INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

Investigating the Performance of Fake News Prediction involving Political Activity with the use of Artificial Intelligence

Leander Pace

Supervisor: Mr Silvio Abela

June, 2022

Submitted to the
Institute of Information and Communication Technology
in partial fulfillment of the requirements for the degree of
B.Sc. (Hons.) Software Development

Authorship Statement

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr Silvio Abela.

[illegible]

Leander Pace

June 6, 2022

Copyright Statement

L. Pace

Leander Pace

57, Thorazia Court, Flat 3
Spadaro Street,
Zabbar, ZBR3465

June 6, 2022

Dedication

To my friends and family, for all their help and support.

Acknowledgements

I would like to express my deep gratitude to my mentor Mr. Silvio Abela, for his patient guidance, enthusiastic encouragement and useful critiques which were of great help towards the completion of this research.

Would also like to thank my friends, family and girlfriend who were of great support when I really needed it.

Abstract

The aim of this research is to observe the different model performances when attempting to detect general fake news and political fake news. Various classifiers were employed throughout this research; these are: Random Forest (RF), Decision Trees (DT), Support Vector Machine (SVM) and Long Short-Term Memory (LSTM). Various Natural Language Processing (NLP) techniques were also used such as: Term Frequency-Inverse Document Frequency (TF-IDF), CountVectorizer (CV) and Word2Vec (W2V). The primary dataset utilised was the LIAR while a politically configured version of LIAR was used as the secondary dataset. Both these datasets went through various preprocessing stages which include: the application of stemming and lemmatization and removing any stop words, punctuation, special characters and converting everything to lowercase. The data gathering phase consisted of applying all the models on both datasets and compare their findings mainly by their Accuracy, although the Precision, Recall and F-score were also taken into account. The experimental setups were set up in such a way that each model consisted of four experiments, each experiment alternating the test size and hyperparameters and then record any findings. The results obtained from the LIAR dataset achieved a very close accuracy to some other studies using similar preprocessing techniques, NLP techniques and classifiers. When observing the findings from the political dataset the results were a partial success over the LIAR dataset in terms of accuracy, but still inconclusive due to it being a very marginal improvement, leading to this study's hypothesis being partially satisfied. Throughout this research a number of limitations were met, primarily being the difficult construction of the LSTM models due to some unforeseen incompatibilities with the mentioned NLP techniques, aswell as having constructing a lot of various models which led to the generation of too many results, which was hard and time consuming to keep track of. As for future recommendations it is suggested to take caution in what research one shall commit to, keeping in mind the time needed in accomplishing said research.

Keywords— Fake News Detection, LIAR, Politics, Fake News, Preprocessing, Stemming, Lammetization, Random Forest, Decision Trees, SVM, LSTM, TF-IDF, CountVectorizer, W2V

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fake news has been around for as long as humans have existed. Wherever societal groups can be formed, people will always attempt to establish power. Before modern day technologies such as social media and printers, people used to disseminate fake news simply via word of mouth by spreading rumours. The ability to effect the knowledge of the people has been a prized asset for many centuries (Burkhardt 2017). Although the rise of technology and the internet may have bore many fruits, it also gave people the means to make fake news accessible to anyone with the desire to cause harm.

1.1 Purpose Statement

The damage fake news can create can be from the most minuscule of rumours to effecting major political decisions. Looking at the damage fake news have created in the past, not just locally but even around the world, especially during the Covid-19 pandemic, it has sparked the motivation to start understanding how to control the spread of fake news, especially in areas where its effects can be severe, such as politics. Social media helped fan the flames in terms of fake news, therefore manual means of fake news detection is not enough to help contain it from spreading. Automated fake news detection presents a myriad of possibilities to help detect thousands of fake news articles and social media posts in a short amount of time.

In this study we explore a few possibilities through some machine learning and text classification techniques and observe the results. Any findings are to be documented and compared at a later stage of the dissertation.

1.2 Hypothesis & Research Questions

The following text introduces the hypothesis and research questions which this research study will attempt to justify and answer in the conclusion.

Hypothesis: In this research study, it is hypothesized that prediction models will offer a greater accuracy when performed on political fake news rather than the more general fake news.

Research Questions

- *Can this research achieve a satisfactory result in detecting political fake news while corresponding with similar research studies?*
- *Can this research successfully predict the surge of political fake news?*

1.3 Scope

The chosen approach was to research and identify a variety of the current best performing techniques and datasets used in machine learning and text classification. This led to the discovery of four classifiers; Random Forest (RF), Decision Trees (DT), Support Vector Machine (SVM) and Long Short-Term Memory (LSTM), three NLP techniques commonly used for text classification; Term Frequency-Inverse Document Frequency (TF-IDF), CountVectorizer (CV) and Word2Vec (W2V), a number of precprocessing techniques; stop word removal, punctuation removal, special character removal, conversion to lowercase, lemmatization and stemming. The LIAR dataset was used as the primary dataset and an altered version of LIAR was used to act as the political dataset. These techniques were then incorporated within the prototype and observed the different results being generated which were later used for comparison.

The idea of this approach was to gather as much information as feasibly possible in order to construct the most optimal of models, based off the accumulated research found throughout this study in an attempt to determine whether or not political fake news can be detected better than general fake news.

1.4 Research Context

This paper consists of five major chapters; Introduction, Literature Review, Methodology, Results and Conclusion. The Introduction introduces what problem is being researched and what steps are to be taken into solving it. It also makes mention of the hypothesis and research questions which help set the objective of the this study.

The literature review explores the research conducted by other studies within the same research field which would help foreshadow the approach being taken in this research study.

The methodology will give a detailed explanation of the research strategy leading to construction of the prototype and what methods were used into building the models.

The results chapter will discuss the findings by going through every model and compare the yielded results. This will help identify the most optimal model.

The conclusion chapter will give a summary of the findings aswell as mention what limitations were encountered and what recommendations should be given for future work.

Chapter 2

Literature Review

Fake news has been rapidly evolving throughout the years, proving itself to have become a potent threat to our society and national welfare. It was only in 2016 that people labelled it as “Fake News” as it was gaining a high amount of page views due to a high spread of misinformation circling the US Presidential Election, becoming the buzzword we know today. In fact, it was Donald Trump himself who originated the term *Fake News* (Farhall et al. 2019). In mid-2016, Buzzfeed media editor Craig Silverman came across a stream of fabricated articles originating from a town called Veles in Eastern Europe – Silverman claimed: “We ended up finding a small cluster of news websites all registered in the same town in Macedonia called Veles”, who also identified 140 fake news websites which were trending on the social media platform Facebook as mentioned in “*How Teens In The Balkans Are Duping Trump Supporters With Fake News*”[1]. The main motivation of the culprits seems to have been money since they were generating a lot of profit via Facebook advertising and exploiting Facebook’s algorithm to help disseminate their stories.

[1]https://www.buzzfeednews.com/article/craigsilverman/how-macedonia-became-a-global-hub-for-pro-trump-misinfo

2.1 The Dangers of False Information

False information is often coined with propaganda (Nasir et al. 2021) since the people behind the false information stand to gain something from its impact towards others, as fake news tends to be biased in nature (Murayama et al. 2021). In addition to this, another tactic used by individuals generating fake news is the creation of a good title to lure in their readers. A good title can be the make-or-break factor which brings users to click and read on (Mazzeo et al. 2021). A lot can be said regarding the threats fake news is presenting in our modern, technological world, from hoaxes and rumours, to potentially changing the outcome of political elections (Shu et al. 2017).

As Heinrich explains it, the producer of an item often has intentions to misinform, with fake news having a commonality of how it is appropriated. It inherits some features from real news to hide under a veneer of legitimacy to make it seem more credible. Furthermore, through the use of news-bots, fake news imitates real news' omnipresence by building a fake news network (Heinrich 2019).

A fake news article by the name of "*Radiation fear prompt panic buying of salt*" from 2011[2], claimed that iodized salt would help counteract radiation effects, which resulted in sudden salt shortages in supermarkets in China. Further claims were made in another article outlining an incident of tensions, escalating between *Pakistan* and *India* due to a false news article reporting a Balakot strike, killing military personnel and the loss of expensive military equipment as reported on BBC News, *India* and *Pakistan*: "*How the war was fought in TV studios*"[3].

In a research conducted by (Levi 2017), it was reported that the financial market was also a victimised product of false information. A hoax circling around the internet about the asserted death of Ethereum's chief executive, resulted in a $4 billion drop in the company's market value. Additionally, the author makes mention of a Forbes article reporting that in 2013, there was a tweet about an alleged

[2]www.chinadaily.com.cn/2011-03/18/content_12189516.htm - accessed on 29/11/2021
[3]www.bbc.com/news/worldasia–47481757 - accessed on 29/11/2021

explosion, injuring Barack Obama which led to the loss of $130 billion in stock value in a matter of minutes (Levi 2017).

According to a statement made by van der Linden et al., stories that are credited as ‘false’ by fact-checking organisations tend to disseminate faster, farther and deeper than other types of news content. An article published by Monmouth University (Murray 2017) mentions a poll conducted in 2018, depicting 42% of Americans saying that major media sources report fake news to push an agenda, while 26% of Americans claim that said media sources report them by accident and poor fact checking. Left unchecked, trends such as these pose a problem to any healthy democracy, since it functions on accurate and independent news media as a source of information (van der Linden et al. 2020).

2.2 Industrial Mitigation Techniques

According to public discourse, the ideal way to combat fake news is through technological solutions, audience empowerment, and legal solutions. In fact, on the commercial front, the Federal Trade Commission (FTC) already took necessary steps to shutdown numerous websites supporting the spread of false information (Levi 2017). On the technological front, in rise of the adversity that is fake news, major social media companies such as Facebook, Google and Twitter, along with public institutions such as the European Commission, acknowledge this wide spread of misinformation and its negative impact on democracy, and are taking steps in technological advancements with the aim to monitor and delete fake accounts/bots, draft policy initiatives, and root out disinformation (Corbu et al. 2020). In addition to this, there are a numerous researchers and data scientists making their own technological contributions towards this cause via Machine Learning as seen in the following section.

2.3 Similar Studies

Classifiers such as Random Forests and Decision Trees together with the Term Frequency-Inverse Document Frequency (TF-IDF) NLP technique have been used to extract key words from fake news articles. These classifiers were both applied to the same dataset from Kaggle.com containing 20,800 articles. The results yielded showed that the Decision Tree (DT) classifier outperformed the Random Forest (RF) in terms of accuracy with a score of 89.11% and after 10 minutes execution time whilst the accuracy from the RF classifier had an accuracy score of 84.97% with 20 minutes execution time (Jehad & Yousif 2020).

In their research, Vijayaraghavan et al. examines the field of natural language processing (NLP) and makes use of three pre-training models; the CountVectorizer (CV), WordToVectorizer (W2V), and similarly to (Jehad & Yousif 2020), the TF-IDF. In addition to this, five other classifiers were used for the purpose of the research, namely: Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Long Short-Term Memory (LSTM), and Artificial Neural Network (ANN). The 20,800 record dataset employed in their research consisted of five features: ID, title, author, text and label. The model which obtained the most promising results utilised the LSTM classifier using the CV model with an accuracy of 94.88%. A close second was the Logistic classifier using the TF-IDF model with an accuracy score of 94.79%, followed by the ANNs classifier using the Word2Vec model with an accuracy score of 93.06% (Vijayaraghavan et al. 2020).

When the COVID-19 pandemic began during the late months of 2019, little did the people know how much it would affect the whole world especially in creating a major global unrest which in turn helped fake news to thrive amidst all the chaos and confusion. In a 2021 study, (Mazzeo et al. 2021) made use of a real-world Covid-19 dataset, re-sampling techniques (under/over-sampling) and similar to (Vijayaraghavan et al. 2020), they utilised five classifiers, namely SVM, Stochastic Gradient Descent (SGD), Logistic Regression (LR), Naïve Bayes (NB) and Random Forest (RF). Mazzeo et al. strongly points out that their research does not focus on predictive accuracy but rather on f1 score and recall metrics. The aim of the author's research is to probe imbalanced data, since predictive accuracy tends to be a misleading indicator, reflecting underlying class distributions. (Mazzeo et al. 2021) carries out two feature selection processes for their research; one is a textual feature selection and the other is a URL feature selection. The authors applied the aforementioned classifiers with both over and under sampling techniques using Bag-of-Words (BoW) and TF-IDF as NLP techniques. The most promising f1 score and recall metrics out of all the classifiers can be seen in Tables 2.1 and 2.2 below for both textual and URL feature extractions.

Table 2.1: Textual Feature Selection

Over-Sampling	**Classifier**	**Pre-Processing**	**Recall**	**F1 Score**
	Naïve Bayes	BoW	0.78	0.70
	Logistic Regression	BoW	0.68	0.71
	SVM	TF-IDF	0.60	0.69
Under-Sampling	Random Forest	BoW	0.74	0.57
	Random Forest	TF-IDF	0.68	0.57

Table 2.2: URL Feature Selection

Over-Sampling	**Classifier**	**Pre-Processing**	**Recall**	**F1 Score**
	Naïve Bayes	BoW	0.86	0.81
	SVM	TF-IDF	0.78	0.79

Gathering data to be used for fake news detection can be quite challenging. Although it is said that fake news is abundantly found everywhere, it is still hard to collect a large enough dataset which can be reliably used for fake news detection (Torabi Asr & Taboada 2019). In Pérez-Rosas et al.'s study we see a great example to help tackle the aforementioned problem using crowdsourcing services such as Amazon Mechanical Turk (AMT). The way AMT works is by computers posting tasks on a marketplace via an API. Employers, also known as *requesters*, can use the marketplace to post what's called Human Intelligence Tasks (HIT) for online users to fulfill in exchange for a few cents per HIT (Ipeirotis 2010).

