysis. To compare the general characteristics of title and text among fake and real news, some general aspects of the news article are considered. For example, the number of words in the headline and body of the news, number of sentences in the body, number of stopwords, punctuations and links used in the news article. Table 3.5. shows the descriptive statistics for these general aspects separately for fake and real news.

Table 3.5. *Descriptive statistics for some properties of the dataset*

Variable	Descriptive Statistics	Real news N = 10387 (51.4%)	Fake News N = 9816 (48.6%)
Word Count Headline	Mean	13.66	11.08
	Standard error	0.0286	0.0474
	95% conf int - LB	13.609	10.992
	95% conf int - UB	13.722	11.178
Word Count Body	Mean	878.99	674.29
	Standard Error	7.25	9.93
	95% conf int - LB	864.77	654.81
	95% conf int - UB	893.21	693.78
Sentence Count Body	Mean	50.85	30.79

	Standard error	0.45	0.53
	95% conf int - LB	49.95	29.73
	95% conf int - UB	51.75	31.84
# of Stopwords	Mean	357.63	270.13
	Standard error	3.08	4.28
	95% conf int - LB	351.59	261.73
	95% conf int - UB	363.68	278.54
# of non-alphabetic	Mean	16.08	15.37
	Standard error	0.16	0.28
	95% conf int - LB	15.77	14.82
	95% conf int - UB	16.40	15.93
# of Nouns	Mean	282.88	220.65
	Standard error	2.17	3.42
	95% conf int - LB	278.62	213.94
	95% conf int - UB	287.13	227.36
# of Verbs	Mean	146.60	107.75
	Standard error	1.26	1.65
	95% conf int - LB	144.13	104.50
	95% conf int - UB	149.08	111.00

# of Adjectives	Mean	412.09	325.39
	Standard error	3.54	4.95
	95% conf int - LB	405.14	315.69
	95% conf int - UB	419.04	335.10
# of URLs link	Mean	0.1093	0.0911
	Standard error	0.0121	0.0075
	95% conf int - LB	0.0855	0.0764
	95% conf int - UB	0.1331	0.1058
Average # of Words in sentence	Mean	22.22	28.16
	Standard error	0.062	0.151
	95% conf int - LB	22.09	27.87
	95% conf int - UB	22.34	28.46

LB and UB are lower bound and upper bound of 95% confidence interval.

It can be seen in Table 3.5. that there is no significant difference between # of punctuations and # of URL links in the fake and real news. But word counts, sentence counts, number of stopwords, nouns, verbs, adjectives, and words in a body sentence are different in two types of fake and real news. The number of sentences is less in fake news of this data. It could be seen that the average number of words in sentences of the body is more in fake news compared with real news.

3.3 Preprocessing

Preprocessing is the most crucial step in machine learning. Preprocessing information is an information mining technique that involves changes to an intelligible configuration over crude

detail. Certifiable material is frequently incomplete, temperamental, or otherwise absent. Such propensities or examples are likely to include a few errors. Knowledge pre-processing is a way of addressing these problems. Whatever data we get from Twitter are unfinished, inexact, or it might have some errors, like missing values, null values etc. Before we perform any task in NLP, we must preprocess the data or clean the data to increase the data's quality and make it meaningful and readable. After we process the data, the data's size would be decreased so we can handle it very accurately [23]. In this research, python and its libraries were used to perform preprocessing on the data. Preprocessing will cause all the digits, punctuations, stopwords, URLs to be removed from the news article. Preprocessing includes the following steps:

3.3.1 Removing the stopwords

While working with text analysis, it should be made sure that the data are filtered out. As observed previously, the news or tweets consists of some useless words that occur frequently in the data but are unnecessary which are called **Stop_Words.** We can easily eliminate the stop words using the NLTK library in python, which has already stored Stop_Words in 16 different languages [24]. Stop word removal is enabled by NLTK and a list of stop words can be found in the corpus module. NLTK list of stopwords can be used to split the text into words and erase stop words from a sentence.

Stopwords are words which are used more frequently in all statements, like: 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at',

'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't". These words occur very much in the statement and do not add much information about the text type because they are commonly used words in normal speech and writing. These words are removed in most analyses as they add no informative knowledge about the target class. In this study the stop words were removed from the Headline and Body of the text in the preprocessing stage of the analysis.

3.3.2 Removing the punctuations and digits

Digits and string punctuations do not add information about the target class and these strings are considered redundant. They are removed in most of the text classification analysis. Digits and string punctuations are also removed from the analysis in the preprocessing stage. The following characters are considered as string punctuations:

'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

3.3.3 Removing URLs from the news

URLs are removed from the dataset to improve the data readability and to make it meaningful. In python there is a “re” package which is “**Regular Expression**” and is used to get the data into one variable and is applied to check whether it contains the hyperlinks or not. If it has some

hyperlink, it will remove it and make it a simple text so it will be easy to process that text as well as it will reduce the size of the dataset. The syntax for removing URLs using “re.sub” is shown below.

```
word = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\

'(? %[0-9a-fA-F] [0-9a-fA-F]))+',", word)

word = re.sub (r'http\S+',", word)
```

3.3.4 Using punkt statement tokenizer to separate the sentences in the news

To realize and separate the sentences of the article news, natural language processing is used. The NLTK library includes punkt statement tokenizer, a pre-trained unsupervised learning model which can tokenize and identify the sentences in the statement, the punctuation, and the position of the words in the sentences. The pre-trained model was trained on English corpus and it can distinguish between the punctuation of each sentence and find the position of the words in the statement.

3.3.5 Using word tokenizer to find the words in the sentences

Term tokenization is the way to split up a vast number of texts into phrases. In the processing of natural languages, this is a prerequisite for each word, such as classifying and counting for feelings etc., to be collected and further learned. The Natural Language Tool Kit (NLTK) [24] is a library used for this purpose. NLTK [24] is installed before using the Python tokenizer program.

Tokenization divides an entire dataset into individual words and each word is called as tokens. Word Tokenization is a common task when working with text. For example, if to tokenize one sentence [‘This is a report’], after tokenization it will divide the whole sentence into tokens

[‘this’, ‘is’, ‘a’, ‘report’]. Before processing NLP, the words must be identified that are Strings of characters as the meaning of the words [24] can be easily determined. Word tokenizing was done by word tokenizing () command in NLTK library.

3.3.6 Specifying the position of the words in the sentences

By using punkt statement tokenizer on the news articles and using the pos_tag command in NLTK library, the position of the words in the sentences can be specified, whether it is a noun, verb, or adjective.

3.3.7 Making the words characters as lower case

In the feature extraction, the program does not realize the words with lowercase and uppercase as the same word. So, they are considered as two different terms in the feature extraction. To avoid this, the words are converted into lowercase. This is done in python by using: word.lower().

3.3.8 Lemmatizing the words based on the position of the word in the sentences

This technique is called **Word Normalization** in the Natural Language Processing and is used to convert a word into its base form.

“In grammar, inflection is the alteration of a word to represent various grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. It expresses one or more grammatical categories with a prefix, suffix or infix, or another internal alteration such as a vowel change" [25]. Figure 3.4. is the example of how each word has its root form?

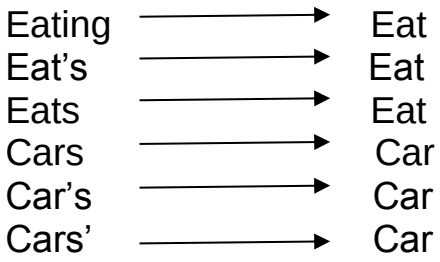


Figure 3.4. *Normalization of text*

3.3.9 Tagging the words and concatenate them together for each news article

After specifying the position of the words and doing lemmatization, the position of the word will be concatenated with the lemmatized word so that in the feature extraction there will be difference between a term when it is used as a verb or when it is used as a noun.

Chapter 4

Feature Extraction and Classification

4.1 Feature Extraction Methods

After cleaning the data, it should be mapped into the numeric presentation in the form of vectors. It is a part of the reduction process of dimensionality. A large dataset contains many variables. Feature extraction helps to get the best features from big datasets to increase the accuracy of the model by extracting features. Using Feature Extraction, words can be counted and the importance of the words in the dataset can be determined, which can help to reduce the redundant data from the dataset. Feature Extraction helps to minimize the number of features in a dataset by generating (and then discarding the original features) new features from the current ones. Much of the details found in the original set of features should then be represented by this new reduced set of features. In this way, from a combination of the original dataset, a summarized version of the original features can be generated. In this research, three types of feature extraction techniques were used, Count vectorizer, TF-IDF vectorizer and Word2vector Embedding, which are described in the next section.

4.1.1 Count Vectorizer

Count Vectorizer is a very effective tool given by the scikit-learn library in python to transform the text into a vector-based on each word's count in the whole text [26]. The count of the terms in the whole document will be considered as the token count for each term. It is also called bag-of-

words. The frequency count of occurrence of each term in the document is considered as a feature. These words could be considered as 1-gram (each token is considered separately) or n-gram (n consecutive token will be considered as one term). An example of bag-of-words, considering two documents is presented below:

1- "she got her license as a medical technician in wildfire fighting”

2- "Firefighters are fighting to contain California-wildfire."

1-gram and 2-gram bag-of-word after removing punctuations and stopwords will be as below:

1_gram_BoW = {“got”: 1, “license”: 1, “medical”: 1, technician: 1, “wildfire”: 2,

“fighting”: 2, “Firefighters”: 1, “contain”: 1, California: 1}

2_gram_Bow = {“got license”: 1, “license medical”: 1, “medical technician”: 1, “technician wildlife”: 1, “wildfire-fighting”: 1, “Firefighters fighting”: 1, “fighting contain”: 1, “contain California”: 1, “California wildfire”:1}

"wildfire" and "fighting" are seen with frequency 2 in 1_gram, while using 2-gram, the frequency will be 1 for them.

Some thresholds could be used as a minimum and maximum frequency counts to avoid features that occur very much in the whole document.

4.1.2 TF-IDF Vectorizer

Term frequency (TF) – Inverse Document Frequency (IDF) is an algorithm that will convert the text into an meaningful representation of numbers. TF-IDF is an abbreviation for Inverse Document Frequency of Word Frequency. It is a ubiquitous algorithm for translating the text into a meaningful number representation that is used to suit the prediction machine learning algorithm. It can be assumed that a higher number of reputations of the word has more weight in the data. Word occurrence within a document is called term- frequency, and word occurrence in

the document in the study are document frequency. There are some words with high occurrence in the document, but it gives less meaning of the word, such as 'a', 'the', etc. to reduce those words, we can use inverse document frequency. Formula and examples of TF-IDF are given below.

$$\text{TF-IDF}(t,d,D) = tf(t,d) \cdot idf(t,D)$$

where t is the term (a word for which the TF-IDF is calculated), d is the given document that the word inside it is evaluated, and D is the whole document in the study. Hence TF-IDF for term t is high when the word is used many times in each document while it does not occur many times inside all the documents which are at hand.

For both term frequency and inverse document frequency, there are several representations. Considering the raw counts of a specific word as the term frequency in a document, depending on whether the document length varies much from one document to another. The term frequency for each document could be used as its raw value [26]. Also, it could be scaled with the length of the document. So, the raw counts of the term frequency will be divided by the number of words in that document [27]. Another way to scaling is the logarithmic scale, in which they will be used as the term frequency in the calculation. The number 1 is added inside the logarithm in the case that a term has zero frequency in a document, the log of zero will be infinite. To avoid having infinite value, 1 (one) was added to the term frequency, and then it was scaled to a logarithmic scale. The inverse document frequency also has several representations. The non-smooth IDF is calculated by:

$$\text{idf} = \log(n / d(t)) + 1$$

where n is the total number of the documents in the study and d(t) is the document frequency of term t (term t was seen in how many documents inside the whole document sets). Number 1 was

added to IDF because when a term t is repeated in all the documents then n/d(t) will be equal to 1. The logarithm of 1 is zero. So, the effect of that term will be zero. To avoid this 1 was added to the formula of IDF [26]. The above formula is non-smooth because there is still a chance of having denominator as zero (d(t) = 0)). To avoid division by zero the smooth IDF could be used which adds 1 to numerator and denominator of the IDF formula:

$$\text{idf} = \log((n + 1) / (d(t) + 1)) + 1$$

This formula is the same as the previous one and just 1 was added to numerator and the denominator [26].

4.1.3 Word2vec Embeddings

Word to vector embedding was developed by this idea that each word has a relationship with the sequence of words occurring with that word in the sentences. word2vec embedding includes a cloud of words for each unique word in the model. In that cloud of words, the similarity between the words used with that unique word are added to it. For instance, cosine similarity could be calculated between each unique word and the words occurring with it. Hence it could be said that each unique word includes an embedded vector with it which shows the similarity between that unique word with its consecutive occurring words. word2vec algorithm uses a two-layer neural network model to fit and learn the relationship between the words with each unique word in the document. based on word2vec embedding simple operations like addition and subtraction can be done on the tokens. The operations on expressions are done by Mikolov et al, (2013) using bitwise (AND) operator. Each word has an embedding vector which includes the most similar words to it. Then the adding operation is done by checking the most similar tokens which are in both embedding words. For example, adding two words of “Czech” + “currency” is done by

getting the embedding words which are in both subsets of “Czech” and “currency”. Four closets tokens to both of these expressions are {koruna, Check crown, Polish zolty and CTK} [28].