Pérez-Rosas et al. made use of the AMT service to create a dataset by providing workers with real news articles and generate a fake version of them, this included a fake headline and a fake body. It took them approximately five days to create a dataset containing 240 fake news articles and named it FakeNewsAmt. The authors constructed a second dataset, consisting of fake news articles about public figures naturally occurring from the web and called it *Celebrities*. For preprocessing, the author carried out a 5 stage process; *Ngrams, Punctuation Removal, Psycho-linguistic Features (Deception Detection via the LIWC tool), Readability* and *Syntax*. The classifier of choice for this research was the SVM along with a five-fold-cross-validation technique to help calculate the accuracy, precision, recall, and f1 score (Pérez-Rosas et al. 2017)

A research conducted during 2017 by Wang, addresses the challenging problems of fake news, stating how problematic its consequences can be on both a social and political scale. In conjunction, it has also been stated how challenging it is to detect fake news automatically due to the limitations caused by the lack of labelled gold standard datasets. In light of this, the LIAR dataset was presented and claiming to be an order of magnitude larger than any previous datasets of similar type. All statements found in the LIAR dataset were collected from a number of broadcasting media such as; TV interviews, speeches, tweets and debates, covering a broad range of topics such as the economy, health care, taxes and elections (Alhindi et al. 2018). Wang makes use of five baselines, namely; a majority baseline, a regularized LR classifier, a SVM, a Bi-Directional Long Short-Term Memory model (Bi-LSTM), and a Convolutional Neural Network model (CNN). The best performing model was the CNN with an accuracy score of 26% followed by the least performing classifier Bi-LSTMs, achieving an accuracy score of 22% (Wang 2017).

2.4 Classifiers and NLP Techniques

Classifiers and Natural Language Processing (NLP) techniques play an important role in the Machine Learning process of evaluating, pre-processing and reading the dataset in order to correctly determine the nature of fake and real news. Hence why in the following sub-sections, an in-depth overview of the above-mentioned classifiers and NLP techniques will be explained.

2.4.1 Random Forest

As explained in the SciKit Learn documentation, a Random Forest (RF) is an estimator classifier made up of multiple Decision Tree (DT) classifiers and uses averaging to improve the predictive accuracy and control over-fitting (Pedregosa et al. 2011).

Due to their build-up, RF predictors tend to lead to a dissimilarity measure between observations made on unlabeled data (Shi & Horvath 2006). It is known that due to their computational efficiency when handling large amounts of data, RFs are commonly used to work with large datasets (Oshiro et al. 2012), therefore, it begs the question; "*do more trees mean better performance quality?*". Oshiro et al. carried out a study to determine if there is an optimal number of trees within a RF. In their research, they concluded that more trees in a RF does not equal better performance due to the fact that it requires more computational power. Therefore, one needs to asses the data that is to be used in order to determine if RF is to be applied and how many trees should it contain.

2.4.2 Decision Trees

Decision Trees (DTs) are a supervised machine learning method commonly used for classification and regression by using a large number of if-then-else decision rules. The complexity of the model is determined by how deep the tree is (Pedregosa et al. 2011). The leaves of the tree are called nodes and the starting node is called the root. Multiple nodes branch out from the root depending on how complex the tree is. After determining the output value of the root node, which is a boolean, the algorithm will either take the left or right branch, depending if the output from the root was 1 or 0, repeating this process until every leaf node is reached (Wu et al. 2015).

2.4.3 Support Vector Machine

Support Vector Machine (SVM) is a machine learning classifier used for producing the outcome of the optimal hyperplane (Battineni et al. 2019). As Battineni et al. further explains, the hyperplane is a distinctive line separating two classes, each laying on either side of the hyperplane.

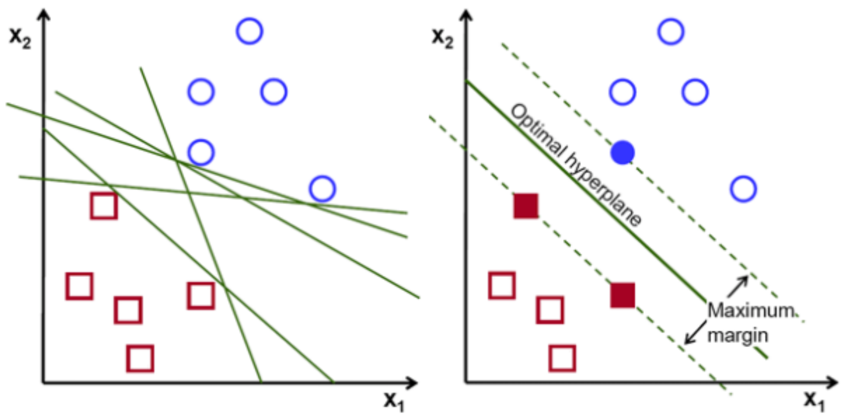


Figure 2.1: SVM Figure Example

A clear example of a simple SVM can be seen in Figure 2.1. On the left hand side, the figure is shown to have multiple hyperplanes (i.e. green lines) with different elevations and the SVM is to choose the most optimal hyperplane as shown in the graph on the right hand side of Figure 2.1 (Huang et al. 2018). A clear example of SVM being utilised for text classification can be seen in (Cușmaliuc et al. 2018). The dataset used in this paper is a collection of personally annotated tweets and the author in this paper makes use of the hyperplane to segregate the classes of the training data from Fake and Real. The algorithm will then output the most optimal hyperplane which categorises that data as Fake or Real.

2.4.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a member of the Recurrent Neural Network (RNN) family with the difference that is a bit more advanced with handling the vanishing gradient which is a known problem RNNs face. RNNs work by remembering past information but cannot remember long-term dependencies due to the vanishing gradient. LSTMs solve this problem by being more capable at remember long-term dependencies (Greff et al. 2016). The structural composition of LSTM consists of three parts known as gates, each handling an individual function[4]:

- Forget Gate - decides whether it should keep the information or not depending on the previous timestamp.
- Input Gate - quantifies the importance of the inputted information.
- Output Gate - determines the required output by understanding the text's context.

2.4.5 CountVectorizer

CountVectorizer is a tool by the scikit-learn Python library used to convert a collection of text documents to a matrix of token counts, producing a sparse representation of the counts using scipy.sparse.csr_matrix (Pedregosa et al. 2011). CountVectorizer is used when the text in question lacks semantic meaning since it simply counts the repetition of every word and replaces it with its corresponding frequency (Bogach & Kovenko 2020). In Table 2.3, (Kulkarni & Shivananda 2019) depicts how CountVectorizer works in a simple manner.

[4]https://www.analyticsvidhya.com - accessed on 12/12/2021

I love NLP and I will learn NLP in 2 months

Table 2.3: Word Frequency

I	**love**	**NLP**	**is**	**future**	**will**	**learn**	**in**	**months**
2	1	2	0	0	1	1	1	1

2.4.6 Word2Vec

Word2Vec is a word vectorizing algorithm developed by Google in 2013. It works by taking words from a large mass of text as input and learns how to output their vector counterpart[5]. Goyal elucidates this even further as he mentions how Word2Vec converts each unique word into its own multi-dimensional vector, meaning that words with similar meanings will have similar word vectors (1 vector = 300 numbers). Additionally, Goyal continues to point out how word similarity is measured by using a metric called cosine similarity, used to compare the word vectors of 2 words. A word score would represent whether or not the similarity is high (similar) or low (dissimilar), for instance “good” and “great” would have a high word score whereas “good” and “earth” would have a low word score. The author decided to use this approach to determine the tweet similarity and their respective category (Goyal 2021).

2.4.7 Term Frequency - Inverse Document Frequency (TF-IDF)

Qaiser & Ali provides us with an in depth explanation of how TF-IDF works. The term TF-IDF is made up of two words, Term Frequency (TF) and Inverse Document Frequency (IDF).

TF is used to measure the frequency of a term present in a document. The author proceeds to point out that documents with varying sizes can be problematic when calculating the term frequency in a document. To rectify this issue, the

[5]https://www.section.io/engineering-education/what-is-word2vec/ - accessed on 15/12/2021

frequency of every term has to be divided by the total number of terms. *TF = frequency of term / total number of terms*

IDF is used to assign low priority to terms with high frequency and high priority to terms with low frequency. This is because once the TF is calculated, certain words with no statistical value like "of" or "and" having a high TF, and IDF will help eliminate that. The log of IDF needs to be taken in the case of documents containing a large number of terms[6]. **N** = total number of documents and **df** = total number of terms in documents.

$$IDF = log\_e(N/df) \tag{2.1}$$

TF-IDF is simply the multiplication of the TF with the IDF (Qaiser & Ali 2018).

$$TF - IDF = TF * IDF \tag{2.2}$$

Qaiser & Ali's study mentions a text mining scenario were TF-IDF can be useful. They explain in their research how they used TF-IDF in a web browser using PHP. The user would produce a document via the web browser as a source of textual data which then would be preprocessed. A word count of the corpus is then obtained and check the word frequency for each word, followed by the application of the TF formula. The total number of documents is then counted and each word is checked to see if it is present in all documents. The IDF formula is then applied followed by the TF-IDF. The program will then output the serial number, word, word frequency, TF, IDF and TF-IDF which is the target variable of this study (Qaiser & Ali 2018).

Another scenario were TF-IDF was proven to be useful can be seen in (Marcos-Pablos & García-Peñalvo 2020). The authors methodology investigates the possibility of delivering a solution for researchers to alleviate redundant data by retrieving

[6]https://towardsdatascience.com tfidf from scratch in python - accessed on 04/02/2022

relevant information from scientific electronic databases with the help of TF-IDF. Studies such as this will create a more accessible environment for researchers who's job is vital in gathering as much reliable data as possible.

2.5 Chapter conclusion

By reviewing the above results obtained by other studies and observing the amount of times the same classifiers were used for multiple researches, justifiably, the most common classifiers yielding the best results are SVM, LSTM and DT. An in-depth review of some of the setups used will be made in the next chapter, outlining some of the best machine learning practices and a further discussion of their application.

Chapter 3

Research Methodology

This chapter will demonstrate in further detail studies related with this paper's research and their methodology. Additionally, it will also showcase an in depth overview detailing the application process of this research study which will go over a number of things, such as; the dataset, quantitative and qualitative, research pipeline and the model construction's approach.

3.1 Related Studies

In Section 2.3, the Random Forest (RF) and Decision Tree (DT) classifiers were mentioned in various studies, were both classifiers were applied to a dataset consisting of 20,800 articles obtained from various articles found on the internet. Since the articles in the dataset contain irrelevant information like stop words, such as *"a"*, *"be"* and *"should"* the data requires preprocessing by removing any redundancies which can effect the model's accuracy, there the dataset is reduced to exclusively contain significant information.

During their research of 2019, Alam & Yao carried out a study to investigate the correlation between preprocessing and accuracy performance with the use of three learning classifiers; Naive Bayes, Maximum Entropy (ME), and Support Vector Machine (SVM). The results where then calculated before and after preprocessing, indicating that preprocessing more often than not improves accuracy, with Naive Bayes showing considerable improvements after preprocessing by increasing its accuracy score from 84% to 92% (Alam & Yao 2019).

The preprocessing phase is commonly made up of a multi-step procedure which includes the removal of any stop-words, punctuation, case-sensitive characters, white space, special characters and digits. A combination of the previously mentioned steps, depending on the contents of the dataset, can be adopted; for example (Jehad & Yousif 2020) integrated all of the aforementioned steps whereas (Vijayaraghavan et al. 2020) only incorporated the removal of numbers, punctuation, special characters and stop words. As a result of preprocessing, the number of articles decreased to 20,761 (10,432 real and 10,432 fake). For testing the classifiers, the authors made use of two of the dataset's features; the text (the preprocessed content of the article) and the label (1 or 0 for real or fake, respectively). Subsequently, the same research outlined the implementation of the Feature Selection (FS) process which is carried out through the TF-IDF vectorizer to filter out any irrelevant features (Jehad & Yousif 2020).

During their research, Vijayaraghavan et al.'s research, defined their application using five classifiers; Support Vector Machine (SVM), RF, Logistic Regression (LR), Long Short-Term Memory (LSTM) and Artificial Neural Network (ANN) along with three pre-training models; CountVectorizer (CV), WordToVectorizer (W2V) and TF-IDF on a dataset of 20,800 records. With the help of Parts-of-Speech (PoS) Distribution, the authors managed to determine the percentage of adverbs, adjectives, nouns, verbs and proper pronouns. The role of PoS is to determine the proper PoS tag of each word in a text, which is the fundamental task of various Natural Language Processing applications such as machine translation, information retrieval/extraction and sentiment analysis (Thavareesan & Mahesan 2020). Finally, an Isolation Forest (IF) algorithm is performed to remove outliers on the features generated from the TF-IDF, CV and W2V and a percentage of outlier removal was calculated (Vijayaraghavan et al. 2020).

As for the yielded results they are illustrated in Tables 3.1 and 3.2 obtained from (Jehad & Yousif 2020) and (Vijayaraghavan et al. 2020), respectively. Table 3.1 shows the DT classifier outperforming RF in both before and after preprocessing. It is important to notice the significant increase in accuracy of the 'after preprocessing' result, which further supports the previously made claim on the importance of preprocessing. Table 3.2 evidently depicts the best performing classifier being LSTM with CV, having a mean test score of 94.88%. The least performing classifier was RF with CV and TF-IDF, both yielding a mean test score of 87.64%. It is interesting to notice that the RF results with TF-IDF achieved in (Vijayaraghavan et al. 2020) are similar to the ones achieved in (Jehad & Yousif 2020) as illustrated in table 3.1, since they both make use the same classifier and NLP technique, but followed different preprocessing techniques.

Table 3.1: Test Scores

	Random Forest	**Decision Tree**
After Preprocessing	84.79%	89.11%
Before Preprocessing	73.07%	78.13%

Table 3.2: Mean Test Scores

	SVM	ANNs	LSTMs	LR	RF
CV	93.06%	94.29%	**94.88%**	94.45%	87.64%
TF-IDF	94.58%	93.73%	93.89%	**94.79%**	87.64%
W2V	91.17%	**93.06%**	92.29%	91.30%	88.60%
Average	92.94%	93.69%	93.69%	93.51%	87.96%

An average accuracy for each classifier was calculated, indicating that the best performing classifiers were ANN and LSTM, both having an average mean test score of 93.69%. With that being said, the only drastic difference in accuracy when comparing all five models was that of the RF, the difference in accuracy of the other four models was very marginal.

During 2020, a group study was conducted by the University of Bangladesh on Bangla news articles (Hussain et al. 2020), by scraping various Bangla newspapers to construct a fake news dataset based on the writers' native language. The dataset is made up 2541 articles (1548 real and 993 fake). The proposed classifiers were the SVM with a linear kernel and Multinomial Naive Bayes (MNB), using CV and TF-IDF as NLP techniques. Similar to other studies, the authors in (Hussain et al. 2020) carried out a preprocessing method to remove any special characters from the text. A writer for the notorious artificial intelligence website 'insideBIGDATA' by the name of Daniel Gutierrez wrote an article in 2013, supporting the idea of distinctly splitting the dataset into training and testing sets, which can make a substantially positive impact on the accuracy results. Guiterrez also makes mention of how an Australian medical research team (Skafidas et al. 2014) trained and tested their risk classifier on the same group of people. Guitarrez stated that this can be seen as bad practice as it 'contaminates' the model with information about the test set[1]. Additionally, a team study led by a member from the Massachusetts General Hospital by the name of Benjamin Neale, replicated the Australian group's research on a larger sample and discovered that the study conducted by Skafidas et al. does not accurately predict ASDs[2]. Hussain et al. follows with a similar type of procedure in their research by randomly splitting the dataset into two sets; 70% for testing and 30% for training. The reason a linear kernel was integrated into the SVM was because it is preferred for text classification since most text is linearly separable. The results yielded indicate that SVM with linear kernel (96.64%) performed marginally better than MNB (93.32%) (Hussain et al. 2020) and in comparison, outperforming the SVM found in (Vijayaraghavan et al. 2020), albeit the difference is rather minimal.

3.2 LIAR Dataset

Three datasets were investigated and considered as suitable candidates, eventually choosing LIAR to be the most applicable for this research. The other two datasets

[1]https://insidebigdata.com – *The importance of training and testing* – accessed on 22/02/2022

[2]https://www.the-scientist.com - genetic test for autism refuted - accessed on 22/02/2022

were FakeNewsNet and RealNews. FakeNewsNet was being considered because it was being claimed to have the potential to boost studies with open research problems related to fake news as well as being vetted for its legitimacy by *Politifact*[3] (**P** for short) and *Gossipcop*[4] (**G** for short) (Shu et al. 2018). As for RealNews, it was being considered due to its large corpus of news articles obtained from 5000 domains indexed by Google News via a scraper scraping data dumps from *Common Crawl* (Zellers et al. 2019). LIAR, the dataset used in this research was obtained from paperswithcode.com and is a publicly available dataset containing 11,505 manually labeled statements made up of different contexts from the past decade, verified by **P** and was last updated in 2019. The labeling of the LIAR dataset is a bit different from the more conventional type of labeling commonly found in fake news datasets, which normally assigns 1 or 0 depending on whether the article is fake or real. Instead, with LIAR the label feature consists of six types of unique labels rather than two, with each label representing the type of news, which are; true, mostly true, half true, barely true, false and pants on fire. Table 3.3 shows two tables for Training and Testing, outlining the number of articles under each label, while Figures 3.1 and 3.2 display its visual representation. The reason LIAR was selected over the other two previously mentioned datasets was because of the larger number of statements it consists, along with its unconventional method of labeling, which is believed to provide more accurate results over the more customary method of 1's and 0's as seen in the results obtained by (Yang et al. 2019).

[3]https://www.politifact.com
[4]https://www.suggest.com

Table 3.3: LIAR Label Count

(a) Train

Type of News	**Count**
true	1676
mostly true	1962
half true	2114
barely true	1654
false	1995
pants on fire	839

(b) Test

Type of News	**Count**
true	208
mostly true	241
half true	265
barely true	212
false	249
pants on fire	92

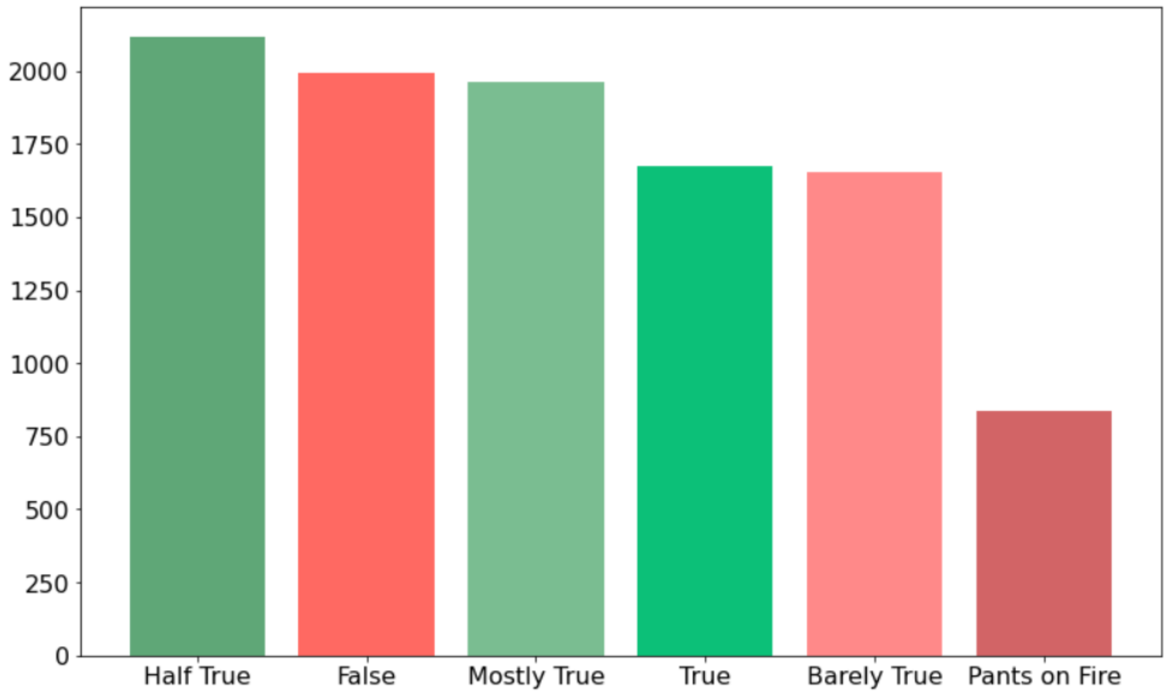


Figure 3.1: Train Label Distribution

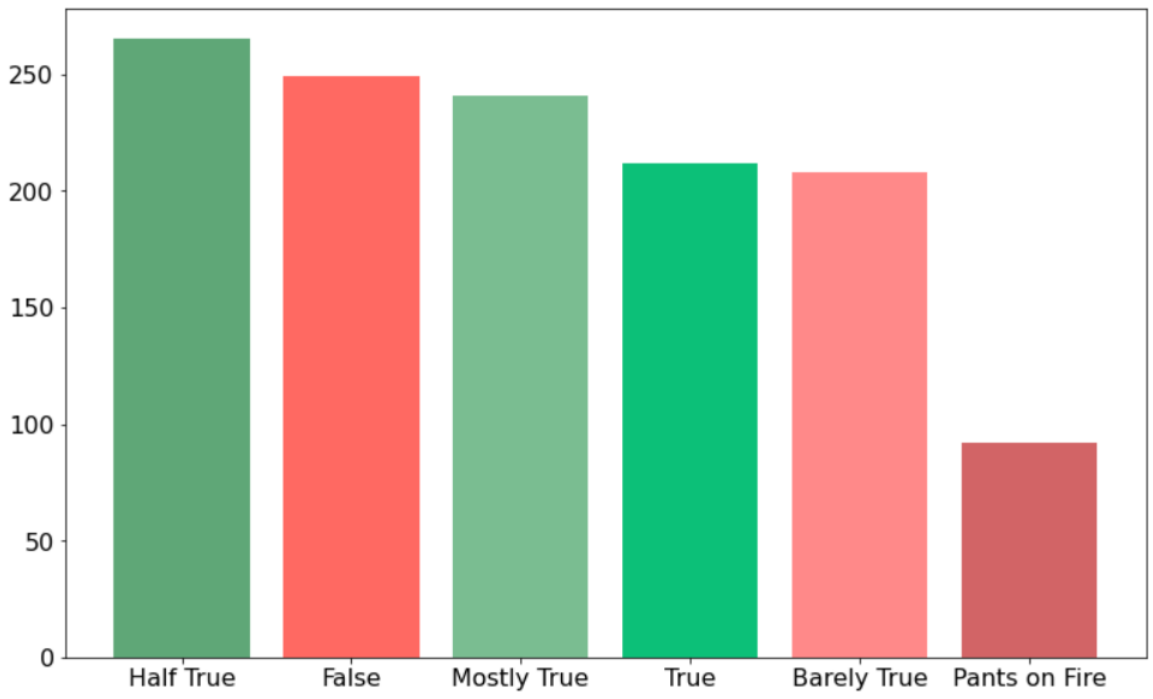


Figure 3.2: Test Label Distribution

3.3 Political Dataset

Despite LIAR being the ideal dataset for this research, ultimately it still failed to align with its political scope since it contained statements outside the political field. Following a thorough dissection of the dataset, it was found to be made up of mostly political statements. This brought around the opportunity to construct a political dataset without the need to heavily alter LIAR itself. The first step was to identify which attributes can be used to help filter out non political statements, which were found to be *Subject* and *Party*. Observably, 1,958 statements in the *Party* attribute didn't have an affiliation with a particular party and were labeled as 'none', therefore leading to their exclusion from the dataset. Using the *Subject* attribute, certain keywords such as 'sports', 'environment', 'energy' and 'education' were identified and statements associating with said keywords were also excluded from the dataset. Understandably, some of the mentioned keywords can have a political sense but upon reading their affiliated statements they were found to have lacked any political context. With this in mind, it still cannot be said that this dataset is purely political since the statements were not reviewed individually, but it has enough political alignment to help justify the hypothesis.

3.4 Nature of Study

It is important to acknowledge the context of the nature the research is being conducted in, in order to understand what it is trying to achieve. Outlined below are two definitions stating different types of natures that define a study as well as pointing out which category this type of research falls under.

3.4.1 Qualitative and Quantitative

Qualitative research helps to understand the human condition in a different context of a perceived situation, depending on the resources, time and effort researchers are willing to invest in trying to understand a study's phenomena (Bengtsson 2016). This type of research takes an inquiry approach at exploring key concepts of a study, gathering information by asking participants a set of questions (broad and general) to obtain detailed opinions in the form of words or images (Creswell & Clark 2004).

Quantitative research makes use of numbers and statistics to systematically measure variables and hypothesis[5]. It uses content validity, meaning whether the instrument covers all the content in respect with the variable and content reliability, which means how consistent the study is in its results when being executed repeatedly with the same parameters. Another important component needs to be factored in while conducting a quantitative study and that is rigour. Rigour reflects the work put into the research to enhance the quality of the study. This is achieved through evaluating the validity and reliability of the tools utilised in the study (Heale & Twycross 2015).

Examining the definitions of both quantitative and qualitative studies, it is

[5]www.scribbr.com - Qualitative vs Quantitative Research - accessed on 08/03/2022

clear that this research study falls under the quantitative category. This is because this research relies heavily on statistical and numerical data in every aspect of its construction, from cleaning the dataset, building the models to delivering the achieved observations, it is all dependant on the numerical data being manipulated through computational techniques[6].

3.4.2 Inductive and Deductive Reasoning

Inductive reasoning, also known as the “bottom up” approach moves from specific observations to broader generalisations and theories. This type of approach begins with specific observations, detects patterns and regularities, formulates exploratory hypothesis and finally develops general conclusions and theories[7]. Inductive reasoning is the way how people reason on a day to day basis, for example the possibility of raining tomorrow since induction works by predictions about novel objects or situations based on existing knowledge (Hayes & Heit 2018).

Deductive reasoning is the logical process of forming a general idea and it progresses to a specific, proven conclusion. A common argument in deductive reasoning often begins with a premise, followed by another premise to support the initial one and finally forming a conclusion based on the two premises, this is called “premise-premise-conclusion”[8].

Upon reviewing both definitions of inductive and deductive reasoning, it outlines this research falling under the deductive category. This is because this paper follows the same approach as the one in the aforementioned paragraph, where a general idea is generated, followed by a hypothesis and then forming a conclusion supported by multiple sources.