Table 4.1. *Operations on Tokens wor2vec embedding [28]*

Expression	Four Nearest tokens
Czech + currency	1 - koruna 2- Check crown 3- Polish zolty 4- CTK
Vietnam + capital	1- Hanoi 2- Ho Chi Minh City 3- Viet Nam 4- Vietnamese
German + airlines	1- airline Lufthansa 2- carrier Lufthansa 3- flag carrier Lufthansa 4- Lufthansa
Russian + river	1- Moscow 2- Volga River 3- upriver 4- Russia
French + actress	1- Juliette Binoche 2- Vanessa Paradis 3- Charlotte Gainsbourg 4- Cecile De

4.2 Classification Methods

The following classification methods were applied for the detection of fake news: Support Vector Machine (SVM), Logistic Regression (LR), Naïve Bayes (NB), Decision Trees, Random Forest, K-nearest Neighbours (KNN), and Recurrent Neural Networks (RNN). These methods are discussed briefly.

4.2.1 Support Vector Machines (SVM)

SVM is a classification technique that is simple, reliable, and effective. It is easily adaptable and finds broad applicability in the computer vision industry. SVM is applied on clustered or classed data and has proved to be excellent in its performance for both regression and classification. It has also proved helpful in machine learning models for smaller but complex datasets but provides optimal results.

SVMs consider each data item as a point in n-dimensional space with feature values as coordinates. The classification is performed by extracting a hyperplane that greatly differentiates between the two different classes. Here, the support vectors are the coordinates of the data points. For linearly separable datasets, straight-line approximations are used based on some **ω** and **b** to segregate the two hyper planes while classifying. Figure 4.1. shows the strategies in which optimal hyperplane could be extracted for the data [29]. The loss function approximation for SVMs is given as follows.

$$C(x, y, f(x)) = \begin{cases} 0, y * f(x) \geq 1 \\ 1 - y * \dot{f}(x) elsewhere \end{cases}$$

$$min_w \lambda \|w\|^2 + \sum \sum_{i=1}^{n} (1 - y_i \langle x_i | w \rangle)$$

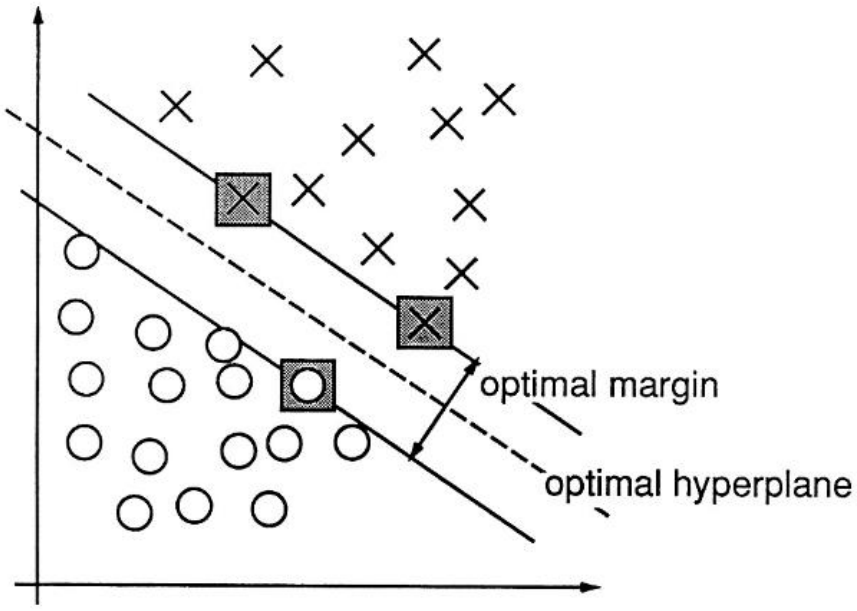


Figure 4.1. *Optimal Hyper plane in SVM [29]*

4.2.2 Logistic Regression

Regression analysis refers to finding the relationship between one dependent variable say ‘x’ and a bunch of independent variables say ‘y’. In machine learning, we take a training set and make a model that predicts ‘y’ given the ‘x’. Various regression strategies could be applied to any process, depending on the application. These strategies include linear regression, logistic regression, and polynomial regression. The outcome in logistic regression is a binary response with 0 as first class and 1 as the second class. for instance, whether it is real news (1) or fake news (0).

4.2.2.1 Overfitting

An additional significant factor in choosing of the model is the model fit. The fluctuation calculation clarified in the log opportunities (generally stated as R2) is continuously extended by adding independent variables to the measured recurrence model. However, increasing numbers of variables in the model can promptly overfit, decreasing the model's generalization over the knowledge that matches the model [30].

The linear regression has a limitation of modelling the binary functions for linear decision boundaries, which is overcome using the logistic regression with 'k' number of filters. This could be observed through Figure 4.2. [31]. The logistic regression model yields the probabilities for the data classification and is an effective way to classify the data. It first makes 'k' latent processes, and that 'k' dimensional feature vector is then used to apply logistic regression, which ultimately yields the outcome. These 'k' features are sent through the sigmoid function, which maps it from 0 to 1 to introduce the probabilities. This adds modelling flexibility and allows considering non-linear decision boundaries in feature space. The equation for the logistic regression is as follows

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n$$

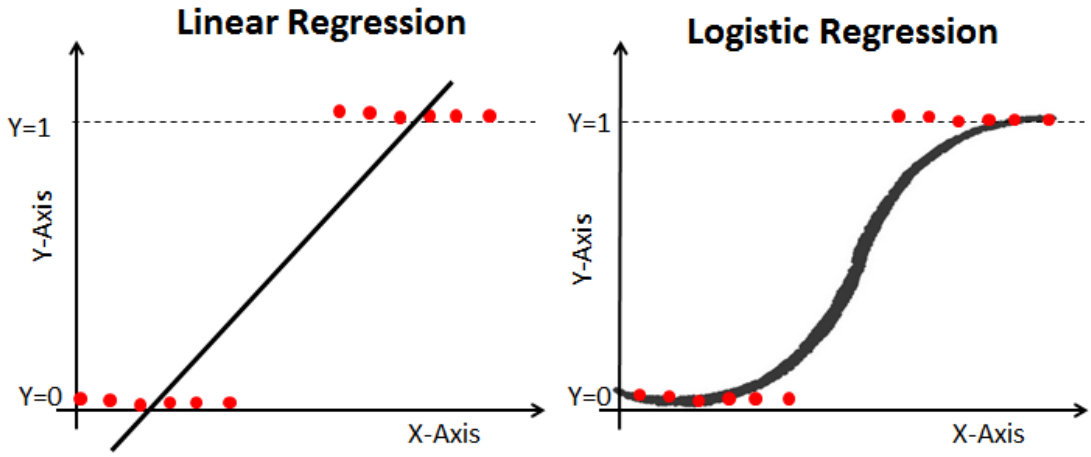


Figure 4.2. *Linear vs. Logistic Regression [31]*

4.2.3 Naïve Bayes (NB)

It is a classification scheme used in machine learning to differentiate between different objects based on the targeted features [32]. The strategy is quite fast. Naïve Bayesian classification is based on the simple Bayes theorem of probability which simply states that,

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

It simply tells that we can find the probability of event **A** happening given that event **B** has occurred. In case of multiple event occurrences, the probability becomes.

$$P(y|x_{1'}x_2, \ldots, x_n) = \frac{P(x_1|y) \cdot P(x_2|y) \cdots P(x_n|y)P(y)}{p_{(x_1)P(x_2)\cdots P(x_n)}}$$

The variable ‘y’ is considered as binary producing only 1 or a 0. For multi-class problems, the y outcome should have the maximum probability. If we are given the predictors, we can find the classes using the following equation,

$$y = \arg max_y \; P(y) \prod_{i=1}^{n} p(x_i|y)$$

Various applications of Naïve Bayes classification are sentimental analysis, spam filtering, recommendation systems, etc. It has a requirement of independent predictors, otherwise diminishes the performance of the classifier.

4.2.4 Decision Trees

Decision trees-based classification is a supervised learning scheme used in machine learning for classification and regression purposes. It arranges the data in tree formation and creates a model for predicting the target variables based on simple decision rules extracted from the data features [33].

This technique requires less data pre-processing. The more the depth is specified for the tree structure, the better the tree performs on predictions. Still, the structure becomes complicated due to many leaf nodes and dependencies upon one another. It has the capability of handling both numerical and categorical data for training and testing the model performance.

The strengths of decision tree methods:

1. Decision trees can make rules that are understandable.

2. Decision trees classify without requiring significant calculation.

3. Decision trees, as well as categorical variables, can handle continuous variables.

4. Decision trees provide a simple indicator to forecast or identify the most suitable areas.

The weaknesses of decision tree methods:

1. Decision trees are less suitable for estimation tasks where the aim is to estimate the value of a continuous attribute.
2. Multi-group decision trees and relatively small numbers of training examples are prone to errors in classification issues.
3. Decision tree preparation can be expensive in terms of computing. In terms of computation, the process of increasing a decision tree is expensive. At each node, each candidate splitting field must be sorted before it can find its best split.

Random Forest

Random forest is another supervised learning-based classifier. It is a technique that is based on many decision trees operating as an ensemble. The decision trees predict, and the class with the greatest votes becomes the final forecast result of the random forest model. Irregular forests or random forests are a grouping learning method to the group, relapse and various tasks that operate by creating many selected trees at the time of planning and creating the class that is the method of the individual groups of trees (arrangement) or mean/normal expectation (regression). Random choice backwoods are right for the propensity for choice trees.

The given example of a random forest in Figure 4.3. shows a scheme of random forest. In random forest after bootstrapping several decision trees, the results of the classification are derived by ensembling the predicted class from each decision tree.

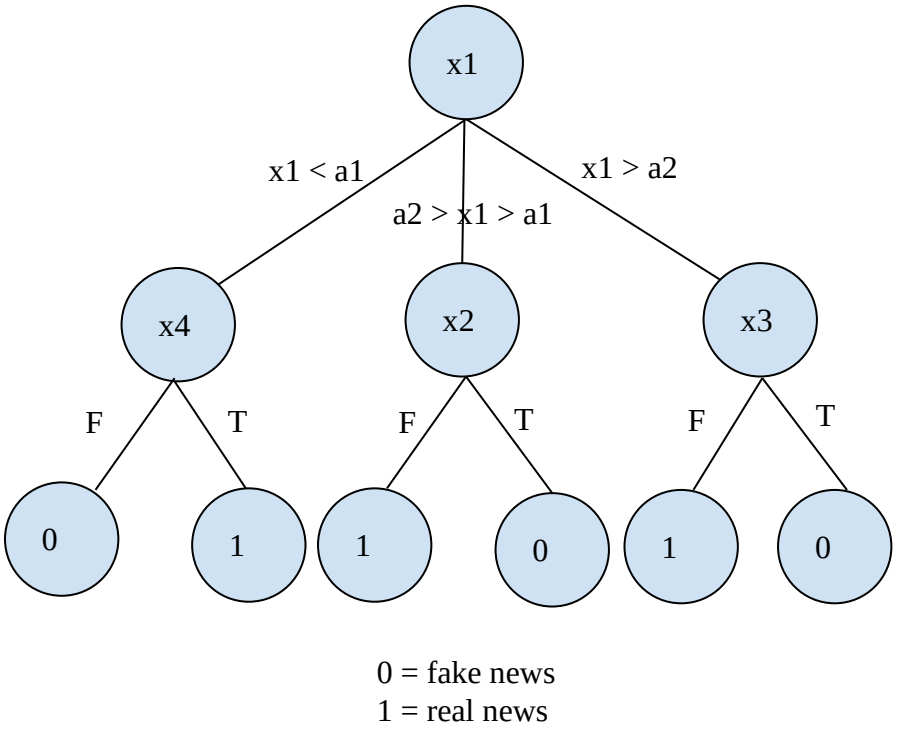


Figure 4.3. *Random Forest based Classification*

Advantages of Random Forest:

One of the most significant benefits of random forests is their resilience. It can be used for both regression and classification tasks, and the relative importance it assigns to the input features is also simple to view. Random forest is also a convenient algorithm because a good prediction result is always generated by the default hyperparameters it uses. It is effortless to grasp the hyperparameters, and there are not too many of them either.

Overfitting is one of the foremost difficulties in machine learning, but thanks to the random forest classifier, this will not happen most of the time. The classifier will not overfit the model if

there are enough trees in the forest. since in random forest the trees are drawn many times so random forest does not have the problem of overfitting.

Limitation of random forest:

The critical drawback of random forests is that for real-time predictions, many trees will make the algorithm too slow and ineffective. These algorithms are usually easy to learn, but they are very slow to make predictions once they are trained. More trees are needed for a more precise prediction, which results in a quieter model. The random forest algorithm is quick enough in most real-world applications, but there can certainly be situations where runtime efficiency is crucial, and other approaches would be preferred.

4.2.5 K-Nearest Neighbors (KNNs)

K-NN is a supervised learning algorithm that is commonly used for regression but is more predictive for classification purposes. It is simple and easy to implement. The algorithm operates on the data clusters where similar types of data are grouped together, and this assumption limits the performance of the algorithm [34].