[6]https://libguides.usc.edu/writingguide/quantitative - accessed on 10/03/2022
[7]www.conjointly.com - Deduction and Induction - accessed on 14/03/2022
[8]www.scribbr.com - deductive reasoning - accessed on 15/03/2022

3.5 Prototype Implementation

The prototype for this research was implemented using the Python programming language as well as various machine learning libraries which can be found in Appendix A. The hardware used to execute the models found in this research can be seen in the following list:

- CPU: AMD Ryzen 5 3600X 6-Core Processor 4.25 GHz
- RAM: 16GB 3000MHz
- GPU: Geforce RTX 2070

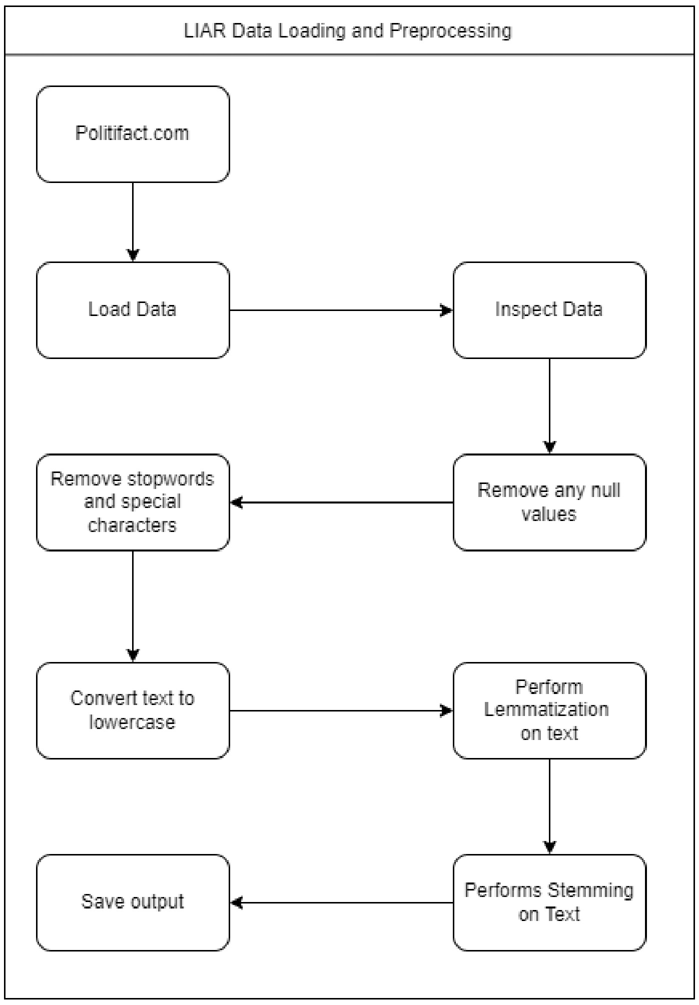
Throughout this research, a total of twelve models were constructed with the aim of identifying which one is the most effective in detecting fake news. The models are made up of four classifiers which are; Random Forest (RF), Decision Trees (DTs), Support Vector Machine (SVM) and Long Short-Term Memory (LSTM) along with three Natural Language Processing (NLP) techniques, namely; Term Frequency - Inverse Document Frequency (TF-IDF), CountVectorizer (CV) using the TF-IDF Transformer and Word2Vec (W2V). To elucidate better the process of the model construction, the following subsections will describe in detail the multiple steps taken towards the completion of these models up until the point of their fruition.

3.5.1 Pipeline

The following figures will show the process pipeline taken into constructing the models. Each step will be explained in detail in the following subsections. The following explanation will explain further the process happening in Figure 3.5. The primary dataset, which is LIAR was obtained from politifact.com[9] and was already split into training and testing sets. Both sets were then preprocessed (see

[9]https://www.politifact.com

Figure 3.3) and combined as one dataset. To create the political dataset, the LIAR was filtered from non political statements and then preprocessed (see Figure 3.4) and also combined as one dataset.


LIAR Data Loading and Preprocessing
Politifact.com
Load Data
Inspect Data
Remove any null values
Remove stopwords and special characters
Convert text to lowercase
Perform Lemmatization on text
Performs Stemming on Text
Save output
Figure 3.3: LIAR dataset Preprocessing Procedure

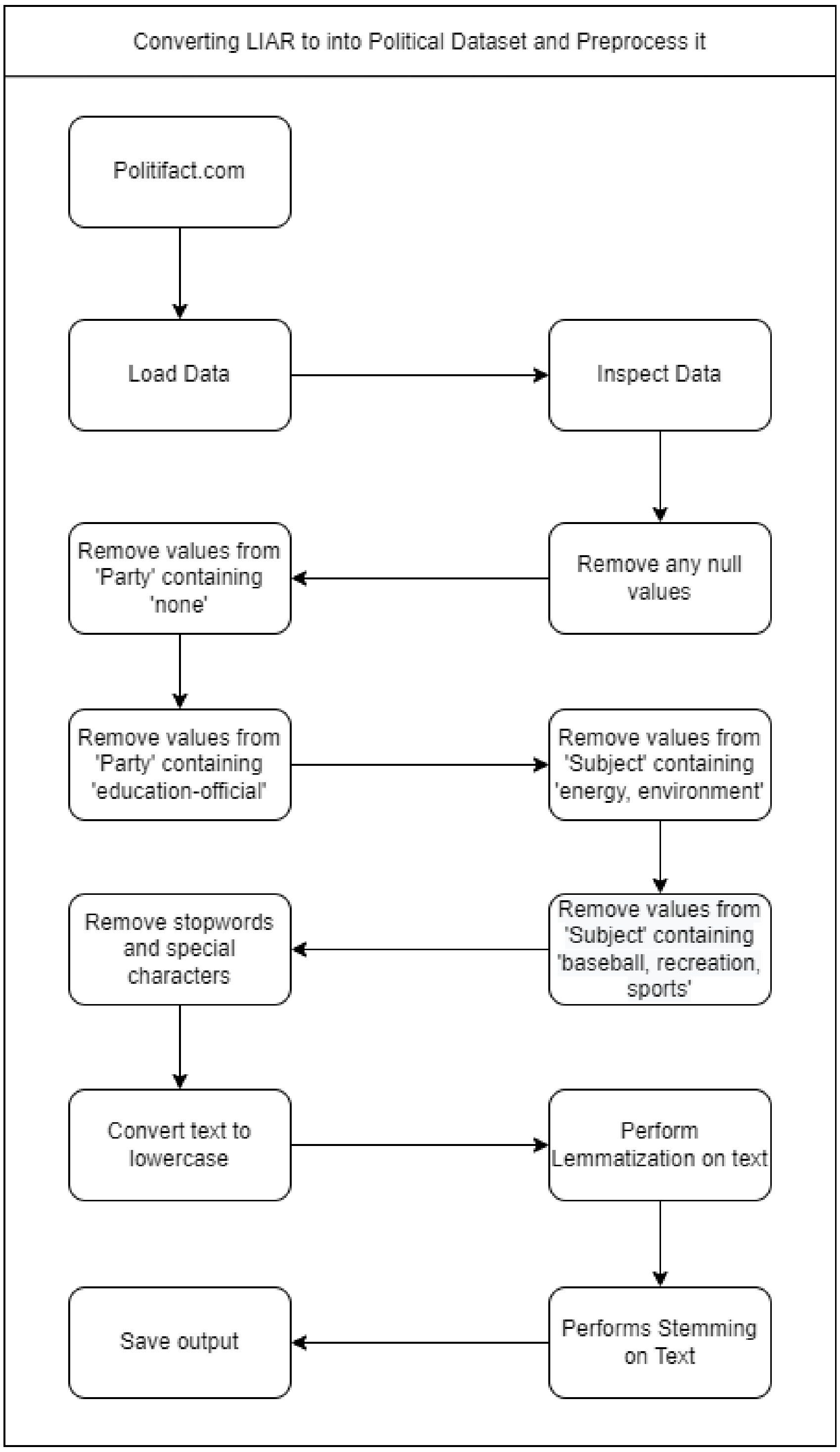

Converting LIAR to into Political Dataset and Preprocess it
Politifact.com
Load Data
Inspect Data
Remove any null values
Remove values from 'Party' containing 'none'
Remove values from 'Party' containing 'education-official'
Remove values from 'Subject' containing 'energy, environment'
Remove values from 'Subject' containing 'baseball, recreation, sports'
Remove stopwords and special characters
Convert text to lowercase
Perform Lemmatization on text
Performs Stemming on Text
Save output

Figure 3.4: Political dataset Preprocessing Procedure

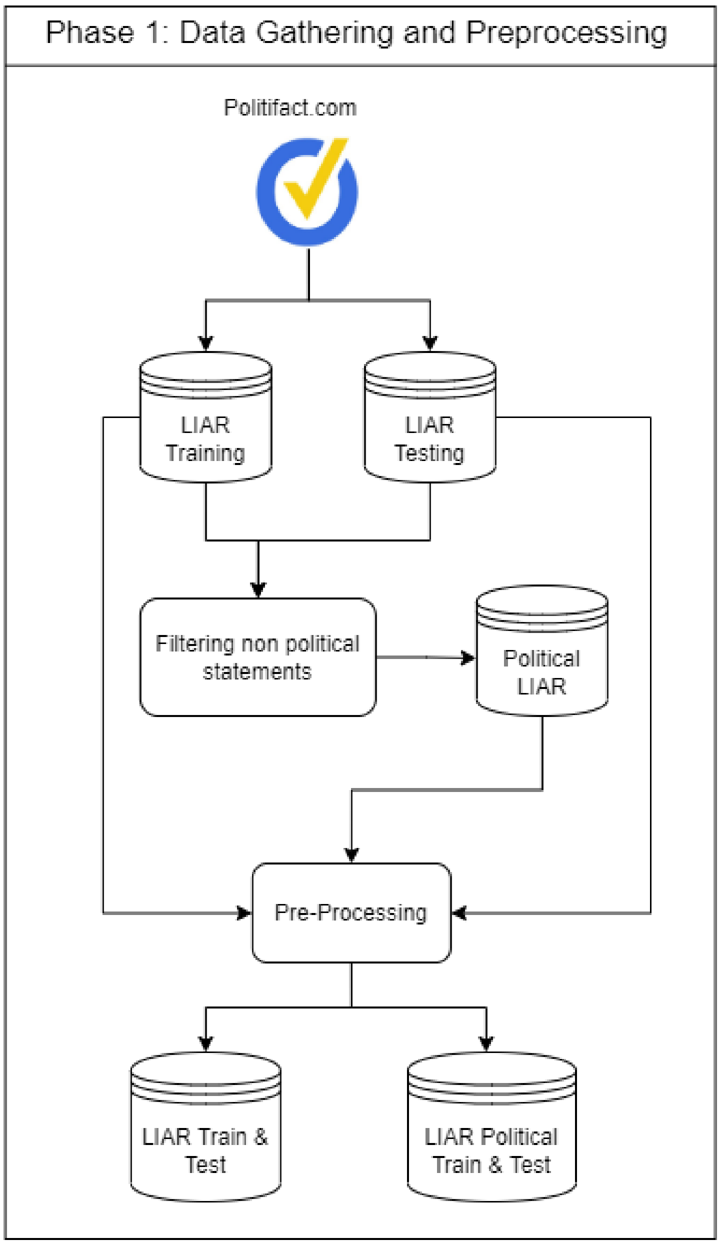

Phase 1: Data Gathering and Preprocessing
Politifact.com
LIAR Training
LIAR Testing
Filtering non political statements
Political LIAR
Pre-Processing
LIAR Train & Test
LIAR Political Train & Test

Figure 3.5: Data Gathering & Preprocessing

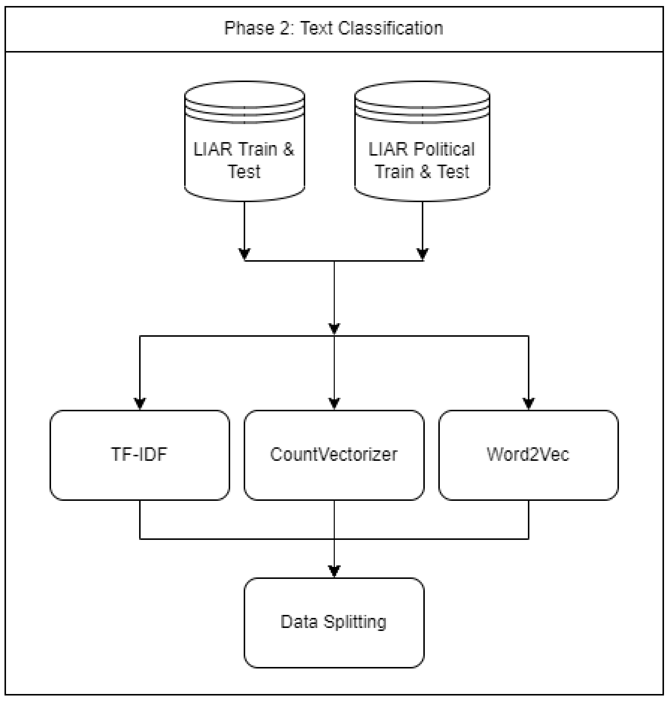


Figure 3.6: Text Classification

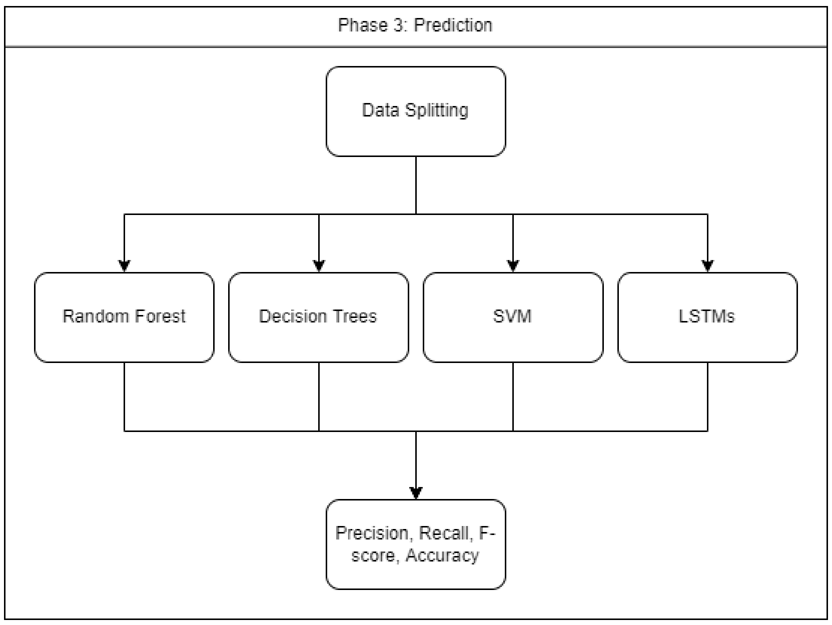


Figure 3.7: Prediction

3.5.2 Data Loading and Preprocessing

When loading both the LIAR dataset and the political LIAR dataset, it is important to inspect its contents for any irregularities that might make the data prone to errors such as null values. Multiple null values were found across all attributes in the LIAR dataset but for this particular study we are interested in the ones present in the '*ID*', '*Statement*' and '*Label*' attributes. The next step would be to preprocess the data by lammetizing the text via the nouns and verbs, stemming, removing special characters and stop words and converting everything to lowercase. The stop word removal process was done by using the NLTK library to import the Stop Words while using the Gensim library to execute the preprocessing by comparing the imported Stop Words with the given corpus and deleting any words that match the Stop Words and any words with a character length of less than 3. The Matplotlib library was used to display Figure 3.9, outlining the average word length for before and after preprocessing. Moreover, the Seaborn library was used to plot a Bubble plot defining the word count as shown in Figure 3.8. This depicts the top 10 words with highest frequencies in the entire corpus. Additionally, it is important to point out that this process was carried out after successfully preprocessing both the Train and Test sets.