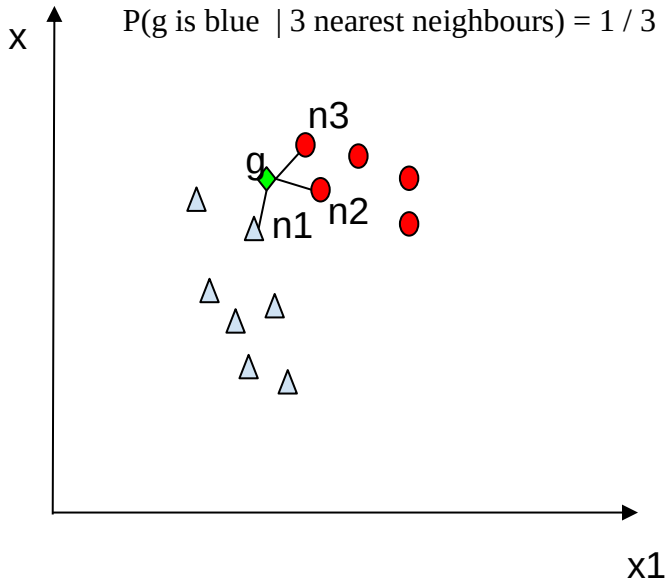
K-Nearest-Neighbors is a popular algorithm in the machine-learning classification arsenal. It is so widely used that most cluster models frequently start with KNNs first.

In simple recommendation boxes, creativity in picture recognition and dynamic models, KNN is commonly used. Calculators like Netflix and Amazon use it to prescribe different films or books for buying. The algorithm has the flexibility of choosing the number of neighbours one wants to involve for each prediction. The algorithm starts at a random point and calculates the Euclidean, Manhattan or Hamming distances between the random sample and the current sample point. All the spaces are calculated and then arranged. Out of these, the nearest 'K' entries are filtered out. It then finds the class of these K labels and assigns a label to the current sample. One example is

shown in Figure 4.4., where one new data point must look for the nearest neighbours and class for it is predicted using KNN. This way, the predicted sample gets the class of the nearest labels. KNNs find a lot of applications in recommender systems. In this example K = 3. hence for point g the Euclidean distance between point g and all the points were calculated. Then the 3 nearest neighbours were found as those which have the minimum Euclidean distance with point g. Then the probability of being in class red is calculated by the proportion of red class to whole the classes in 3 nearest neighbours. probability for class blue is calculated as the proportion of blue class to the whole the classes in the 3 nearest neighbours.

P(g is red | 3 nearest neighbours) = 2 / 3

P(g is blue | 3 nearest neighbours) = 1 / 3



Euclidean Distance (between point g and its 3 nearest neighbours) :
{ (x1_g - x1_n1)^2 + (x1_g - x1_n2)^2 + (x1_g -1_n3)^2 +
(x2_g - x2_n1)^2 + (x2_g - x2_n2)^2 + (x2_g - x2_n3)^2 } ^0.5

Figure 4.4. *K-Nearest Neighbors based classification*

4.2.6 Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) belongs to the class of deep learning in which a feedback path is devised such that previous outputs become the inputs in the presence of hidden states. An intermittent neural organization (RNN) is a type of bogus neural organization, where a directed diagram connects hubs around a moving arrangement structure. This allows for complex worldly events to be illustrated. This makes the RNNs ideal for businesses, for instance, unsegmented, similar penmanship recognition [35] or discourse recognition. The RNN's can use in-house (memory) to manage variable-long arrangements of information sources from feedforward neural networks. In time management, it is useful precisely because the variable also informs of past contributions. This is also called Long Short-Term Memory (LSTM). Intermittent neural organization of coevolutionary layers is also used to expand the feasible pixel spectrum. These networks have a memory element, which helps them to retain the outputs and feedback them for iterative prediction operation. The activation function for the RNN is as follows:

$$a^{\langle t \rangle} = g_I\left(w_{aa}a^{\langle t-1 \rangle} + w_{ax}x^{\langle t \rangle} + b_a\right)$$

Other activation functions that can be used in RNN are Sigmoid, ReLU, Tanh etc. and the output is expressed as,

$$y^{\langle t \rangle} = g_2\left(w_{ya}a^{\langle t \rangle} + b_y\right)$$

RNNs can work with any dimension of the input. They are dependent on the previous outcomes, but the modelling size does not increase with it. Due to the presence of a feedback mechanism, the computations are slow. They find complete applications in NLP and speech recognition. Other applications include sentimental classification, music generation, name entity recognition and machine translation. Due to its widespread applications in language processing, it is one of

the most relevant candidates in the fake news detection problem. Figure 4.5. presents a model of RNN.

Advantages of Recurrent Neural Network:

1. Each data is remembered by an RNN over time. It is useful only because of the feature to remember previous inputs in time series prediction as well. This is called Long Short-Term Memory.
2. To expand the efficient pixel neighbourhood, Recurrent Neural Networks are also used with convolutional layers.

Disadvantages of Recurrent Neural Network:

1. Gradient problems that disappear and blast.

2. It is a very challenging job to train an RNN.

3. In using tanh or relu as an activation function, it cannot process very long sequences.

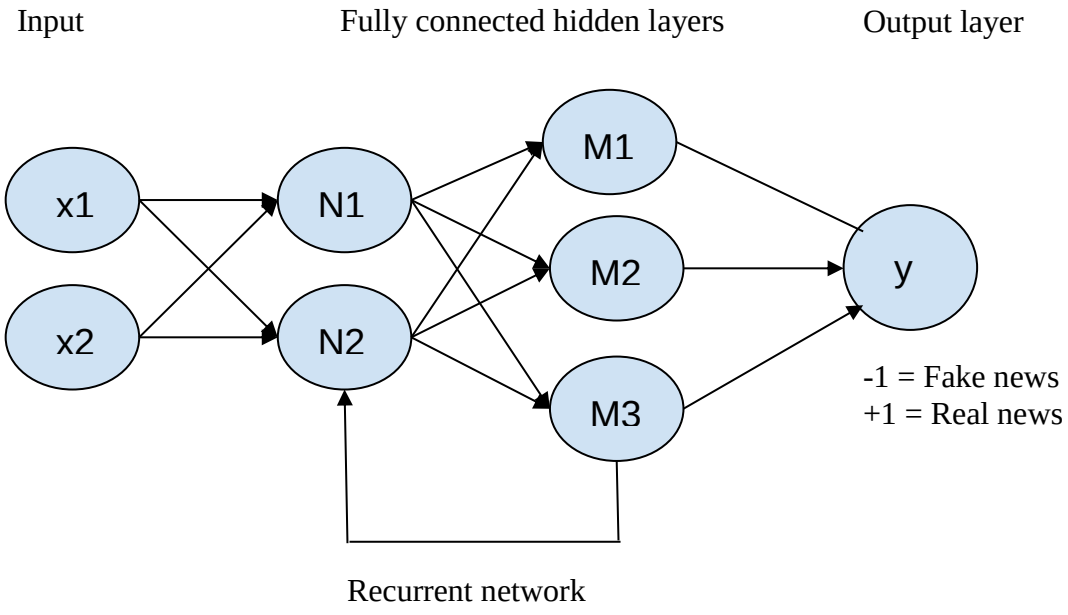


Figure 4.5. *Recurrent Neural Network (RNN) structure*

Chapter 5

Results and Discussion

In this chapter the classification analysis for fake news detection is discussed for the two datasets described in Chapter 3. To perform classification analysis, first preprocessing on the data is required. After preprocessing, stop words, punctuations, digits, and URLs are removed from the dataset. Although it seems that the stopwords, string punctuations, digits and URLs do not include useful information about the outcome of the classification but removing all of them in the preprocessing step and forgetting about them in the rest of the analysis could lead to missing some information about the outcome that may help in the prediction. Hence, before doing preprocessing to remove these items from the dataset, the influence of some of these characteristics will be considered on the class label.

5.1 Fake-news detection for the first dataset

5.1.1 Influence of general characteristics on class label

General parameters are considered as {word count in the headline, word count in the body, sentence count of the body, number of non-alphabetic tokens, number of URLs, number of stopwords, number of verbs in the body, number of nouns and number of adjectives}. To see the influence of these parameters one linear logistic regression model was fitted. Logistic regression model using these parameters and the class label as outcome was applied. Before implementing linear logistic regression, the data was split into two portions of 70% as training data and 30% as testing data. Results shows that the number of non-alphabetic words and number of URLs in the body are not statistically significant, with the (Z score = -0.076, P-value = 0.939) for non-

alphabetic words and (Z score = 0.707, P-value = 0.480) for number of URLs in the body. Both p values are much greater than 5% significance level and show that these parameters coefficient are not significantly different from zero. Removing these two parameters, the logistic regression model with the remaining parameters is given in Table 5.1.

Table 5.1. *Logistic regression results*

Parameters	Coef	std err	z	p > |z|	LB	UB
WordCountH	-0.0825	0.006	-13.683	0.000	-0.094	-0.071
WordCountB	-0.0441	0.004	-10.303	0.000	-0.052	-0.036
SentCount	0.0395	0.007	6.021	0.000	0.027	0.052
Stopwords	-0.0489	0.004	-13.624	0.000	-0.056	-0.042
Noun	0.0451	0.005	9.823	0.000	0.036	0.054
Verb	0.1029	0.006	16.728	0.000	0.091	0.115
Adj	0.0649	0.005	13.013	0.000	0.055	0.075

The likelihood ratio test shows that the model with these parameters is better than the simpler model with just an intercept log likelihood ratio (LLR) p-value = 2.88e-151 < 0.01 which rejects the null hypothesis of having a model with only intercept. McFadden's Pseudo R-Squared is 0.18 which shows that the model could improve the log-likelihood of the null model (model which contains only intercept) by 18%. The coefficients of all the parameters are significantly different from zero (P-value = 0.000 < 0.01).

Among the parameters, word counts in headline and number of stopwords have negative coefficients. It means that the odds for the news to be Fake news spreads by increasing the number of words in the headline. For example, by adding one word in the headline the odds of being fake news will be 1/exp (-0.0825) = 1.085. So, the news will be 8.5% more likely to be

fake news. Adding more words in the body depends whether the words are noun, verb, adjective or stopword will affect differently on the odds ratio. If the added word is a stopword, the odds ratio will be 1/exp (-0.0441-0.0489) =1.097. Adding one stopword will cause the article news to be 9.7% more likely to be fake news. If the added word in the body is a noun the difference will be 0.0451 - 0.0441 = 0.001 which is less than standard error (0.005), so the effect is not actually significant. So, no positive or negative effect on odds ratio could be seen by adding one noun word to the body. If the added word in the body is verb, hence the effect will be positive. The odds ratio will be exp (0.1029-0.0441) = 1.06, so the news is 6% more likely to be real news. If the added word in the body is adjective, then the odds ratio will be exp (0.0649-0.0441) = 1.02. So, the article is 2% more likely to be real news. Hence, the number of words negatively affects the news type in the headline, but for the body if the added word is adjective or verb then it has a positive effect and increases the log-odds (Log odds is logarithm of odds ratio. The logistic regression includes the logit link function so that the predictors are linearly related with the logit which is probability of being in class 1 (positive class) divided by probability of not being in class 1. This is called the odds ratio of being in class 1. so, each coefficient shows the log-odds of the class 1 and exponential of the coefficients will be the odds ratio for being in that class) of being real news. but if the added word is a noun then it has no significant effect on the log odds. If the added word is a stop word it has a negative effect, and the news will be more likely to be fake news.

Sentence count in the body has a positive effect on Real news. The log odds are 0.0395. The odds ratio for increasing 1 sentence in the news is exp (0.0395) = 1.04. which means adding 1 sentence will cause the news to be 4% more likely to be as real news.

It was seen that the number of sentences is effective in increasing the log-odds of being real news. Now a question comes up, which types of sentences have more effect on the log-odds, whether the longer or shorter sentences? Another parameter that could be included in the model is the average number of words in the sentences of the body article. The average number of words in each sentence was calculated by taking the results of the punkt statement tokenizer. The average of token lengths will be then the average number of words in all sentences in each article news. This variable was added to the existing model. It was seen that the log-odds of this parameter are positive and is significantly different from zero. The Akaike information criterion (AIC), Bayesian information criterion (BIC) which are for comparing the models both show that adding this parameter to the model improves the model. AIC and BIC before adding this parameter were 3158.21 and 3199.75, respectively. After adding this parameter, the AIC and BIC are 3139.28 and 3186.75, respectively. both are reduced compared with the model with less parameters. This shows that the model was improved by adding this parameter which counts words in the sentences of the body article. The pseudo-R squared (This is one minus the ratio between the Log likelihood of the full model to the log likelihood of the model with only intercept (constant) and it shows that how much improvement occurred in the model after using the predictors compared with the null model which include only intercept) also increases after adding this parameter to 0.19. The logistic regression model after adding this parameter is given in Table 5.2.

Table 5.2. *Logistic regression results - word in sentences added to the model*

Parameters	Coef	std err	z	p > |z|	LB	UB
WordCountH	-0.1296	0.012	-10.579	0.000	-0.154	-0.106
WordCountB	-0.0442	0.004	-10.236	0.000	-0.053	-0.036

SentCount	0.0495	0.007	7.124	0.000	0.036	0.063
Stopwords	-0.0489	0.004	-13.564	0.000	-0.056	-0.042
Noun	0.0445	0.005	9.586	0.000	0.035	0.054
Verb	0.1033	0.006	16.644	0.000	0.091	0.115
Adj	0.0642	0.005	12.788	0.000	0.054	0.074
Wordsinsent	0.0224	0.005	4.533	0.000	0.013	0.032

Other parameters log-odds are almost the same as the previous model. The log-odds of word count in the headline are less compared with the previous model. In the previous model it was -0.0825 while it decreased to -0.1296 in this model, showing an odds ratio of 1/ exp (-0.1296) = 1.1383 in this model. This means that adding 1 word in the headline increases the odds of being fake news by 13.8% while it was 8.5% in the previous model. The sentence count in the body had log-odds of 0.0395 in the previous model while it is 0.0495 in this model. which shows an odd ratio of exp (0.0495) = 1.05. The odds of being real news is 5% more by adding 1 sentence to the body of the article while it was 4% in the previous model. Words in sentences have positive log-odds (0.0224). This coefficient is significantly different from zero (p-value = 0.000 < 0.01). The odds-ratio of being real news is exp (0.0224) = 1.0226. it means in the sentence which has 1 more word the odds of being real news increases by 2%.