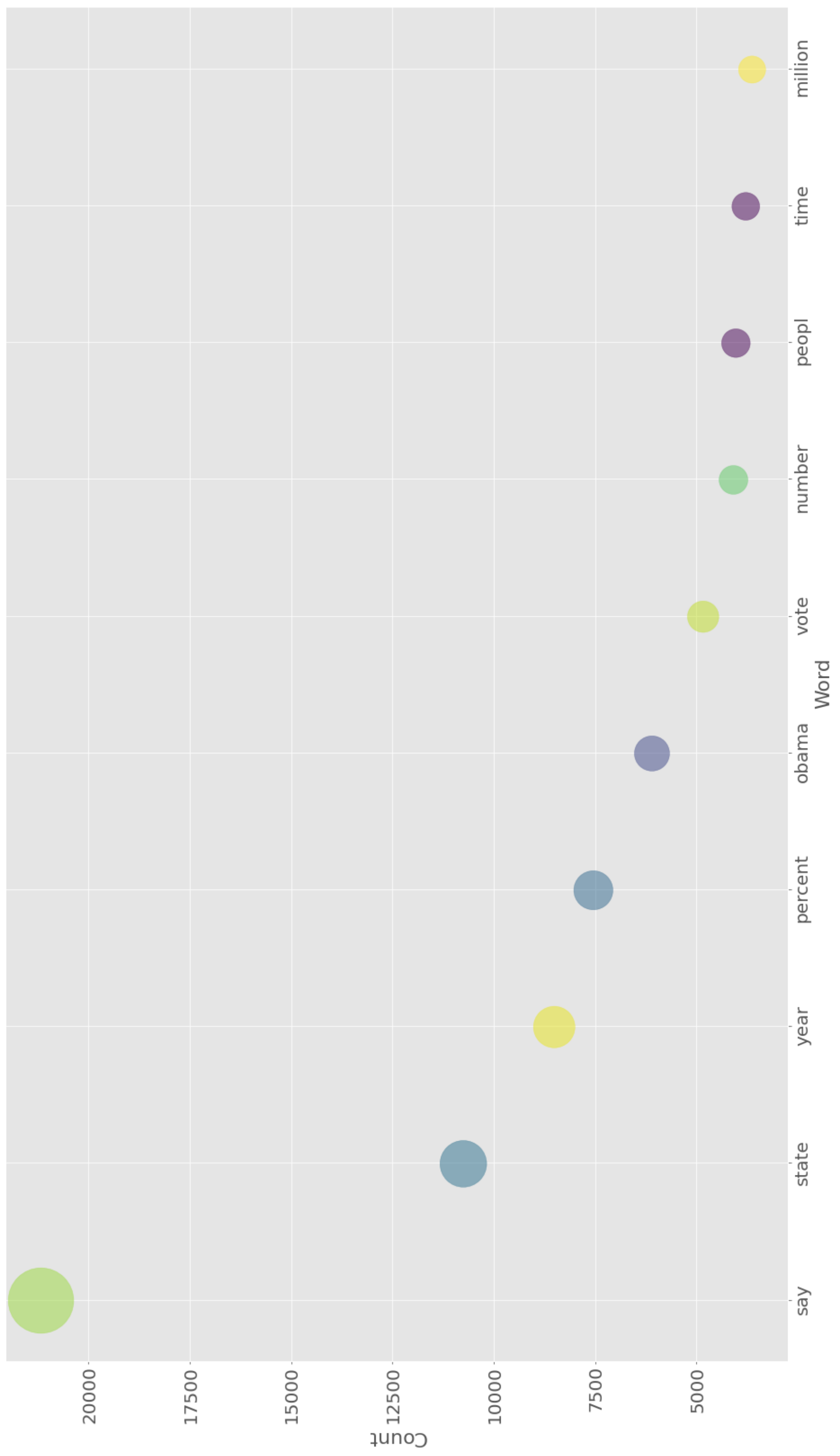


Figure 3.8: Top 10 Words with the Highest Frequencies

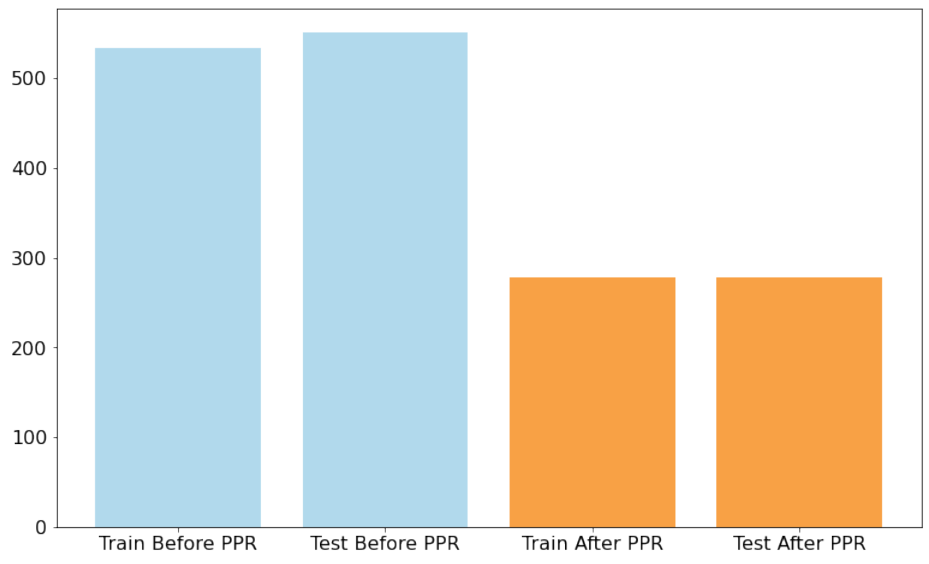


Figure 3.9: Before and After Preprocessing (PPR)

3.5.3 Model Construction & Experimental Setup

As previously mentioned in Section 3.5 the NLP techniques seen is this study are CV utilising the TF-IDF Transformer, TF-IDF and W2V. These techniques were chosen due to their seemingly good results observed throughout the interim of this research. Subsequently, four classifiers were implemented to process the output given by the NLP techniques to ultimately form our models. This approach was also applied uniformly to the political dataset. The following subsections will explain in detail the steps taken into completing these models.

3.5.4 Random Forest

Each of the vectorised output received from the NLP techniques was fed into the RF classifier, conducting four experiments for each, changing the test size, n-jobs and n-estimators with each iteration. The following information is based upon data obtained from (Pedregosa et al. 2011):

- Test Size - a portion of the dataset the model uses as testing, measuring in this case the fake news detection accuracy. The train set determines the size of the test set, meaning that if the train size is 70%, the test size would be 30%.

- N-Estimators - the number of trees Random Forest generates and chooses the most optimal one out of the N generated trees.
- N-Jobs - determines the number of jobs running in parallel. 1 meaning no jobs and -1 meaning there aren't any limits and the model is utilising all the processors, significantly decreasing the execution time.

Table 3.4: RF Hyperparameters for CV, TF-IDF and W2V

Experiment No.	**Test Size**	**N-Estimators**	**N-Jobs**
1	30%	100	1
2	20%	250	1
3	25%	300	-1
4	35%	350	-1

Table 3.4 shows the hyperparameters for the three RF models.

3.5.5 Decision Trees

The same process of RF was incorporated for DTs, obtaining all three NLP technique outputs and inputting them into the DT classifier. As for the hyperparameters they were left at their default settings with changes made only to the test size.

3.5.6 Support Vector Machine

For SVM, the implementation process was also similar to that of the previous two classifiers with the difference that it had a few hyper-parameters that required tuning. The hyper-parameters in question are the Kernel, Gamma and C which were changed over the course of four experiments with the aim of finding the most optimal combination of hyper-parameters. The following information seen in the list below was obtained from (Pedregosa et al. 2011), techvidvan.com, dataspirant.com and data-flair.training.com.

- Kernel - a function to help solve problems efficiently even at higher dimensions. Kernels help form the hyperplane without raising the complexity. There are five kernel types in total, namely; linear, poly, rfb, sigmoid and precomputed. Out of the five kernel types three were selected for the purpose of this research since they are the most relevant;
  - RFB - this is a general purpose kernel type, commonly used when there is a lack of prior knowledge about the dataset. It is the default kernel type if none were selected and It was chosen to observe the model's performance with its default configuration. Equation is:
    $$k(x, y) = exp(-\frac{||x-y||}{\sigma})$$
  - Linear - commonly used in text classification and when dealing with large sparse data vectors. It was chosen since it is the most relevant kernel type to this study due to its text classification capabilities. Equation is:
    $$F(x, xj) = sum(x * xj)$$
  - Sigmoid - commonly used in neural networks due to its neuron activation function. It was chosen to observe its performance against the other neural network (NN) model seen in this study, the LSTM. Equation is:
    $$F(x, xj) = \tanh(\alpha xay + c)$$
- Gamma - this makes the model very sensitive, meaning that if the gamma is too big the radius of the area of the support vectors will only include the support vector itself which can lead to C unable to regulate overfitting, whereas if it's too small the model will not be able to capture the complexity of the data.
- C - is a regularization parameter which tries to accommodate between correct

classification and maximization of the decision function's margin. Changing this parameter will move the margin to prioritize one of the two options, up in favor of accurate classification and down in favour of having simpler decision function.

Table 3.5 shows the hyperparameters for the three SVM models. The parameters shown in experiment 1 are set to their default values.

Table 3.5: SVM Hyper-parameters for CV, TF-IDF and W2V

Experiment No.	**Test Size**	**Kernel**	**C**	**Gamma**
1	30%	RFB	1	Scale
2	20%	Sigmoid	0.5	Auto
3	25%	Linear	0.5	Auto
4	35%	Linear	0.5	Scale

3.5.7 Long Short-Term Memory

Since LSTM is a recurrent neural network, the approach taken to construct it was quite different than any of the previous models. After the integration of the CV, TF-IDF and W2V a Word Tokenizer from the Keras TensorFlow library was implemented. Initially, the first LSTM model was executed on the base formula of hyperparameters which was producing a high amount of overfitting due to the 350 max features being inputted. This was reduced to 6 max features, drastically improving overfitting. Some fine tuning was required since the results were still indicating some fluctuation in performance. The major contributor to this fluctuation was the batch size, by reducing its value to 3 it was producing more consistent results. Some additional fine-tuning was made towards the input and output dimensions, setting them to 10 and 50 respectively. The optimizer function has been set permanently as *Adam* since it has been known to be the best overall optimizer[10]. Through multiple iterations it was determined that the epoch value is set to 200 since the model stopped showing any improvements beyond that

[10]https://deepdatascience.wordpress.com - accessed on 5/05/2022

point. A total of 4 layers were implemented for all 3 models starting from the Embedding Layer, Spatial Dropout 1D, the LSTM layer and finally the Dense layer. Four experiments were carried out changing different hyperparameters with every experiment. The following bulleted list will explain how the LSTM models were utilised in this scenario.

- Embedding Layer - takes 3 parameters, the input dimension, output dimension and the input length [11].
- Spatial Dropout 1D - for the dropout layer, SpatialDropout1D with a value of 0.2 was used rather than a regular dropout because it helps promote independence between feature maps which is known to help regularise activations[12].
- LSTM Layer - for the LSTM layer a unit value of 25 was given since it represents the dimensionality of outer space.
- Dense Layer - the Dense layer was given a value of 6 since our model consists of 6 classification labels.

Table 3.6 will represents the hyperparameters that were changed for each experiment which are the test size and activation.

Table 3.6: LSTM Hyper-parameters for CV, TF-IDF and W2V

Experiment No.	**Test Size**	**Activation**
1	30%	Relu
2	20%	Sigmoid
3	25%	Tanh
4	35%	Relu

[11]https://machinelearningmastery.com
[12]https://docs.w3cub.com

Chapter 4

Analysis of Results & Discussion

This chapter will analyse and discuss in detail the results that the models in Section 3.5.3 have procured. It will show the different types of results achieved by each model and what configurations were used for said model. Additionally, by comparing and contrasting the major results each model has yielded, it will determine the best fake news detection model in this scenario. A number of graphs and tables were also used to help visualise the results.

4.1 Analysing Results

The first section will be divided in subsections, each discussing a classifier and the results each model achieved pertaining to said classifier. The results in this section will be displayed in a tabular format and the list below will explain the meaning of the table headings for each model.

The following list of table headings are the ones which are common for all tables:The next list will outline the table headings that are exclusive to certain classifiers:

- Test Size - this symbolises the size of the data the model will be tested with.
- Prec - this is the Precision and it calculates to the quality of the performance

of the model's positive prediction.

$$Precision = TP/(TP + FP)$$

- Recall - calculates the amount of items that were properly predicted out of the entire dataset.

$$Recall = TP/(TP + FN)$$

- F-score - represents the harmonical balance between Precision and Recall and calculates the average between the two.

$$F1Score = 2 * (Recall * Precision)/(Recall + Precision)$$

- Acc - this refers to the Accuracy and calculates the ratio of the correctly labeled items out of the entirety of the dataset.

$$Accuracy = (TP + TN)/(TP + FP + FN + TN)$$

The next list will outline the table headings that are exclusive to certain classifiers:

- Random Forest
  - N-Jobs - determines the number of jobs running in parallel.
  - N-Estimators - this represents the number of trees Random Forest generates and chooses the most optimal one out of the generated trees.
- Support Vector Machine
  - Kernel - this will help form the hyperplane without raising the complexity.
  - C - increasing this parameter's value will favour accurate classification and decreasing it will favour having a simpler decision function.
  - Gamma - this changes the size of the area of how many support vectors it is trying to include.

- Long Short-Term Memory
  - Optimizer - a function that affects the model's attributes such as weight and learning rate, therefore reducing the overall loss and improves the accuracy[1].
  - Activation - a function in neural network to define how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network[2].
  - Batch Size - number of sample items is sent to the model at a time[3].

4.1.1 Random Forest

The initial experimental setup for Random Forest (RF) was executed on the original preprocessed LIAR dataset, which only consisted of the removal of stop words, punctuation and converting the text to lowercase. The results seen in Table 4.1 indicate an under-performance in precision, recall, f-score and accuracy.

Table 4.1: RF with TF-IDF before Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.2432	0.1775	0.1640	20.37
20%	250	1	0.2676	**0.1842**	0.1640	**21.23**
25%	300	-1	**0.2919**	0.1787	0.1570	20.82
35%	350	-1	0.2551	0.1839	**0.1670**	21.09
Average			0.2645	0.1811	0.1630	20.88

In light of this, the dataset was preprocessed further by performing lemmatization. Results of this procedure can be seen in Table 4.2. The average difference of precision decreased from **0.2645** to **0.2264**, recall, f-score and accuracy has very marginal differences.

[1]https://www.analyticsvidhya.com - accessed on 13/05/2022
[2]https://www.v7labs.com - accessed on 13/05/2022
[3]https://www.analyticsvidhya.com - accessed on 13/05/2022

Table 4.2: RF with TF-IDF after Lemmatization

Test Size	**N-est**	**N-jobs**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	100	1	0.2206	0.1786	**0.1644**	20.94
20%	250	1	0.2037	0.1835	0.1630	21.29
25%	300	-1	**0.2428**	**0.1862**	0.1600	21.04
35%	350	-1	0.2384	0.1846	0.1620	**21.51**
Average			0.2264	0.1832	0.1624	21.19

Seeing how there was a slight decrease in performance after applying lemmatization, another experiment was carried out applying only stemming. Table 4.3 displays these results depicting a minimal precision decrease from that of Table 4.2, as for recall, f-score and accuracy the value remained more or less the same.

Table 4.3: RF with TF-IDF after Stemming

Test Size	**N-est**	**N-jobs**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	100	1	0.2028	0.1815	**0.1656**	20.83
20%	250	1	0.2326	**0.1915**	0.1644	**21.85**
25%	300	-1	**0.2366**	0.1831	0.1634	21.35
35%	350	-1	0.2026	0.1867	0.1632	21.35
Average			0.2187	0.1857	0.1641	21.34

The results still appeared to be indicating some under performance therefore another attempt at some further preprocessing was made by performing stemming with lemmatization as seen in Table 4.4. Observing the average of the yielded results when compared with Table 4.3, it is noticeable that the average precision decreased from **0.2264** to **0.2187** whilst the recall increased from **0.1832** to **0.1857** and f-score increased from **0.1624** to **0.1641** while the accuracy had a very marginal increase. Additionally, it can be deduced that the best performing experiment of Table 4.4 is the second one since it has the highest values in precision, recall, f-score and accuracy. This could possibly mean that it has the most ideal combination of n-estimators and test size.

Table 4.4: RF with TF-IDF after Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.2096	0.1762	0.1600	20.20
20%	250	1	**0.2279**	**0.1828**	**0.1630**	**21.23**
25%	300	-1	0.2199	0.1787	0.1590	21.04
35%	350	-1	0.2156	0.1806	0.1540	21.15
Average			0.2182	0.1796	0.1590	20.91

The above results were all obtained from the RF model with TF-IDF. Tables 4.5 to 4.8 will outline the results yielded from the RF with the CountVectorizer (CV) NLP technique. The average precision decreased from **0.2476** to **0.2148** after lemmatization and increased again to **0.2419** after stemming was performed. Interestingly, a similar pattern can be seen in Tables 4.5 and 4.6 as that of Table 4.4, where the second experiment performed better than others. Table 4.8 shows the CV model results after lemmatization with stemming, showing a decrease in the average precision after stemming but remained the same as that seen in after lemmatization. As for the average recall, f-score and accuracy the difference was extremely minimal across all four tables.

Table 4.5: RF with CV before Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.1639	0.1788	0.1640	20.70
20%	250	1	**0.3050**	**0.1940**	**0.1730**	**22.10**
25%	300	-1	0.2420	0.1868	0.1680	21.47
35%	350	-1	0.2793	0.1854	0.1600	21.17
Average			0.2476	0.1863	0.1663	21.36

Table 4.6: RF with CV after Lemmatization

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.1953	0.1783	0.1660	20.78
20%	250	1	0.2296	**0.1889**	**0.1690**	**21.86**
25%	300	-1	**0.2304**	0.1879	0.1590	21.58
35%	350	-1	0.2039	0.1797	0.1570	20.82
Average			0.2148	0.1837	0.1628	21.26

Table 4.7: RF with CV after Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.2533	0.1838	0.1661	21.00
20%	250	1	0.1787	**0.1892**	0.1652	**22.15**
25%	300	-1	**0.3118**	0.1880	**0.1716**	22.11
35%	350	-1	0.2239	0.1792	0.1535	20.82
Average			0.2419	0.1851	0.1641	21.52

Table 4.8: RF with CV after Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.2113	0.1823	**0.1680**	21.19
20%	250	1	0.1675	0.1815	0.1560	20.85
25%	300	-1	0.2322	0.1821	0.1590	21.01
35%	350	-1	**0.2571**	**0.1877**	0.1650	**21.85**
Average			0.2170	0.1834	0.1620	21.23

Tables 4.9 to 4.12 will now exhibit the RF models with Word2Vec (W2V). The results depict a poor performance when compared with the previous models. Average precision values start at **0.2068**, drop to **0.1778** after lemmatization and remain at a similar value in both Tables 4.11 and 4.12. The average recall, f-score and accuracy have minimal differences across all the experiments seen from Tables 4.9 to 4.12.

Table 4.9: RF with W2V before Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.1966	0.1772	0.1669	20.47
20%	250	1	0.1791	0.1891	**0.1680**	21.92
25%	300	-1	**0.2415**	**0.1893**	0.1671	**22.17**
35%	350	-1	0.2100	0.1882	0.1666	21.98
Average			0.2068	0.1860	0.1671	21.63

Table 4.10: RF with W2V after Lemmatization

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	**0.1877**	0.1839	**0.1683**	21.03
20%	250	1	0.1800	**0.1897**	0.1674	21.93
25%	300	-1	0.1732	0.1891	0.1655	**21.97**
35%	350	-1	0.1705	0.1846	0.1607	21.51
Average			0.1778	0.1868	0.1655	21.61

Table 4.11: RF with W2V after Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.1715	0.1840	**0.1693**	21.31
20%	250	1	0.1740	0.1873	0.1607	21.54
25%	300	-1	0.1731	0.1890	0.1649	22.17
35%	350	-1	**0.1778**	**0.1906**	0.1658	**22.24**
Average			0.1741	0.1878	0.1652	21.82

Table 4.12: RF with W2V after Lemmatization & Stemming

Test Size	N-est	N-jobs	Prec	Recall	Fscore	Acc(%)
30%	100	1	0.1758	0.1842	**0.1688**	21.19
20%	250	1	0.1762	0.1895	0.1685	**22.28**
25%	300	-1	**0.1830**	0.1917	0.1649	22.06
35%	350	-1	0.1817	**0.1918**	0.1602	22.28
Average			0.1792	0.1893	0.1656	21.95

4.1.2 Decision Trees

The following model results will only showcase the ones executed on the preprocessed dataset with lemmatization and stemming. Although some notable differences in precision can be seen before lemmatization and stemming is performed, the recall, f-score and accuracy only had marginal differences and in some cases even some improvements can be observed. Another reason was because the execution time is reduced drastically after performing lemmatization and stemming, making it easier to test.

Table 4.13: DT with TF-IDF after Lemmatization & Stemming

Test Size	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	0.1756	0.1747	0.1744	18.75
20%	0.1809	0.1801	0.1802	**19.22**
25%	**0.1820**	**0.1817**	**0.1812**	19.17
35%	0.1722	0.1721	0.1717	18.23
Average	0.1777	0.1771	0.1768	18.84

Table 4.14: DT with CV after Lemmatization & Stemming

Test Size	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	0.1657	0.1658	0.1651	17.92
20%	0.1721	0.1722	0.1715	18.51
25%	**0.1767**	**0.1760**	**0.1761**	**18.72**
35%	0.1716	0.1698	0.1696	18.19
Average	0.1715	0.1709	0.1706	18.33

Upon comparing the average results in Table 4.14 with that of Table 4.13 there are no notable differences, but when comparing the results on an individual basis it is noticeably that each result of Table 4.14 is roughly 1% lower than Table 4.13. Interestingly though, the third experiment of both Tables 4.14 and 4.13 stand out further than others since the results with the best yield in those tables seem to consistently land on the third experiment (25% test size).

Table 4.15: DT with W2V after Lemmatization & Stemming

Test Size	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	**0.1768**	**0.1771**	**0.1768**	**18.73%**
20%	0.1744	0.1753	0.1746	18.35%
25%	0.1754	0.1751	0.1750	18.69%
35%	0.1729	0.1731	0.1729	18.29%
Average	0.1749	0.1751	0.1748	18.51%

When comparing Table 4.15 with Tables 4.14 and 4.13, although minuscule in difference, the overall results indicate an improvement over the CV model but not quite as good as the TF-IDF model.

4.1.3 Support Vector Machine

The following tables represent the three SVM models executed on the preprocessed dataset with lemmatization and stemming.

Table 4.16: SVM with TF-IDF after Lemmatization & Stemming

Test Size	**Kernel**	**C**	**Gamma**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	RFB	1	Scale	**0.1829**	0.1770	0.1478	20.53
20%	Sigmoid	0.5	Auto	0.0332	0.1667	0.0553	19.90
25%	Linear	0.5	Auto	0.1590	**0.1865**	**0.1590**	**22.00**
35%	Linear	0.5	Scale	0.1563	0.1745	0.1311	20.16
Average				0.1328	0.1762	0.1233	20.65

Starting from Table 4.16, the first notable difference can be seen in the second experiment (20% test size), having a precision value of just **0.0332**. Although the recall's performance was on par with the other three experiments, naturally the f-score still performed poorly since it is dependent on both the precision and the recall. The experiment with the best yield in this table is the third one (25% test size) which although it has a lower precision than that obtained by the default value (RFB), it overcompensated for it via the recall value, obtaining the best f-score and accuracy.

Upon comparing Table 4.17 with Table 4.16 the same pattern can be observed with the precision of the experiment utilising the *Sigmoid* kernel function, yielding a

Table 4.17: SVM with CV after Lemmatization & Stemming

Test Size	**Kernel**	**C**	**Gamma**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	RFB	1	Scale	**0.1838**	0.1815	**0.1783**	21.37
20%	Sigmoid	0.5	Auto	0.0353	0.1667	0.0582	21.16
25%	Linear	0.5	Auto	0.1617	0.1723	0.1440	20.30
35%	Linear	0.5	Scale	0.1780	**0.1827**	0.1503	**21.55**
Average				0.1397	0.1758	0.1327	21.09

value of **0.0353**. The CV model indicates a marginal improvement in performance when compared with the TF-IDF model seeing an improvement in the average accuracy and f-score. Although execution time does not affect the result yield, it was this model's biggest flaw, with some experiments such as the second and third taking over 3 hours to finish executing, making it impractical to use when hardware is an issue.

Table 4.18: SVM with W2V after Lemmatization & Stemming

Test Size	**Kernel**	**C**	**Gamma**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	RFB	1	Scale	0.1838	0.1990	0.1783	22.62%
20%	Sigmoid	0.5	Auto	**0.2412**	0.1989	**0.1893**	22.63%
25%	Linear	0.5	Auto	0.1889	0.1977	0.1793	22.28%
35%	Linear	0.5	Scale	0.1868	**0.2003**	0.1855	**22.94%**
Average				0.2002	0.1989	0.1831	22.62%

Table 4.18 represents the results yielded by the W2V model and is the most promising out of the three SVM models, achieving the highest average results in precision, recall, f-score and accuracy. One of the most notable of results is the precision in the second experiment (20% test size) since both TF-IDF and CV achieved extremely poor precision when using the *Sigmoid* kernel function. The opposite seemed to have happened in the W2V model where the precision result of the *Sigmoid* function scored the highest with a value of **0.2412**. The reason to this cannot be completely determined in this research since there are not enough experiments investigating this particular situation, but it is probably due to W2V being more compatible with Neural Networks (NN) than TF-IDF and CV (Suglia et al. 2017) since the *Sigmoid* function is an NN based kernel function.

4.1.4 Long Short-Term Memory

To keep the results consistent throughout the longevity of this research, the LSTM models were given the same treatment as the previous models by being executed on the pre-processed dataset following lemmatization and stemming. As for the hyperparameters, the Activation types used were *Relu*, *Sigmoid* and *Tanh*, *Relu* being used twice due it achieving the best results across all three models. Table 4.19 shows the first set of results.

Table 4.19: LSTM with TF-IDF after Lemmatization & Stemming

Test Size	**Activation**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	Relu	0.1044	**0.1696**	0.0877	21.14
20%	Sigmoid	0.0984	0.1634	0.0819	**21.68**
25%	Tanh	**0.1258**	0.1683	**0.1085**	19.95
35%	Relu	0.0722	0.1678	0.0608	19.34
Average		0.1002	0.1673	0.0847	20.53

The first notable difference there is from the previous models is the precision, having an average value of **0.1002** which is directly affecting the f-score. The original precision and f-score values were a lot lower due to overfitting but was discovered to be improving by running multiple tests with different batch sizes, number of epochs, input dimensions and output dimensions.