This model was tested using cross validation, the score of the classification on the testing data using 5-fold cross validation shows accuracy of 70.77% which is more than the best model achieved by (Agarwalla et al, 2019) [11] for penalized logistic regression on the features found from the text of these article news 66.57%. It means that in the paper released by (Agarwalla et al, 2019) due to removing many aspects of the document and keeping a low proportion of

features, the analysis shows low accuracy on the dataset for predicting the news type. While here without using the text itself and just by employing general aspects like number of words, number of sentences and words position, number of stops words the accuracy is 4.2% more compared with the one derived in the paper.

In this section it can be seen that some of the parameters which are going to be removed in the preprocessing step (stopwords) are not actually useless. The stopwords were seen to be helpful in the prediction of the class label. In the next step the analysis will be done without stopwords. But the influence of the number of stopwords on the class label could be added to the model by adding this variable to the model instead of keeping the stopwords in the analysis.

5.1.2 Feature extraction

After doing preprocessing on the data, the dataset is cleaned, and it is ready for feature extraction. In this section, features will be extracted by using the text words and tags of each news article. Two feature extraction methods were used in this study:

1- count vectorizer

2- Term frequency inverse document frequency vector (Tf-IDF)

5.1.2.1 Count vectorizer

Vectorizing the article news based on the frequency count of each word in the document was done, to avoid commonly used words which are used in most of the documents many times. To reproduce the results obtained by (Agarwalla et al, 2019), after removing the stopwords in preprocessing, a threshold was used to filter the words that occur in more than 20% of the documents. In total there are 3988 non-missing documents. So, the words which occur in more

than 797 documents were filtered. The results of classification were close to what obtained by (agarwalla et al, 2019) by keeping 100 features in the feature matrix. They have also used 1-Gram which considers the words separately in the feature matrix. As it was mentioned in the previous section the results obtained by them are low due to removing much information from the data. A simple investigation shows that around 82 news articles have zero frequency in all 100 features that exist in the analysis. Some descriptive aspect of count vector for top 12 features with t-test to test the difference between the count vector between fake and real news are presented in Table 5.3.

Table 5.3. *Statistics count vector for top 12 features in the dataset*

Term	min freq	max freq	sum freq	Mean in Real news	Mean in Fake news	t statistics	p-value
trump/n	0	28	2507	0.979	0.319	9.628	**0.000***
team/n	0	18	2094	0.475	0.568	-1.849	0.064
photo/n	0	72	2067	0.971	0.119	7.467	**0.000***
season/n	0	38	1950	0.258	0.692	-7.013	**0.000***
player/n	0	67	1724	0.487	0.383	1.594	0.110
company/n	0	21	1721	0.709	0.186	11.978	**0.000***
image/n	0	31	1629	0.751	0.106	12.268	**0.000***
home/n	0	25	1518	0.478	0.294	4.311	**0.000***
run/v	0	25	1485	0.315	0.422	-2.600	0.009
york/n	0	14	1430	0.542	0.196	10.997	**0.000***
woman/n	0	94	1430	0.603	0.142	6.526	**0.000***
group/n	0	40	1370	0.541	0.168	9.030	**0.000***

*p-value is less than 0.001, the null hypothesis of having difference of zero is rejected

Among 12 top features, the frequency of nouns “team” and “player” do not significantly differ between fake and real news. The p-values are 0.064 and 0.11 respectively, both have more than 5% significance level. In other features the difference is significant. In 8 of them the mean frequency of occurrence in real news is more than fake news. In the rest 2, the mean frequency in fake news is more compared with real news (noun “season” and verb” run”). For the word “season” the mean frequency in real news is 0.25 while the mean frequency for fake news is 0.69. For the verb “run” the p-value is 0.009 which is not very low, but it is less than 5%, so it is considered significant at 5% significance level, the mean frequency for “run” is 0.31 in real news and it is 0.42 for fake news. The features are sorted by the sum of frequency in the whole document.

5.1.2.2 TF-IDF vectorizer

This feature is term frequency divided by document frequency. This feature adjusts the term frequency by the number of documents the term was seen in. This feature is useful for excluding the terms which are more commonly used through the whole document. The same threshold of maximum occurrence in 20% of the whole document was used. In maximum 100 features were kept in the dataset. The table for the top 12 features having maximum sum of TF-IDF is presented in Table 5.4. with the t-test showing the difference between Real and Fake news.

Table 5.4. *Statistics TF-IDF vector for top 12 features in the dataset*

Term	min TF-IDF	max TF-IDF	sum TF-IDF	Mean in Real news	Mean in Fake news	t statistics	p-value
trump/n	0	1.0	351.24	0.0980	0.0792	2.593	0.009
video/n	0	1.0	319.06	0.0226	0.1305	-17.228	**0.000***
company/n	0	1.0	286.00	0.0912	0.0544	6.140	**0.000***
law/n	0	1.0	234.85	0.0373	0.0779	-6.753	**0.000***

team/n	0	1.0	228.93	0.0437	0.0694	-5.289	**0.000***
season/n	0	1.0	220.19	0.0329	0.0748	-8.182	**0.000***
reuters/n	0	1.0	206.85	0.1087	0.0017	26.747	**0.000***
system/n	0	1.0	202.72	0.0261	0.0725	-9.029	**0.000***
photo/n	0	1.0	194.15	0.0740	0.0263	10.377	**0.000***
great/a	0	1.0	193.32	0.0273	0.0671	-7.820	**0.000***
player/n	0	1.0	186.80	0.0449	0.0484	-0.739	0.459
york/n	0	1.0	181.12	0.0612	0.0314	7.325	**0.000***

*p-value is less than 0.001, the null hypothesis of having difference of zero is rejected

From top 12 features, only TF-IDF of noun "player" is not significantly different between fake and real news. In other features the difference is statistically significant. "trump", "company", "reuters", "photo" and "york" have higher mean TF-IDF value in real news compared with fake news. So, 5 features have higher average TF-IDF score in real news and the rest 6 features have higher average TF-IDF score in fake news. Features like "video", "law", "team", "season", "system" and "great" occurred more in the fake news articles.

5.1.3 Classification Results for the First dataset

The classification was done on the dataset using the extracted features. For each model TF-IDF features were used for training the models by using the headline, body and combination of headline and body. To compute the accuracy of the classification models, 5-fold cross validation has been used by splitting the data randomly 5 times into 70% as training data and 30% as testing data. The accuracy of the model is presented in the Table 5.5.

Table 5. 5. *Results of three classification models compared with (agarwalla et al, 2019)*

Feature set	Naive Bayes with lidstone smoothing		Support vector machine (SVM)		Logistic Regression	
	Current Study	(Agarwalla et al, 2019)	Current Study	(Agarwalla et al, 2019)	Current Study	(Agarwalla et al, 2019)
Headline + body	**95.99**	83.16	**98.50**	81.65	**98.25**	65.88
Body	**96.16**	82.53	**98.41**	81.65	**98.08**	65.88
Headline	**89.22**	68.05	**89.31**	66.24	**89.22**	66.57

The results of classification are much higher compared to the results obtained by (Agarwalla et al, 2019).

In the next step, using grid search for finding the best tuning parameters for each model, the results could be improved much by using more features. The number of features, threshold for document frequency, tuning parameters for each classification model was found by using pipeline and grid search through various parameter ranges. 5-fold cross validation was used to find the optimal model.

5.1.3.1 Grid search for Support Vector Machine (SVM)

Finding the optimal tuning parameter was done by using pipeline and grid search. In the feature extraction the maximum document frequency was used as ratios of 0.1, 0.2, 0.5, 0.75 and 1.0. These values specify a threshold for the features that they should not exist in more than 10% of the documents for 0.1. For 0.2, 0.5, 0.75 the threshold means that the document frequency should not be more than 20%, 50% or 75% respectively. The value of 1.0 lets the feature exist in all the documents. For the maximum features to be extracted five values of None, 50, 100, 150 and 200 have been used. None means that no threshold should be considered for the maximum number of features. Unigram and bigrams features were extracted in the grid search. The penalty of L2 and

elastic.net were used by alpha (L1 penalty is value of the absolute of the coefficients and L2 penalty is value of square of the coefficients and elastic.net is combination of L1 and L2 penalty) taking values of 1e-5, 0.1, 1/64. and 1/128. For normalization parameter T a value of 12 and None were considered in the grid search. After implementing the grid search with 5-fold cross validation for each grid, it was seen that among these parameters the maximum document frequency of 1.0 which do not set limits for document frequency, maximum feature as None which sets no limitation for maximum features, bigram which considers a sequence of two consecutive words. Penalty L2 with alpha equal 1e-05, normalization parameters of 12 were found to be optimal. The accuracy of 5-fold cross validation on combination of headline and body for the support vector machine is 0.98. This accuracy is 15% more than the accuracy found by (Agarwalla et al, 2019). This shows that using more features and considering bigram instead of unigram in the feature extraction will lead to much improvement in the classification model.

5.1.3.2 Grid search for Naive Bayes classifier

For Naive Bayes classifiers with lidstone smoothing parameter alpha, four values of (0.1, 0.2, 0.5, 1) have been used in the grid search. Other parameters for feature extraction like maximum document frequency, maximum number of features in the feature matrix were considered with the same ranges as the SVM model. The grid search results for Naive Bayes classifier show that smoothing parameter alpha = 0.1 is optimal. The maximum document frequency was found to be 0.2. No threshold for maximum features was considered. Bigram was found to be better than unigram. The 5-fold cross validation accuracy for Naive Bayes using the optimal parameters was found to be 0.963. The accuracy is much more than the one found by (Agarwalla et al, 2019). Although the maximum document frequency was found to be almost the same as what was used

by them, but the difference here is that no maximum feature is considered for the number of features and bigram was used instead of unigram.

5.1.3.3 Grid search for Logistic Regression

Logistic Regression was applied with the same parameters as in SVM. The optimal values were found to be L2 penalty with alpha = 1e-05. The maximum document frequency for Logistic Regression was found to be 0.75. No threshold for maximum features were considered. The bigram was found to be superior compared with unigram. The accuracy of Logistic Regression was found to be 0.986. It is much higher compared with (Agarwalla et al, 2019) which was 0.6588.

5.1.3.4 Grid search for Random Forest

For Random Forest, the grid search results show that unigram features are superior compared with bigrams in contrast to all other classifiers which worked better with bigrams features. The optimal maximum document frequency was found to be 0.5. No maximum depth was considered for each decision tree. The random forest model with 100 estimator trees, the minimum samples for splitting each node was 2 and the maximum features in the split was taken as the square root of the number of features in the model. Gini impurity was used as criteria for fitting the Random Forest model. Gini impurity is a measure of likelihood that a new random variable being incorrectly classified. The accuracy of the Random Forest model was found to be 0.969 using the optimal tuning parameters.

5.1.3.5 Grid search for K-nearest Neighbour

For k-nearest Neighbour in the grid search, value of k = 20 which considers 20 neighbours for each observation was found to be optimal among 5 values which were tested (5, 10, 15, 20, 25). Maximum document frequency of 0.5 was selected as the threshold. The bigram was found to be

superior compared with unigram. The k-nearest Neighbour classifier shows the accuracy of 0.939 with the 5-fold cross validation using optimal parameters.

5.1.3.6 Recurrent Neural Network (RNN)

Further analysis was done by implementing Recurrent neural networks (RNN). LSTM structure used include one embedding layer with Embedding dimension equal 256. The maximum number of words 65000 and maximum sequence length was set to 300. After tokenizing. sequences of words were created and padded somehow to have the same length using pad_sequences function from tensorflow preprocessing functions. The embedding layer is followed by two LSTM layers with filter size 256 and 128 respectively. Each LSTM layer include recurrent sequences and recurrent drop out equal 20%. between each layer a drop out of 25% has been used to avoid overfitting of the neural network model. after two LSTM layer. One fully connected hidden layer with filter size 32 has been used. The hidden layer includes rectifier activation function which sets the negative values to zero and keeps the positive values. The last layer or output layer is with size 2 (include 2 outcomes fake or real news) with sigmoid activation function. Sigmoid mapps the probability to 0 or 1 based on logit function. The LSTM model was running by using Adam optimizer. categorical cross entropy was used as loss function, the evaluation metric of the model is accuracy. The model was running by batch size = 64 and for 10 epochs. The analysis using this LSTM model has been used, the accuracy of the LSTM model for combination of headline + body is 0.98 % with loss equal 0.137.

5.1.3.7 Summary of classification

The optimal values found using grid search were entered for each model. The data was split randomly by 70:30, keeping 70% of the data in the model for training and setting 30% out for testing. All 5 classification models were trained, and the models were tested using testing data.

The accuracy of the testing data (precision, recall, F1 score and total accuracy) and confusion matrix are presented in the Table 5.6.