Table 4.20: LSTM with CV after Lemmatization & Stemming

Test Size	**Activation**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	Relu	0.0657	0.1638	0.0898	20.36
20%	Sigmoid	0.0714	0.1679	0.0606	19.50
25%	Tanh	0.0688	0.1674	0.0939	**21.10**
35%	Relu	**0.1072**	**0.1711**	**0.0948**	20.16
Average		0.0783	0.1676	0.0848	20.28

The results displayed in Table 4.20 seem to be almost identical to that of Table 4.19 with the only difference being the average precision scoring a bit lower. Another difference is that in the CV table, the best results seem to consistently land on the fourth (35% test size) experiment with the *Relu* Activation, which although it yielded 1% lower in accuracy than the third (25% test size) experiment, the precision, recall and fscore compensate for the difference.

Table 4.21: LSTM with W2V after Lemmatization & Stemming

Test Size	**Activation**	**Prec**	**Recall**	**Fscore**	**Acc(%)**
30%	Relu	0.1821	**0.1953**	**0.1767**	**22.36**
20%	Sigmoid	**0.2045**	0.1878	0.1718	21.72
25%	Tanh	0.1646	0.1826	0.1515	21.55
35%	Relu	0.1669	0.1832	0.1668	20.93
Average		0.1795	0.1872	0.1667	21.64

Table 4.21 outlines the best LSTM model, showing great improvements in f-score and precision. Albeit a marginal increase, the accuracy also yielded a good result compared with the previous LSTM models, especially in the first (30% test size) experiment, yielding the highest accuracy and recall.

4.1.5 Political Results Analysis

In this subsection, an analysis of the results obtained from the political dataset will be observed. The aim of the approach taken towards this was to identify the best performing model from each classifier executed on LIAR, then identify the best experiment from that model and execute it on the political dataset while keeping the same hyperparameters. Table 4.22 will display the four best performing models on LIAR while Table 4.23 will display the same models but when executed on the political dataset.

Table 4.22: Top Models from each Classifier from the LIAR dataset

	Prec	**Recall**	**Fscore**	**Acc(%)**
RF_W2V_Exp4	0.1778	0.1906	0.1658	22.24
DT_TF-IDF_Exp2	0.1809	0.1801	0.1802	19.22
SVM_W2V_Exp4	0.1868	0.2003	0.1855	22.94
LSTM_W2V_Exp1	0.1821	0.1953	0.1767	22.36

Upon comparing the two tables, although the improvement is extremely marginal, the political dataset seems to outperform the LIAR dataset in terms of accuracy.

Table 4.23: Results from Political dataset

	Prec	**Recall**	**Fscore**	**Acc(%)**
RF_W2V_Exp4	0.1816	0.1891	0.1631	22.45
DT_TF-IDF_Exp2	0.1819	0.1820	0.1813	19.76
SVM_W2V_Exp4	0.1808	0.1883	0.1797	21.99
LSTM_W2V_Exp1	0.1869	0.1892	0.1537	22.57

The political dataset's f-score on the other hand indicates a poorer performance than LIAR due to a lower yield in recall. The precision seemed to have performed similarly between the two. The Y axis in Figure 4.1 represent the accuracy while the X axis represent the experiment of the model pertaining with its respective classifier.

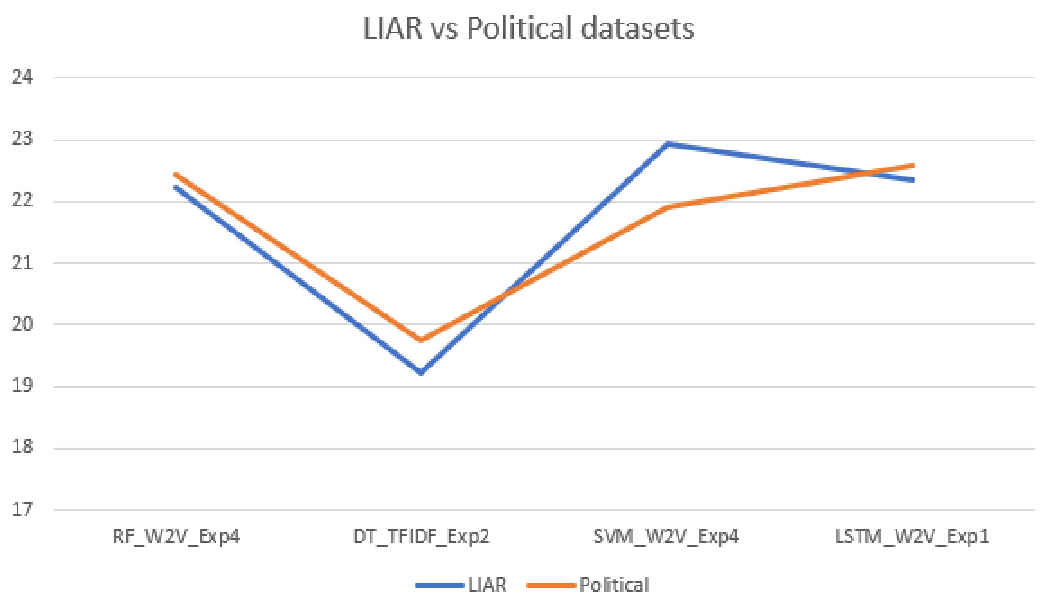


Figure 4.1: LIAR vs Political

4.2 Results Overview

This section will give an overview of the observed results and goes into an in depth comparison. Additionally, the achieved results will also be compared with findings against other studies to obtain a better understanding of the models' performance found in this paper. Table 4.24 exhibits all the average results from the models of the RF classifier.

Table 4.24: Average results of RF models

	Prec	**Recall**	**Fscore**	**Acc(%)**
TFIDF pre Lemm & Stem	**0.2645**	0.1811	0.1630	20.88
TFIDF after Lemm	0.2264	0.1832	0.1624	21.19
TFIDF after Stemming	0.2187	0.1857	0.1641	21.34
TFIDF post Lemm & Stem	0.2182	0.1796	0.1590	20.91
CV pre Lemm & Stem	0.2476	0.1863	0.1663	21.36
CV after Lemm	0.2148	0.1837	0.1628	21.26
CV after Stemming	0.2419	0.1851	0.1641	21.52
CV post Lemm & Stem	0.2170	0.1834	0.1620	21.23
W2V pre Lemm & Stem	0.2068	0.1860	**0.1671**	21.63
W2V after Lemm	0.1778	0.1868	0.1655	21.61
W2V after Stemming	0.1741	0.1878	0.1652	21.82
W2V post Lemm & Stem	0.1792	**0.1893**	0.1656	**21.95**

Although most of the differences among the models were very minimal, when comparing the overall averaged results of the twelve RF tables we can conclude that the best performing model was RF with W2V since it had the best results of recall, f-score and accuracy. With that being said its precision performed the most poorly, with the lowest one being a value of **0.1741** and the highest was that of RF with TF-IDF before lemmatization and stemming with a precision value of **0.2645**. The poorest overall performing model was CV, while although some individual experiments performed very well it had some poor performers which decreased its average score. Comparing the best RF result of Table 4.24 which has an accuracy value of **21.95%** with that obtained in (Hakak et al. 2021), which has an accuracy value of **25.92%**. The author's preprocessing approach in his paper consisted of tokenization, stop-word removal, stemming and feature extraction whereas in this research the approach taken was stop-word removal, punctuation removal, stemming, lemmatization and NLP techniques (TF-IDF, W2V and CV). Another comparison worth making since the author only applied stemming is the author's results with this research's best RF stemming experiment which are **25.92%** and

21.82%, respectively.

Table 4.25: Average results of DT models

	Prec	**Recall**	**Fscore**	**Acc(%)**
TFIDF	**0.1777**	**0.1771**	**0.1768**	**18.84**
CV	0.1715	0.1709	0.1706	18.33
W2V	0.1749	0.1751	0.1748	18.51

Unlike the previous tables, the tables for the remainder of the models will be a lot smaller due to the experiments only being carried out on the original preprocessed dataset along with lemmatization and stemming. Table 4.25 outlines the model utilising TF-IFD to be the best DT model with an f-score and accuracy value of **0.1768** and **18.84%**, roughly underperforming by 3% when compared with RF. In (Hakak et al. 2021), the prediction score DT achieved had a value of **21.23%**.

Table 4.26: Average results of SVM models

	Prec	**Recall**	**Fscore**	**Acc(%)**
TFIDF	0.1328	0.1762	0.1233	20.65
CV	0.1397	0.1758	0.1327	21.09
W2V	**0.2002**	**0.1989**	**0.1831**	**22.62**

Tables 4.26 and 4.27 represents the averaged results of the SVM and LSTM models indicating the W2V model to be the best performing of the three in both tables, respectively having an accuracy score of **22.63%** and **21.64%**. The f-score of W2V in the SVM model is also alot better than that of TF-IDF and CV possibly due to W2V being more compatible with the Sigmoid kernel function, therefore leading to a much higher precision and recall. Upon comparing SVM's performance with DT, RF and LSTM, SVM evidently comes out on top, surpassing DT by roughly 4% and RF and LSTM by 1% in terms of accuracy. In (Wang 2017) the author's yielded accuracy score for SVM is **25.5%** making this research's SVM under perform by 3%.

As for the LSTM, the yielded accuracy was **21.64%** indicating some slight under-performance than the Bi-LSTM found in (Wang 2017) which has a value of

Table 4.27: Average results of LSTM models

	Prec	**Recall**	**Fscore**	**Acc(%)**
TFIDF	0.1002	0.1673	0.0847	20.53
CV	0.0783	0.1676	0.0848	20.28
W2V	**0.1795**	**0.1872**	**0.1667**	**21.64**

23%. Wang's approach in his study was quite different than the one seen in this study since the author only made use of the LIBSHORTTEXT toolkit for SVM and pre-trained word2vec embeddings for Bi-LSTM.

The following figures compose a visual representation of model performances. The Y axis represent the Accuracy while the X axis represent the experiment number.

Figure 4.2, although W2V achieved the highest accuracy score, it is indicating that W2V and TF-IDF are performing similarly in terms of improvement, while CV performed well in the first experiment, poorly and in second and third experiments, finally improving drastically in the fourth experiment. This gives us an visual indication how different hyperparameters can affect the data.

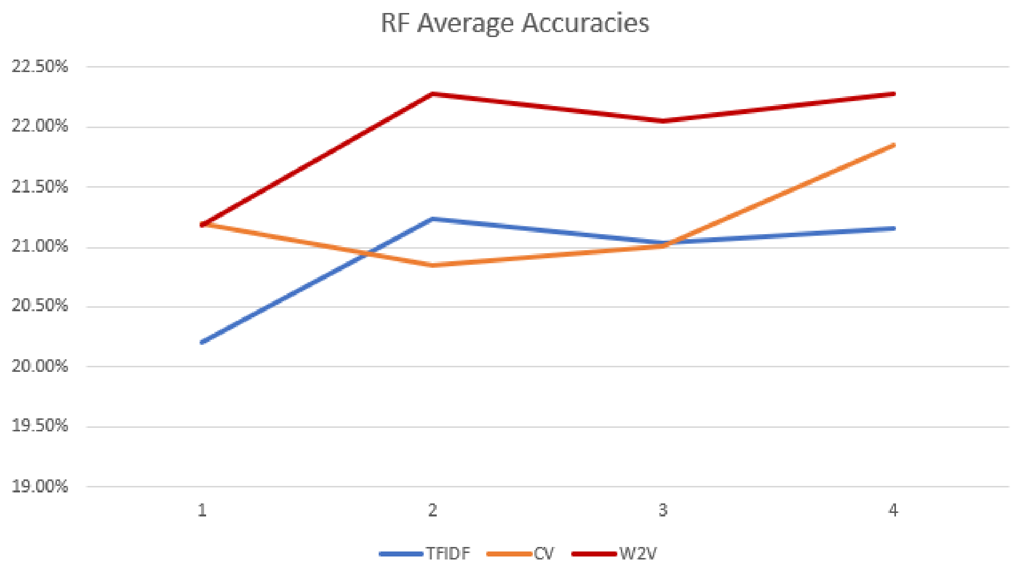


Figure 4.2: Random Forest Average Accuracies

Figure 4.3, although having very different accuracy score, the improvement pattern seen in TF-IDF and CV is the same, improving with each iteration then taking a plunge in the fourth experiment. W2V, although having a much stronger start, at the end it mimicked a similar pattern as TF-IDF and CV by showing a sharp decline in the fourth experiment. This could possibly be due to a hyperparameter common between the three models, negatively affecting performance.