Table 5.6. *Performance of classifiers for the Body and Headline with AUC*

Classifier	headline + body	fake observed	real observed	precision	recall	F1 score	Accuracy	AUC
Naive Bayes	fake - n = 643	607	36	0.98	0.94	0.96	0.96	0.9941
	real - n = 554	12	542	0.94	0.98	0.96		
SVM	fake - n = 643	630	13	0.99	0.98	0.99	0.98	0.9988
	real - n = 554	5	549	0.98	0.99	0.98		
Logistic Regression	fake - n = 643	626	17	0.99	0.97	0.98	0.98	0.9978
	real - n = 554	4	550	0.97	0.99	0.98		
Random Forest	fake - n = 643	599	44	1.00	0.93	0.96	0.96	0.9977
	real - n = 554	2	552	0.93	1.00	0.96		
K-nearest Neighbour	fake - n = 643	591	52	0.95	0.92	0.93	0.93	0.9815
	real - n = 554	32	522	0.91	0.94	0.93		
LSTM	fake - n = 643	621	22	1.00	0.97	0.98	0.98	0.9987
	real - n = 554	1	553	0.96	1.00	0.98		

Among the classifiers, support vector machines (SVM) with predicting 630 out of 643 fake news correctly and 549 out of 554 real news correctly is with the highest accuracy, precision, and recall. Only 13 fake news articles were classified as real news wrongly and 5 real news were

predicted as fake news wrongly. The total accuracy is 98.5%. The ROC curve for each of this classifier is presented in Figure 5.1, Figure 5.2 and in Table 5.6. As we can see, the highest AUC is for SVM model with 0.9988 and LSTM with 0.9987. After that the AUC of Logistic regression is 0.9978, Random forest is 0.9977, Naïve Bayes is 0.9941 and K nearest neighbours is 0.9815. The AUC of all 6 classifiers is very close to 1.0 which shows all classifiers are performed very well.

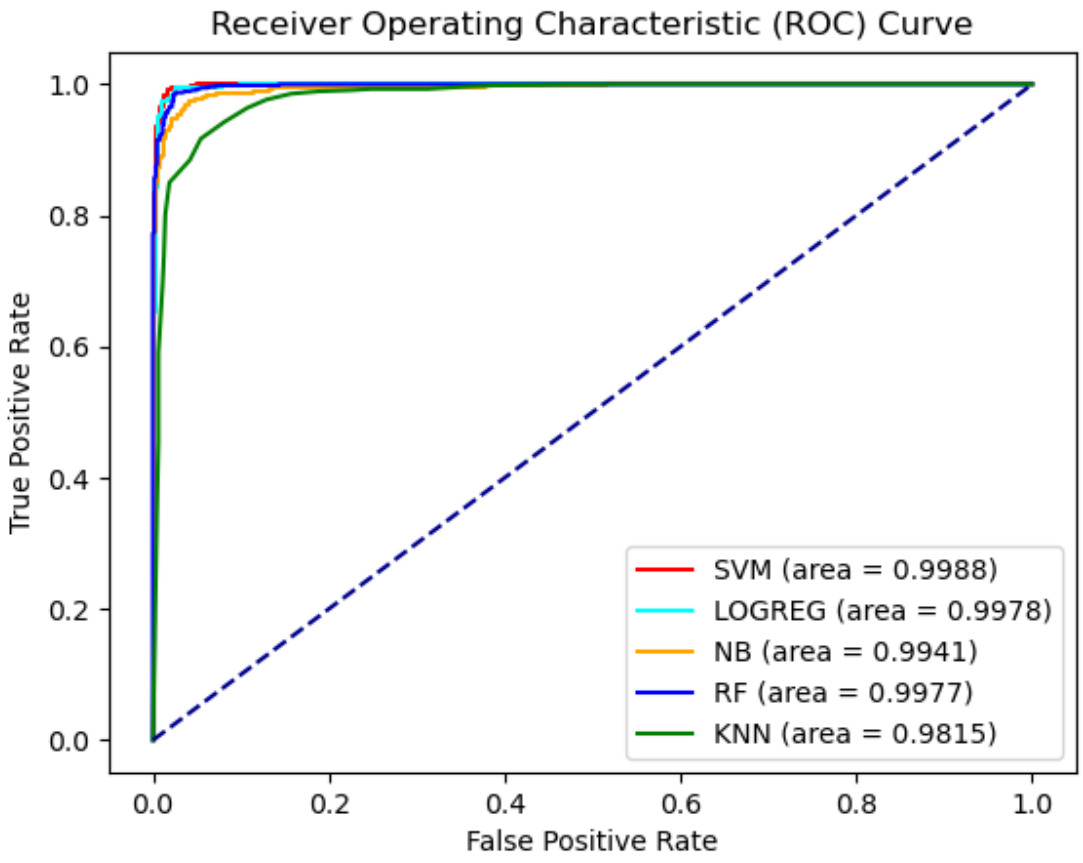


Figure 5.1. *ROC curve for the classification results on the testing data*

Area under the curve (AUC) for the support vector machine (SVM) is the highest compared with other classifiers.

The ROC curve for LSTM is drawn separately in Figure 5.2. The area under the curve AUC of the LSTM is almost close to SVM.

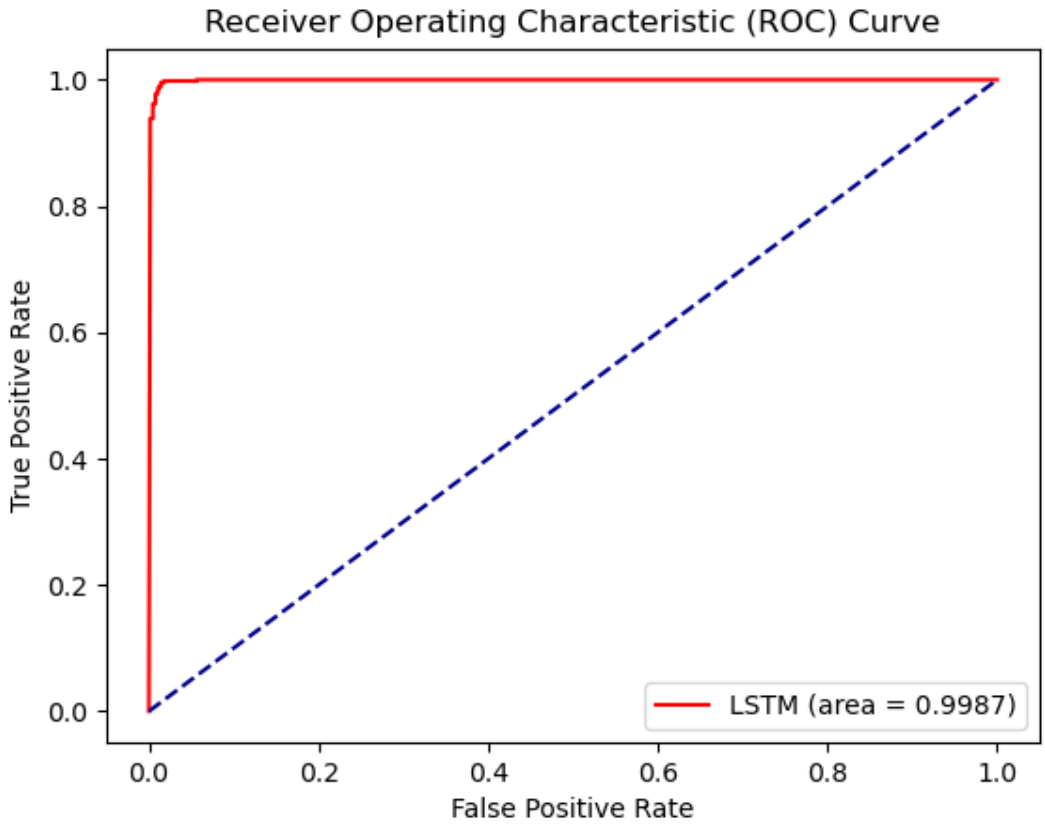


Figure 5.2. *ROC curve for LSTM model.*

Classification on body and headline were also done. The results for body are given in Table 5.7.

Table 5.7. *Performance of classifiers for the Body of the article with AUC*

Classifier	Body	fake observed	real observed	precision	recall	F1 score	Accuracy	AUC
Naive	fake - n = 643	605	38	0.99	0.94	0.96		

Bayes	real - n = 554	8	546	0.93	0.99	0.96	0.96	0.9939
SVM	fake - n = 643	629	14	0.99	0.98	0.99	0.98	0.9985
	real - n = 554	5	549	0.98	0.99	0.98		
Logistic Regression	fake - n = 643	625	18	0.99	0.97	0.98	0.98	0.9980
	real - n = 554	5	549	0.97	0.99	0.98		
Random Forest	fake - n = 643	602	41	1.00	0.94	0.97	0.96	0.9971
	real - n = 554	2	552	0.93	1.00	0.96		
K-nearest Neighbour	fake - n = 643	597	46	0.95	0.93	0.94	0.94	0.9824
	real - n = 554	30	524	0.92	0.95	0.93		
LSTM	fake - n = 643	624	19	0.97	0.97	0.97	0.97	0.9980
	real - n = 554	17	537	0.97	0.97	0.97		

The results for body are almost the same as the results for the combination of body and headline. The AUC of Body for each of this classifier is presented Table 5.7. As we can see, the highest AUC is for SVM model with 0.9985. The AUC of Logistic regression is 0.9980, Random forest is 0.9971, Naïve Bayes is 0.9939, K nearest neighbours is 0.9824 and LSTM with (0.9980). The AUC of all classifiers are close to 1.0 except K-nearest Neighbour which shows Perfect fit.

The results for accuracy and confusion matrix for headline is presented in Table 5.8.

Table 5.8. *Performance of classifiers for the Headline of the article with AUC*

Classifier	Headline	fake observed	real observed	precision	recall	F1 score	Accuracy	AUC

Naive Bayes	fake - n = 643	601	42	0.87	0.93	0.90	0.89	0.9631
	real - n = 554	87	467	0.92	0.84	0.88		
SVM	fake - n = 643	586	57	0.89	0.91	0.90	0.89	0.9499
	real - n = 554	71	483	0.89	0.87	0.88		
logistic regression	fake - n = 643	586	57	0.89	0.91	0.90	0.89	0.9587
	real - n = 554	72	482	0.89	0.87	0.88		
Random forest	fake - n = 643	520	123	0.92	0.81	0.86	0.86	0.9426
	real - n = 554	45	509	0.81	0.92	0.86		
K nearest neighbour	fake - n = 643	603	40	0.74	0.94	0.83	0.79	0.8819
	real - n = 554	210	344	0.90	0.62	0.73		
LSTM	fake - n = 643	563	80	0.84	0.88	0.86	0.85	0.9412
	real - n = 554	105	449	0.85	0.81	0.83		

The accuracy of the model for headline is the highest for support vector machine (SVM) with 89.31%. The model's accuracy is much improved compared with the one obtained by (Agarwalla et al, 2019) which has 68% as the highest accuracy found in Naive Bayes classifier. This shows 21% improvement in the accuracy. Moreover, The AUC of Headline for each of this classifier is presented Table 5.8. As we can see, the highest AUC is for Naïve Bays model with 0.9631. The AUC of Logistic regression is 0.9587, Random forest is 0.9426, SVM is 0.9499, K nearest neighbours is 0.8819 and LSTM with 0.9412.

5.2 Fake-news detection for the second dataset

5.2.1 Influence of general characteristics on the news article

Like the previous data to see the influence of the general characteristics on the outcome, a logistic regression model was trained and tested by using these characteristics items. The data was split into 70% as training data and 30% as testing data. A 5-fold cross validation model was shown that there is a significant model which is much better than the intercept model using these simple characteristics of the data. Log-likelihood of the intercept model was -9795.5. This value was increased by 29.5% to a log-likelihood of -6903.0, showing a pseudo-R squared of 0.295. The p value of the log-likelihood ratio test is 0.000 < 0.01 which is significantly better than the restricted model which has only intercept. The model of logistic regression including these simple characteristics are given in Table 5.9.

Table 5.9. *Logistic Regression Results with Stopwords*

Parameters	Coef	std err	z	p > |z|	LB	UB
WordCountH	-0.1063	0.004	-23.858	0.000	-0.115	-0.098
WordCountB	-0.0885	0.002	-39.309	0.000	-0.093	-0.084
stopwords	-0.0034	0.001	-3.324	0.001	-0.005	-0.001
noun	0.0937	0.002	38.315	0.000	0.089	0.098
verb	0.0787	0.003	29.670	0.000	0.073	0.084
adj	0.0960	0.002	38.578	0.000	0.091	0.101
Wordsinsent	0.0604	0.002	27.669	0.000	0.056	0.065

In this data real news is 0 and fake news is 1. This model shows that word counts in the headline are more in real news compared with fake news. The log-odds of -0.1063, say that adding one more word to the title of the news increases the odds of being real news by 11% (1/exp (-0.1063) = 1.11). This coefficient is reverse compared with the one found for the previous dataset. The sentence count of body is removed from this model since the Bayesian Information Criterion (BIC) of the model without sent count (number of sentences in body of each article news) is 13872. The model including sentence counts of body has BIC equal 13874. Hence the model without sent count (number of sentences in body of each article news) is better than the model with this parameter.

The accuracy of this model after implementing 5-fold cross validation on the testing data is 78.57%. The model accuracy is good by using these simple parameters in the model.

For better interpretation of the odds ratio, it is better to remove the stopwords from the tags, so no count of nouns and verbs are coming from stop words. After removing the stopwords from the tags and adding # of non-alphabetic words to the model the pseudo-R squared becomes 0.34. which means the log-likelihood was improved by 34% compared with the null model (model which includes only intercept). The model without sentence count has BIC equal 12962, while adding sentence count of body to it will be 12992. Hence the model without sentence count is better. The last model after reducing stopwords from counts of tags and adding number of non-alphabetic to the model is given in Table 5.10.

Table 5.10. *Logistic Regression Results after Reducing Stopwords*

Parameters	Coef	std err	z	p > |z|	LB	UB
WordCountH	-0.0958	0.005	-21.162	0.000	-0.105	-0.087
WordCountB	-0.0979	0.002	-39.505	0.000	-0.103	-0.093

stopwords	0.1043	0.003	40.356	0.000	0.099	0.109
noun	0.1050	0.003	38.405	0.000	0.100	0.110
verb	0.0566	0.003	19.698	0.000	0.051	0.062
adj	0.1169	0.003	39.749	0.000	0.111	0.123
Wordsinsent	0.0613	0.002	27.213	0.000	0.057	0.066
Non-alpha	0.0997	0.003	29.113	0.000	0.093	0.106

The above model shows negative log-odds for word counts in headlines. For the body if the word is a verb the log-odds will be negative 0.0566 - 0.0979 = -0.0413. The odds ratio for being real news is 1/exp (-0.413) = 1.04. So, it is 4% more likely to be real news. If the added word is stopwords, noun or non-alphabetic it will be positive. These positive log-odds are around (0.1043 - 0.0979) = 0.0064 > 2 standard error (SE) for stopwords. For nouns it is (0.1050 - 0.1043) = 0.0071 > 2 SE and for non-alphabetic words it is (0.0997 - 0.0979) = 0.001 < SE, but since not all the Non-alphabetic values are included in counts of word for body this coefficient is not removed from the model. More adjectives are also with positive log-odds (0.1169 - 0.0979) = 0.019 > 2 SE. Hence more stopwords, nouns, non-alphabetic words and adjectives make the news to have higher odds of being fake news. The average words in sentences have positive log-odds of 0.0613. So, news articles which have on average longer sentences (more words in the sentences) are more likely to be fake news. More verbs make the news more likely to be real news. In the paper released by (Vijayaraghavan et al, 2020) they also show that using distribution of that tag fake news has more adverbs and adjectives. But they did not mention verbs while in their frequency plot for both before preprocessing and after preprocessing the verb were more in real news compared with fake news. They mentioned that pronouns are more in

real news compared with fake news. In getting the general characteristics in this study, words were tagged as nouns, verbs, and adjectives.

The model accuracy after implementing 5-fold cross validation on the testing data is 79.54%.

5.2.2 Feature extraction

Three feature extraction methods were used for the second dataset which are explained below.

1- count vectorizer

2- Term frequency inverse document frequency vector (TF-IDF)

3- word2vec embedding

5.2.2.1 Count vectorizer

Count vectorizer had the best performance for the second dataset in the analysis performed by (Vijayaraghavan et al, 2020)[18]. They had an accuracy of 94.88% using the count vectorizer feature with the Long Short-Term Memory (LSTM) model. The count vectorizer scales the count for each token to be between zero and one. The top 12 features extracted by count vectorizer with setting threshold of min document frequency as 5 with 1-gram tokens are presented in Table 5.11.

Table 5.11. *1-gram count vector for top 12 features in the dataset for Count Vectorizer*

Term	Mean in Real news	Mean in Fake news	t statistics	p-value
castle_jj	0.000674	0.000306	-1.176	0.239
History	0.000000	0.001019	3.254	**0.001**
sophisticate_vbn	0.000674	0.000306	-1.176	0.239

historic_vb	0.000481	0.000509	0.089	0.928
seclude_vbd	0.000578	0.000407	-0.543	0.586
obstruct_vbn	0.000193	0.000815	1.988	0.0468
let_rbr	0.000289	0.000713	1.355	0.1753
secessionist_jj	0.000193	0.000815	1.988	0.0468
agony_jj	0.000481	0.000509	0.089	0.928
obsess_vbg	0.000481	0.000509	0.089	0.928
wonggetty_nnp	0.000000	0.001019	3.254	**0.001**
group/n	0.000578	0.000407	-0.543	0.586

It could be seen in the Table 5.12 that from top 12 count vectorizer features, four of them have p-value less than 5 percent which show significant difference between the word count in fake and real news. “history”, “obstruct_vbn”, “sessionist_jj” and “wonggetty_nnp” are the tokens which have significant differences between fake news and real news at 5% significance level.

In this study, 1 to 3 grams of the consecutive tokens were considered in the feature extraction, as in the analysis done using Grid search to find optimal values for the tuning parameters it was seen that (1,3) gram was superior. The count vectorizer for top 12 tokens of (1,3) gram is presented in Table 5.12.

Table 5.12. *(1,3) gram count vector for top 12 features in the dataset for Count Vectorizer*

Term	Mean in Real news	Mean in Fake news	t statistics	p-value
suggest_vbd united_nnp state_nnps	0.000866	0.000102	-2.442	0.014
outlook_nnp region_nnp usa_nnp	0.000000	0.001019	3.254	**0.001**

outlook_nnp popular_nnp article_nnp	0.000000	0.001019	3.254	**0.001**
outlook_nnp popular_nnp	0.000000	0.001019	3.254	**0.001**
sit_nn watch_nn	0.000481	0.000509	0.0894	0.928
sit_vbd back_rp	0.000289	0.000713	1.355	0.175
video_nn agency_nn	0.000000	0.001019	3.254	**0.001**
outlet_nns say_vbd	0.000674	0.000306	-1.176	0.239
bachelor_jj	0.000866	0.000102	-2.442	0.014
back_nnp trump_nnp	0.000578	0.000407	-0.543	0.586
heavily_rb attend_vbd	0.000000	0.001019	3.254	**0.001**
heavily_rb militarize_vbn	0.000096	0.000917	2.621	0.0087

In the Table 5-13, tokens from one to three gram could be seen. From top 12 tokens in 8 of them there is a significant difference between the count of token in the fake and real news. “suggest_vbd united_nnp state_nnps” and “bachelor_jj” tokens are seen to be significantly with higher counts in the real news compared with fake news. In contrast, tokens “heavily_rb militarize_vbn”, “heavily_rb attend_vbd”, “video_nn agency_nn”, “outlook_nnp region_nnp usa_nnp” , “outlook_nnp popular_nnp article_nnp” and “outlook_nnp popular_nnp” have significantly more counts in the fake news compared with real news.

5.2.2.2 TF-IDF vectorizer

Term frequency inverse document frequency for the second dataset was calculated. This feature was used with setting minimum document frequency threshold to 5. Since the results for (1,3) gram was seen to be superior compared with 1-gram, the TF-IDF vectorizer was used with 3 grams. In Table 5.13 top 12 TF-IDF features are presented.

Table 5.13. *(1,3) gram count vector for top 12 features in the dataset for TF-IDF vectorizer*

Term	Mean in Real news	Mean in Fake news	t statistics	p-value
channel_nnp list_nn	0.000000	0.000012	3.254	**0.001**
aftermath_vbp trump_nnp	0.000000	0.000013	3.409	**0.000***
nobel_nnp prize_nnp speed_nnp	0.000000	0.000013	3.409	**0.000***
explain_vbn disproportionately_rb large_jj	0.000000	0.000013	3.409	**0.000***
america_nnp ground_nn	0.000000	0.000013	3.409	**0.000***
house_nnp sudden_nnp	0.000000	0.000013	3.409	**0.000***
scientist_nns aftermath_vbp trump_nnp	0.000000	0.000013	3.409	**0.000***
scientist_nns aftermath_vbp	0.000000	0.000013	3.409	**0.000***
explain_vbn disproportionately_rb	0.000000	0.000013	3.409	**0.000***
ground_nn clinton_nnp foundation_nnp	0.000000	0.000013	3.409	**0.000***
house_nnp sudden_nnp rise_nn	0.000000	0.000013	3.409	**0.000***
inevitable_jj move_nn	0.000000	0.000013	3.409	**0.000***

using TF-IDF vectorizer with 1-to-3-gram consecutive tokens, it could be seen that for all top 12 features the tokens have significantly higher TF-IDF score in fake news compared with real news.

5.2.2.3. word2vec embedding

Word-to-vector embedding was used to generate vectors for each word in the vocabulary. It was considered that each word should at least have 10 document frequencies. For implementing word2vec embedding genism library of python has been used. The word2vec embedding uses neural network model to create embedding words for each word in the vocabulary. After fitting word2vec to the data using 5 epochs, a vector is created for each token in the vocabulary. The

model calculates the most similar words to each token as an example for the word “castle_jj” which has high frequency count with the 10 most similar words are as below.

1- ('castle_NN', 0.8477572202682495)

2- ('dusty_JJ', 0.8004491329193115)

3- ('balcony_NNS', 0.7949758768081665)

4- ('expanse_NN', 0.7944581508636475)

5- ('square_NNS', 0.7905918955802917)

6- ('hill_NNS', 0.7862504720687866)

7- ('shack_NNS', 0.7838928699493408)

8- ('magnificent_JJ', 0.7835989594459534)

9- ('garden_NNS', 0.7818024158477783)

10- ('spacious_JJ', 0.7815918922424316)]

These word keys and the values beside them are the most similar words to “castle_jj” and the value is the cosine similarity between them.

The top similar words to the token “history_NN” which had the second rank in the frequency count is as below:

1- ('historian_NNS', 0.664933443069458)

2- ('historical_JJ', 0.618412971496582)

3- ('modern_JJ', 0.6089843511581421)

4- ('great_JJS', 0.5715377330780029)

5- ('century_NN', 0.5286176204681396)

6- ('transformational_JJ', 0.5278278589248657)

7- ('exceedingly_RB', 0.5232303738594055)

8- ('slavery_NN', 0.5128378868103027)

9- ('annals_NNS', 0.4981594681739807)

10- ('achievement_NNS', 0.48913639783859253)]

As we can see, century, modern, historical, historian and annals are in the top words(By listed them with the cosine similarity. and just wrote those which are related to date and seems to be similar below of it) which is quite reasonable. Table 5.14. shows 5 other tokens which have high frequency count with their 10 most similar words below them.

Table 5.14. *Most similar words to top frequent tokens*

sophisticate		seclude		obstruct		secessionist		agony	
similar tokens	similarity	similar tokens	similarity	similar tokens	similarity	similar tokens	similarity	similar tokens	similarity
Interconnect	0.713	mount	0.738	stillness	0.616	republic	0.627	attack	0.622
Manipulate	0.698	village	0.706	impassioned	0.604	eritrea	0.626	assault	0.621
Inevitability	0.694	mural	0.705	vent	0.598	region	0.608	drag	0.618
Dominate	0.689	desert	0.700	respectful	0.595	continent	0.606	bystander	0.615
Myriad	0.675	spring	0.696	disconnect	0.584	destabilize	0.603	wound	0.612

Revert	0.668	upper	0.692	bury	0.582	rwanda	0.589	horrific	0.608
Entanglement	0.665	lake	0.689	blaze	0.581	regional	0.580	disturbed	0.607
Aggregate	0.661	lower	0.686	audible	0.579	colonial	0.580	mateen	0.602
Implement	0.659	built	0.683	fist	0.577	destabilization	0.569	uproar	0.593
Enigma	0.657	head	0.681	react	0.573	djibouti	0.559	suffer	0.593

To visualize the words, we can set the words in the two principal components. The top 100 tokens with mean counts of 500 are visualized in Figure 5.3.

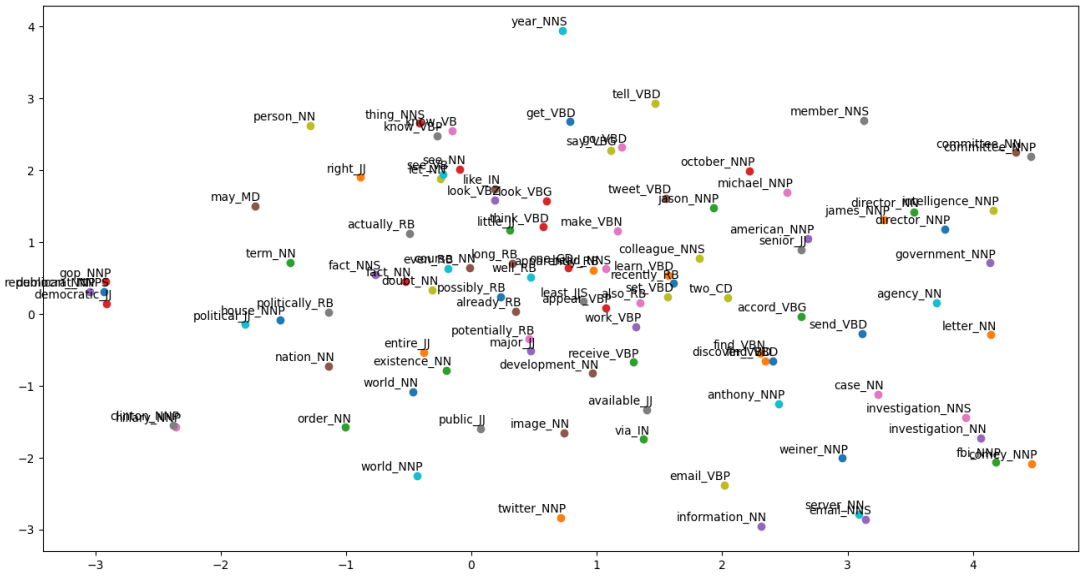


Figure 5.3. *Visualization of top 100 tokens with mean counts of 500*

5.2.3 Classification Results for the Second dataset

Classification analysis has been done using this big data which include 20203 non-missing rows. Vijayaraghavan et al, (2020) have used 3-fold cross validation using this dataset. For comparison, a 3-fold cross validation was done using feature extraction methods of count vectorizer, TF-IDF vectorizer, word2vec embedding and implementing classification models of support vector machine (SVM) with C = 10^{-05}, Logistic Regression with alpha = 0.1 and Random Forest model with number of estimators as 1000. Using these models and implementing cross validation the accuracy achieved is a bit higher (97.22%) than accuracy found by (Vijayaraghavan et al, 2020). They have claimed that the count vectorizer performed best in this dataset, while in our analysis TF-IDF vectorizer with (1,3) gram performs best with highest accuracy compared with other models in Table 5.15.

Table 5.15. *Results of three classification models compared with (Vijayaraghavan et al, 2020)*

Feature extraction	Support vector machine		Logistic Regression		Random forest	
	Current study	(Vijaya Raghavan et al, 2020)	Current study	(Vijayaragh avan et al, 2020)	Current study	(Vijayaragh avan et al, 2020)
countvect	**95.78**	93.06	**97.50**	94.45	**96.19**	87.64
TF-IDF vect	**97.76**	94.58	62.75	94.79	**96.25**	87.64
word2vec	87.11	91.17	82.00	91.30	86.3	88.60

The TF-IDF was the best feature extraction method and like the analysis done by Vijayaraghavan et al, (2020) the word2vec is the worst feature extraction method for these datasets.

5.2.3.1 Grid search for Support Vector Machine (SVM)

Grid search with pipeline was used for the second dataset also to obtain the optimal tuning parameters. In the feature extraction the maximum document frequency was used as ratios of 0.1, 0.2, 0.5, 0.75,0.95 and 1.0. For the maximum features to be extracted 2 values of None and 1000 have been used. None means that no threshold should be considered for the maximum number of features. Unigram, bigrams and trigrams features were extracted in the grid search. For this reason, N-gram features (1,1), (1,2), (2,2), (1,3), (3,3) have been used. The penalty of L2 and elastic.net were used by alpha (L1 penalty is value of the absolute of the coefficients and L2 penalty is value of square of the coefficients and elastic.net is combination of L1 and L2 penalty) taking values of 1e-5, 0.1, 1/64. and 1/128. For normalization parameter T a value of 12 and None were considered in the grid search. Use idf has two options of True and False in grid search which allows checking both using Inverse document frequency (IDF) or just using the term frequency (TF). minimum document frequency also was used for neglecting the tokens which have very low frequencies. A range of 0.001, 0.002, 0.005, 0.007, 0.008, 0.009, 0.01, 0.02 was used in Grid search for minimum documents frequency.

After implementing the grid search with 5-fold cross validation for each grid, it was seen that among these parameters the maximum document frequency of 0.95 which do not set limits for document frequency, maximum feature as None which sets no limitation for maximum features, Penalty L2 with alpha equal 1e-05, normalization parameters of 12 were found to be optimal. It was seen that (1,3) gram which means including 1-gram, 2-gram and 3-gram features in the feature vector has been found as the optimal feature. The minimum document frequency equal 0.007 was found as the optimal value. The accuracy of 5-fold cross validation for the support vector machine is 97.76%.

5.2.3.2 Grid search for Naive Bayes classifier

For Naive Bayes classifiers with lidstone smoothing parameter alpha, four values of (0.1, 0.2, 0.5, 1) have been used in the grid search. Other parameters for feature extraction like maximum document frequency (0.1, 0.2, 0.5, 0.75 and 1.0), maximum number of features in the feature matrix were considered with the values (None, 100, 200, 500, 1000). The N-gram options of (1,1), (1,2), (2,2), (1,3) and (3,3) has been used. The grid search results for Naive Bayes classifier show that smoothing parameter alpha = 1.0 is optimal. The maximum document frequency was found to be 0.5. No threshold for maximum features was considered. (3,3) grams or trigram was found to be best. The 5-fold cross validation accuracy for Naive Bayes using the optimal parameters was found to be 0.90.

5.2.3.3 Grid search for Logistic Regression

Logistic Regression was applied with the same parameters as in SVM. The optimal values were found to be L2 penalty with alpha = 1e-05. The maximum document frequency for Logistic Regression was found to be 1.0. No threshold for maximum features were considered. The (1,3) gram was found to be optimal. Results of the Count vectorizer were better compared with TF-IDF. The minimum document frequency of 0.007 was found to be a good option for removing the tokens with very low frequency. The accuracy of Logistic Regression was found to be 0.975.

5.2.3.4 Grid search for Random Forest

In the Random Forest, values of None and 10 was used for maximum depth of tree. The maximum documents frequency of 0.1, 0.2, 0.5, 0.75 and 1.0 was tested. The maximum features of None, 100, 200, 500 and 1000 have been used. The N-Gram values of (1,1) , (1,2), (1,3), (2,2) and (3,3) have been used. The optimal value of maximum features was found as 500, the (1,1) gram was found to be best in Random Forest. The maximum document frequency 1.0 was found

to be optimal. No maximum depth was considered for each decision tree. The random forest model with 100 estimator trees, the minimum samples for splitting each node was 2 and the maximum features in the split was taken as the square root of the number of features in the model. Gini impurity was used as criteria for fitting the Random Forest model. The accuracy of the Random Forest model was found to be 96.19% using the optimal tuning parameters.

5.2.3.5 Grid search for K-nearest Neighbour

For k-nearest Neighbour in the grid search, k values of (5, 10, 15, 20 and 25) have been used. The options for document frequencies and N-gram were similar to Naive-Bayes. Maximum document frequency of 0.75 was selected as the threshold. The (1,3) gram was found to be superior compared with unigram and bigram. The value of K=15 was found as the optimal value. The k-nearest Neighbour classifier shows the accuracy of 84.9% with the 5-fold cross validation using optimal parameters.

5.2.3.6 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) was implemented, and the analysis was performed by setting 160k as the maximum number of the words, setting embedding dimension as 256, using two recurrent layers with 256 and 128 number of neurons, respectively. One dense hidden layer with 32 neurons and RelU activation function and finally the last layer includes 2 outcomes with sigmoid activation function. Between each layer drop out of 10% was used, for each recurrent layer a drop out of 25% was used and for the last dense layer drop out of 20% was used to avoid overfitting problems. The LSTM model was trained with an Adam optimizer and by using categorical cross entropy as a loss function. A batch size of 64 articles was set in each batch. Then the analysis was implemented for 10 epochs. Features which were used for the LSTM models were generated from sequences of words, maximum number of sequences considered to

be 1100. The LSTM model achieved an accuracy of 97.1% which is a bit more than the best accuracy found by Vijayaraghavan et al, (2020) using three LSTM models.

5.2.3.7 Summary of classification

The optimal values found using grid search were entered for each model. The data was split randomly by 70:30, keeping 70% of the data in the model for training and setting 30% out for testing. All 6 classification models were trained, and the models were tested using testing data. The accuracy of the testing data (precision, recall, F1 score, AUC and total accuracy) and confusion matrix are presented in the Table 5.16.

Table 5.16. *Performance of classifiers for the text with AUC*

Classifier	Text	fake observed	real observed	precision	recall	F1 score	Accuracy	AUC
Naive Bayes	fake - n = 1559	1542	17	0.80	0.99	0.88	0.87	0.9848
	real - n = 1441	386	1055	0.98	0.73	0.84		
SVM	fake - n = 1559	1521	38	0.98	0.98	0.98	0.98	0.9956
	real - n = 1441	29	1412	0.97	0.98	0.98		
Logistic Regression	fake - n = 1559	1520	39	0.98	0.97	0.98	0.97	0.9953
	real - n = 1441	36	1405	0.97	0.98	0.97		
Random Forest	fake - n = 1559	1471	88	0.96	0.96	0.95	0.95	0.9877
	real - n = 1441	64	1377	0.94	0.96	0.95		

K-nearest Neighbour	fake - n = 1559	1441	118	0.82	0.92	0.87	0.85	0.9408
	real - n = 1441	318	1123	0.90	0.78	0.84		
LSTM	fake - n = 1559	1484	75	0.99	0.95	0.97	0.97	0.9960
	real - n = 1441	12	1429	0.95	0.99	0.97		

Among the classifiers, support vector machines (SVM) with predicting 1521 out of 1559 fake news correctly and 1412 out of 1441 real news correctly is with the highest accuracy, precision, and recall. 38 fake news articles were classified as real news wrongly and 29 real news were predicted as fake news wrongly. The total accuracy is 97.7%. The ROC curve for each of this classifier is presented in Figure 5.4 and Figure 5.5 also in Table 5.16. As we can see, the highest AUC is for LSTM with 0.9960. The AUC of Naïve Bays is 0.9848, Logistic regression is 0.9953, Random forest is 0.9877, SVM is 0.9956 and K nearest neighbours is 0.9408.

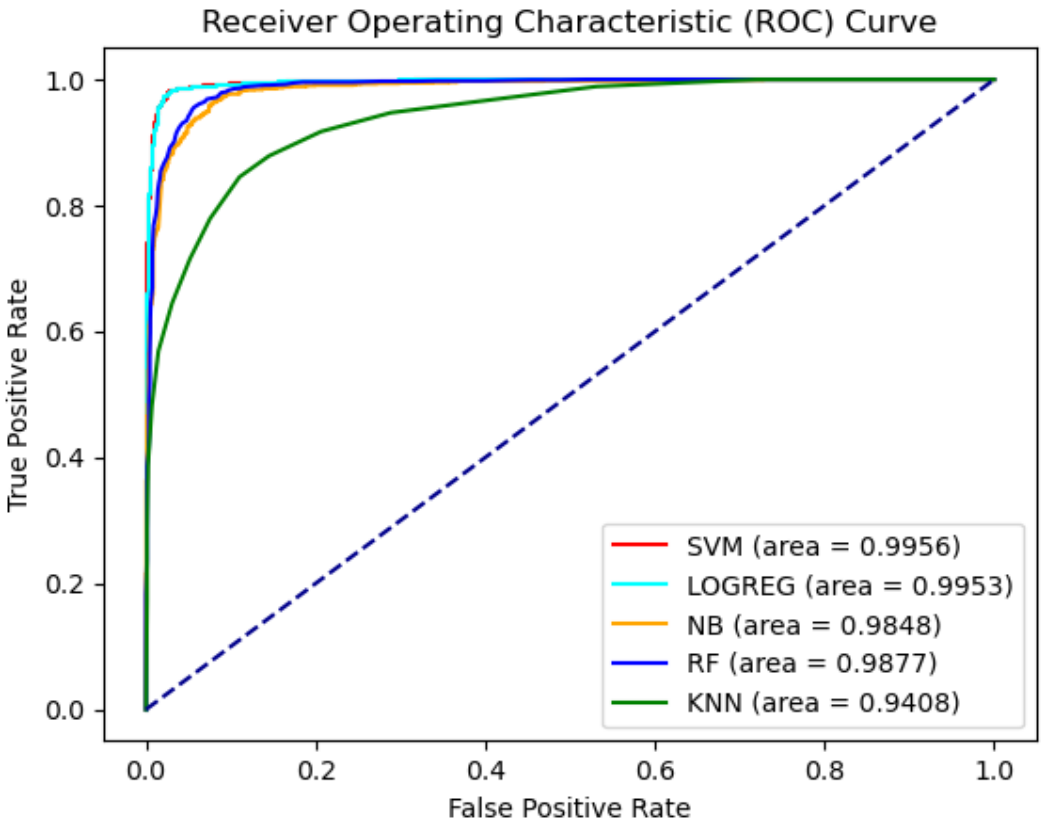


Figure 5.4. *ROC curve for the classification results on the testing data*

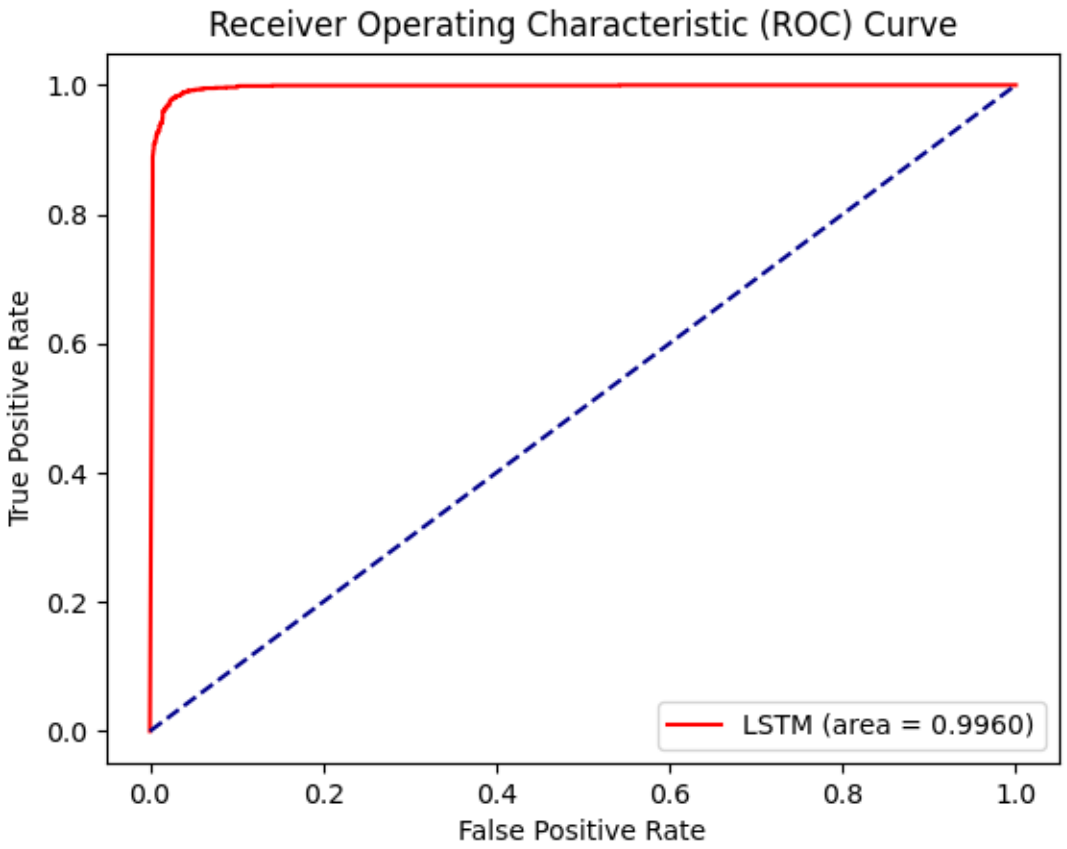


Figure 5.5. *ROC curve for LSTM model.*

Chapter 6

Conclusion and Future Work

A methodology was proposed to detect fake news using deep learning models and natural language processing (NLP). In this study, two datasets were analyzed, and classifiers were trained to develop a model that can predict whether the news is real or fake. The conclusion for each of the two studied datasets is presented in this chapter.

6.1 Conclusion of the study on the first dataset

The first dataset includes article news from 12 news websites. The collected data from each URL is such that all the news is fake or real from each of these 12 websites. For instance, all the news from "reuters.com" is real, and all the news from "beforeitsnews.com" is fake news. The dataset was cleaned by preprocessing, removing the URLs, punctuations, stop words and digits from the texts. Before removing these features, some general aspects like several stop words, punctuations, words, sentences, verbs, nouns, and adjectives were calculated. It was seen that these aspects are significantly different in fake news and real news. Hence, removing them without pulling out these measures yields missing information, which helps distinguish between real and fake news. After preprocessing the data, the Punkt statement tokenizer analyzed the cleaned data to detect the sentences and position of the words in the sentences. The tagged words were used for feature extraction by count vectorizer and TF-IDF vectorizer. It was seen that TF-IDF vectorizer with (1,2) Gram (one and two consecutive tokens) gets the best features since the highest cross-validated accuracy was found by using (1,2) gram features. For classification, the Naive Bayes classifier, Support Vector Machine (SVM), Logistic Regression, K nearest Neighbours (KNN) and Random Forest were used to classify the article news as fake or real. The

model with the support vector machine (SVM) achieved the highest accuracy (98.5%) in combination with body and headline and area under the ROC curve (AUC) (0.9988). Also, the LSTM model with three recurrent layers and one hidden layer was used as a deep learning method to classify the article news. The results of the LSTM model (98.03%) and ROC Curve (AUC) 0.9984. The model's accuracy considering the combination of body and headline is very high, especially in the SVM model.

Some interesting information was found using general features. It was observed that news with a greater number of words in the headline, news with more stop words used in the body part are more likely to be fake news. In contrast, news with a greater number of sentences in the body part, more words which are verbs or adjectives in the body part make the news more likely as real news. Therefore, one can conclude that real news does not tend to use big headlines, but more explanation in the body part. More sentences, especially those which include a greater number of words (as verbs or adjectives), increase the probability of being real news. More stopwords are found in the fake news compared with real news. The nouns in the body of the news articles do not increase the probability of either real or fake news. A notable thing noticed in the features was the word "photo" and "image" appeared more in the real news, while the word "video" appeared more in the fake news. Commonly used words such as "great" are more frequently seen in the fake news compared with real news. The fake news tries to pretend that it is real news, so the feature "law" can be seen more frequently in fake news than real news.

6.2 Conclusion of the study on the second dataset

The second dataset also consisted of news articles taken from Kaggle.com, was analyzed with the same methodology, which was performed on the first dataset. In this data also it was seen that general features such as number of words in the headline, number of words in the body,

number of stopwords used in each article news, number of verbs, nouns and adjectives used in the article news, the average number of words in the sentences of the body of the news have a significant effect on the news class. Some of these features are usually removed from the texts in the preprocessing stage. However, it was seen that there is a significant difference between total stopwords used in the fake news compared with real news. Fake news articles tend to use more stopwords compared with real news articles. This conclusion was common in both datasets used in this study. Non-alphabetic words are also removed in the preprocessing step to perform logistic regression. However, it was found that non-alphabetic words are significantly higher in fake news compared with real news. For the second dataset, it was observed that nouns, adjectives, stopwords and non-alphabetic words were used more in fake news than real news. The real news had more verbs compared with fake news. In the first dataset, it was observed that the fake news has longer headlines. But in the second dataset, the results were reversed. It was seen that real news has longer headlines significantly. So, this parameter differs between the two datasets. Another difference was the use of adjectives, which were significantly more in this dataset's fake news. The classification analysis was done using support vector machines, logistic regression, and random forest. The analysis used tokenizers with 1-gram to 3-gram and tuning parameters of the classifiers. The results demonstrated that 3-gram gave the best accuracy of 97.7% and ROC Curve(AUC) 0.9956 with SVM for this dataset which is Higher then results shown by Vijayaraghavan et al. (2020). Also, using LSTM 97.1% accuracy and ROC Curve (AUC) 0.9960 was achieved which is more then the accuracy achieved by Vijayaraghavan et al. (2020) which is 94.88% and for Logistic Regression 97.50% accuracy and ROC Curve (AUC) 0.9953 was achieved which is an improvement then the accuracy achieved by Vijayaraghavan et al. (2020) which is 94.79%. However, their analysis gets the count vectorizer as the best feature

extraction method and word2vec as the worst performing on this dataset. This study found that word2vec with 3-gram was the best feature extraction method. It can be concluded that the use of certain features should be retained instead of removing them at the preprocessing step to get a higher accuracy of prediction, even without using deep learning models.

6.3 Suggestion for future studies

This analysis was performed on two different datasets. Using the first dataset, some idea could be used in future studies to improve the quality of the models:

1- Collecting data randomly from each world news website to reduce the bias in the dataset.

2- Knowing that after using the maximum document frequency for the words in the dataset, some of the words are missing, which may lead to missing some useful information for distinguishing between real and fake news.

It could be considered to add a feature as several commonly used words in each news article and see its difference between real and fake news.

3- More general features could be extracted from the data like number of hashtags, number of mentions, number of each type of punctuation separately, number of characters, number of digits and unit specifiers like (kg, lbs, inches, feet and so on) to find the difference between them in real and fake news.

The second dataset looked better and was a big dataset. The analysis performed on this dataset showed some common results with the previous dataset and some reverses. Hence it seems to be necessary to find which features are data-related and which are consistent in various data. Also, the published real and fake news behaviour could be investigated by the country of release. Several independent variables can be tested at the time of release to see if the changes are time-dependent or country dependent or just changed because the feature was not significant. It was

observed that considering the tag beside the words in the tokenizing is a useful idea. This idea could be investigated more to see how good it is and how it is possible to improve the tagging portion, which is added to the token. Depending on the tag connected to the word, whether the word is noun, verb or adjective, the frequency of the word can be added for that specific tag. However, it seems important to make a relationship between these tokens, which are very similar to each other in the feature extraction phase.

References

[1] Russell, S., & Norvig, P. (2002). Artificial intelligence: a modern approach.

[2] Kamble,R.,& Shah, D.(2018).Application of artificial intelligence in human life. International Journal of research Granthaalayah, 6(6),177-188. DOI- https://doi.org/10.29121/granthaalayah .v6.i6.2018.1363

[3] Indurkhya, N., & Damerau, F. J. (Eds.). (2010). Handbook of natural language processing, Vol. 2, CRC Press. ISBN: 978-1-4200-8593-8

[4] Seamans, R., & Raj, M. (2018). AI, labor, productivity, and the need for firm-level data (No. w24239). National Bureau of Economic Research. University of Chicago Press. DOI- https://doi.org/10.7208/chicago/9780226613475.001.0001

[5] Roh, Y., Heo, G., & Whang, S. E. (2019). A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Transactions on Knowledge and Data Engineering.

[6] Different Datasets Available at:https://www.kaggle.com/datasets

[7] García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining, pp. 195-243, Cham, Switzerland: Springer International Publishing. ISBN 978-3-319-10246-7

[8] Famili, A., Shen, W.-M., Weber, R., Simoudis, E., 1997. Data preprocessing and intelligent data analysis. Intelligent data analysis 1, 3–23.DOI- https://doi.org/10.1016/S1088-467X(98)00007-9

[9] Dom, B. E., & Vaithyanathan, S. (2003). U.S. Patent No. 6,584,456. Washington, DC: U.S. Patent and Trademark Office.

[10] Zhou, X., Zafarani, R., 2020. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. ACM computing surveys.DOI-https://doi.org/10.1145/3395046. (53), 5-40.

[11] Agarwalla, K. Nandan, S., Nair, V.A., Hema, D. D. (2019). Fake News Detection Using Machine Learning. *International Journal of Recent Technology and Engineering*, 7(6), 2277-3878.

[12] Looijenga, M.S., 2018. The Detection of Fake Messages using Machine Learning [WWW Document]. URL http://essay.utwente.nl/77385/ (accessed 1.16.21).

[13] Ei Wynne, H., & Zar Wint, Z. (2019). Content Based Fake News Detection Using N-Gram Models. *International Journal of Advanced Trends in Computer Science and Engineering*, 8(6), 2806-2810. DOI: 10.30534/ijatcse/2019/20862019

[14] Reis, J., Correia, A., Murai, F., Veloso, A., Benevenuto, F., & Cambria, E. (2019). Supervised Learning for Fake News Detection. *IEEE Intelligent Systems*, 34(2), 76-81. doi: 10.1109/mis.2019.2899143

[15] Ahmed, H., Traore, I., Saad, S., 2017. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques, in: Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 127–138. https://doi.org/10.1007/978-3-319-69155-8_9

[16] Khan, Junaed Younus & Khondaker, Md. Tawkat Islam & Iqbal, Anindya & Afroz, Sadia. (2019). A Benchmark Study on Machine Learning Methods for Fake News Detection. Available at: https://arxiv.org/abs/1905.04749.

[17] Thota, Aswini; Tilak, Priyanka; Ahluwalia, Simrat; and Lohia, Nibrat (2018) "Fake News Detection: A Deep Learning Approach", SMU Data Science Review: Vol. 1 : No. 3 , Article

10. Available at: https://scholar.smu.edu/datasciencereview/vol1/iss3/10

[18] Vijayaraghavan, Sairamvinay & Wang, Ye & Guo, Zhiyuan & Voong, John & Xu, Wenda & Nasseri, Armand & Cai, Jiaru & Li, Linda & Vuong, Kevin & Wadhwa, Eshan. (2020). Fake News Detection with Different Models. Available at: https://arxiv.org/abs/2003.04978.

[19] Jagadeesh, Namratha & Mathias, Sheryl. (2017). Detecting Fake News Tweets from Twitter. College of Information Studies University of Maryland, College Park Maryland, USA. Available at : http://www.jetir.org/papers/JETIR2004387.pdf

[20] Fake News detection [WWW Document], n.d. URL https://kaggle.com/jruvika/fake-news-detection (accessed 1.16.21).

[21] Camisani-Calzolari, M.(2012) Analysis of Twitter followers of the US Presidential Election Candidates: Barack Obama and Mitt Romney.Accessed from https://digitalevalutions.com

[22] Fake News [WWW Document], n.d. URL https://kaggle.com/c/fake-news (accessed 1.16.21).

[23] Kannan, S., Gurusamy, V., 2014. Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks 5, 7–16. Available at: https://www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining.

[24] Loper, E., Bird, S., 2002. NLTK: the natural language toolkit. arXiv preprint cs/0205028. DOI: 10.3115/1225403.1225421.

[25] Korenius, T. & Laurikkala, J. & Järvelin, K. & Juhola, M. (2004). Stemming and lemmatization in the clustering of Finnish text documents. Proceedings of the thirteenth ACM international conference on Information and knowledge management, CIKM'04. 625-633. DOI:10.1145/1031171.1031285.

[26] Teo, J. (2018). Text-Mining & Social Networks Documentation. Available at:https://readthedocs.org/projects/socialnetwork/downloads/pdf/latest/

[27] Aizawa, Akiko (2003). "An information-theoretic perspective of tf–idf measures". Information Processing and Management. **39** (1): 45–65.

[28] Mikolov, T. et al. (2013). Distributed Representations of Word sand Phrases and their Compositionality. . Available at : https://arxiv.org/abs/1310.4546.

[29] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273-297. DOI-https://doi.org/10.1007/BF00994018.

[30] Subramanian, R. Simon. (2013). Overfitting in prediction models–Is it a problem only in high dimensions. *Contemporary Clinical Trials* (36) 636–641. DOI-https://doi.org/10.1016/j.cct.2013.06.011.

[31] (Tutorial) Understanding Logistic Regression In Python [WWW Document], 2019. DataCamp:Community.URL:https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python (accessed 1.16.21).

[32] Berrar, Daniel. (2018). Bayes' Theorem and Naive Bayes Classifier. DOI:10.1016/B978-0-12-809633-8.20473-1.

[33] Swain, P.H., Hauska, H., 1977. The decision tree classifier: Design and potential. IEEE transactions on geoscience electronics 15, 142–147. DOI-https://doi.org/10.1109/TGE.1977.6498972

[34] Zhang Z. (2016). Introduction to machine learning: k-nearest neighbors. Annals of translational medicine, 4(11), 218. DOI-https://doi.org/10.21037/atm.2016.03.37

[35] Kalchbrenner, N. & Danihelka, I. & Graves, A. (2015). Grid Long Short-Term Memory. https://arxiv.org/abs/1507.01526.

[36] Levin, E., 1990. A recurrent neural network: Limitations and training. Neural networks 3, 641–650. DOI-https://doi.org/10.1016/0893-6080(90)90054-O