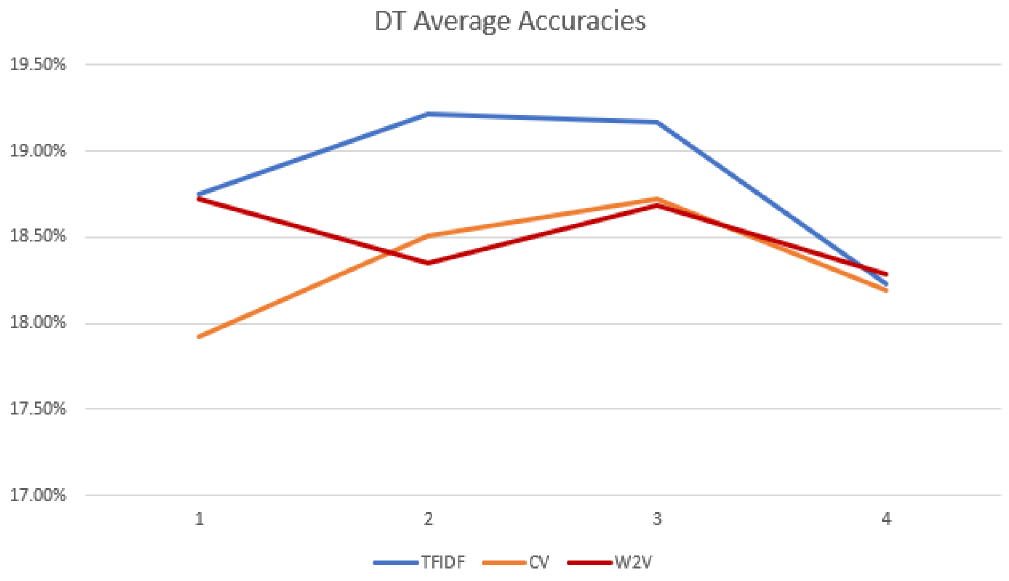


Figure 4.3: Decision Tree Average Accuracies

Figure 4.4 outlines W2V and CV following a similar pattern, showing a decline in accuracy between the second and third experiments and improving drastically in the fourth experiment. TF-IDF on the other hand decreases in performance in the second experiment, improved dramatically in the third experiment and finally exhibiting a sharp decline in the fourth experiment.

Figure 4.5 offers no pattern indications between models, except that each model is showing a similar decline in performance between the third and fourth experiment.

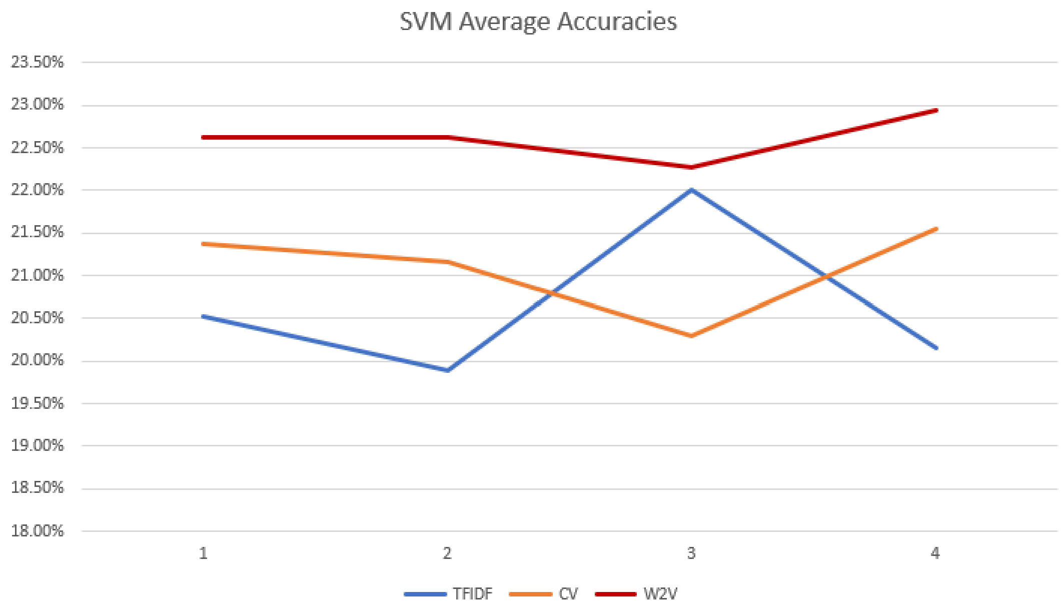


Figure 4.4: SVM Average Accuracies

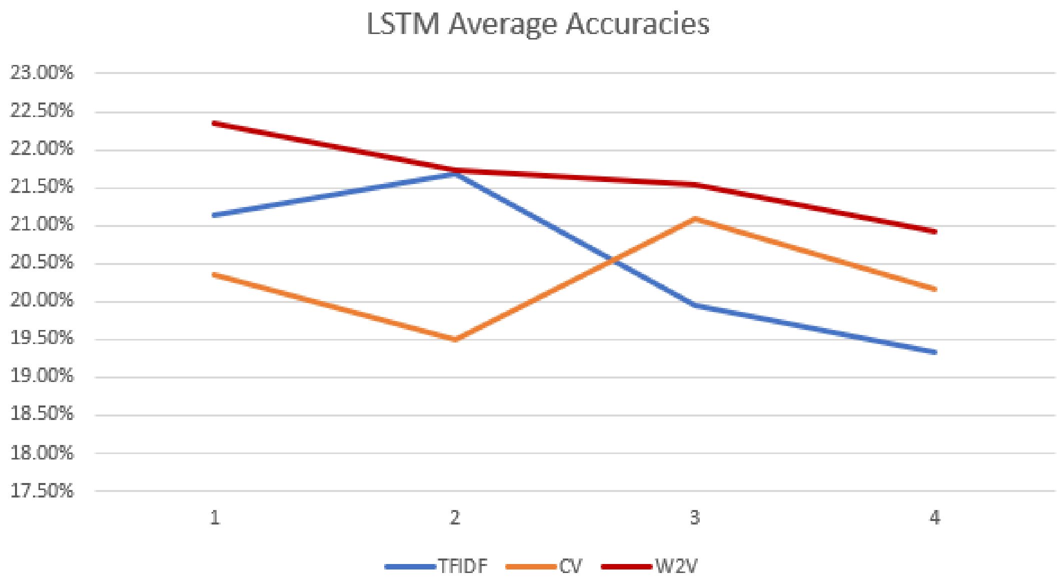


Figure 4.5: LSTM Average Accuracies

4.2.1 Closing Arguments

The below table depicts the average results of the models which were executed on the LIAR dataset, while Figure 4.6 is its visual representation. Upon comparing and reviewing all the presented tables and figures, all evidence seems to outline that the best model is the SVM with W2V, achieving an accuracy score of **22.62%**. A close second was the RF with W2V with an accuracy score of **21.95%**. The worst performing model is DT with CV yielding only an accuracy score of **18.33%**.

Table 4.28: Average results of models

	RF	**DT**	**SVM**	**LSTM**
TF-IDF	20.91	**18.84**	20.65	20.52
CV	21.23	18.33	21.09	20.28
W2V	**21.95**	18.51	**22.62**	**21.64**

In (Hakak et al. 2021), the author's RF and DT results are **25.92%** and **21.23%** while in (Wang 2017), Wang's SVM and Bi-LSTM are **25.5%** and **23.3%**. This indicates that these models outperformed this research's models by about 3%. A number of tests would need to be carried out to properly determine the cause of the under-performance but, judging by the approaches seen in (Hakak et al. 2021) and (Wang 2017) it could possibly be due to some over processing of the data. For example Wang only used word2vec embeddings and the LIBSHORTTEXT toolkit and the results achieved in his SVM were far greater. Another reason for Wang's Bi-LSTM is better is because of its bi-directional nature since Bi-LSTM is proven to perform better than the standard LSTM as seen in this research (Siami-Namini et al. 2019). With this in mind, favour towards this comparison was due to (Wang 2017) being the golden standard of model performances on the LIAR dataset.

Despite incorporating various proven techniques to help improve text classification such as stemming, lemmatization, stop word removal, punctuation removal and the application of various NLP techniques, the models still failed to improve their prediction. Although the experiment on testing the models on a more strictly political dataset was a partial success, the results were still inconclusive because the improvements were extremely marginal in order to construct a more concrete argument.

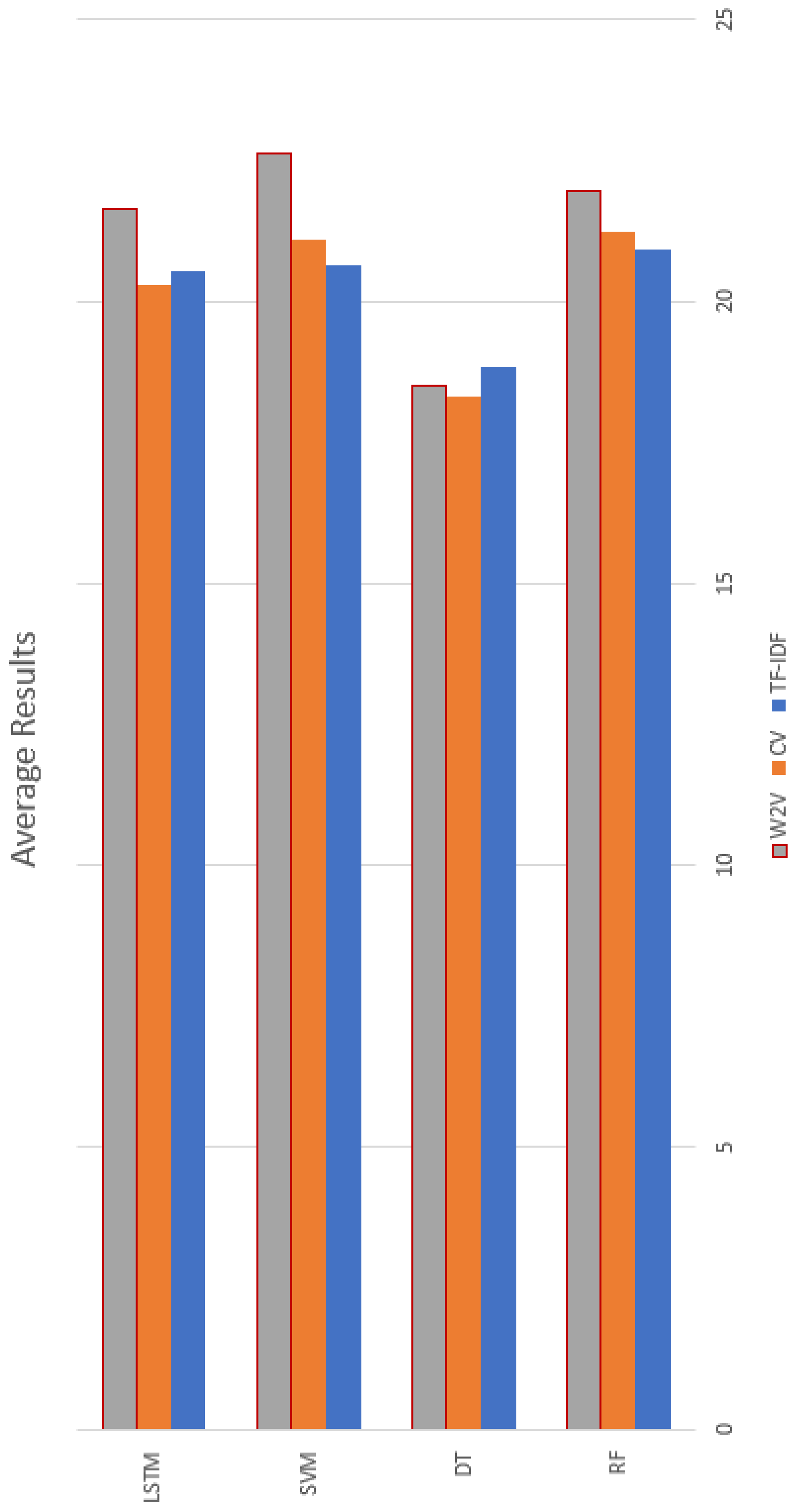


Figure 4.6: Average Accuracies of all Models

Chapter 5

Conclusions & Recommendations

5.1 Conclusion

5.1.1 Chapter Introduction

This chapter is divided into three main sections; the conclusion, which will discuss further the findings of this research study and determine whether or not the hypothesis is satisfied and propose an answer towards the research questions. The second section will make mention of the limitations encountered in this study and the third section will mention any recommendations that might apply for any future work and what approaches could have been tackled differently in this dissertation.

5.1.2 Summary of findings of Random Forest

Multiple experimental setups were configured for the Random Forest (RF) models to determine how lemmatization and stemming affected prediction. The first setup consisted of just the base preprocessing which involved the removal of stop words, removal of punctuation and special characters and converting everything to lowercase. This was showing the highest values in precision. The second setup consisted of the base preprocessing along with lemmatization which had a marginal increase in accuracy but lowered in precision while the recall remained the same. The third setup consisted of the base preprocessing along with stemmming which saw some further improvement in accuracy and an improvement in precision over the second setup. The fourth experimental setup consisted of base preprocessing along with both lemmatization and stemming, yielding the same results as the second setup. A decision was taken to carry out the remainder of the models utilising the fourth experimental setup. The difference in results was very marginal and only affecting the accuracy and precision, therefore it was decided to keep both lemmetization and stemming techniques along with the base preprocessing since evidence through other researches was pointing towards these techniques improving prediction.

Once that was done, comparison between models with different NLP techniques was made. The best performing RF model was the one utilising the W2V NLP technique since it has the highest accuracy, while TF-IDF had the lowest yield in accuracy.

5.1.3 Summary of findings of Decision Trees

In the DT models, TF-IDF shown the best performance out of CV and W2V, having the highest precision, recall, f-score and accuracy, while W2V was a very close second and was the quickest in terms of execution time.

5.1.4 Summary of findings of SVM

The results yielded by the SVM models outlined the W2V to be the best performing, even having the highest accuracy out of all the models in the research study. Additionally it had the best precision, recall and f-score aswell as taking the least amount of time to finish executing, making this model the most optimal.

5.1.5 Summary of findings of LSTM

The findings from the LSTM models also seem to be showing the W2V model as being the best out of the three. When running the experiments on TF-IDF and

CV the precision values were severely under-performing but increased dramatically with the W2V model. The major downside W2V was presenting was the extremely long execution times, making trouble shooting ans debugging very time consuming.

5.1.6 Summary of findings from Political Model

The approach towards obtaining reliable results from the political model consisted of outlining the best experiment from the best model in every classifier, which were; experiment 4 from RF with W2V, experiment 2 from DT with TF-IDF, experiment 4 from SVM with W2V and experiment 1 from LSTM with W2V. Once the findings were obtained from their respective aforementioned models, the same models were executed on the political dataset. In terms of the overall accuracy it was an improvement apart from the SVM, but more tests would need to be carried to properly determine if this research's models detect political fake news more accurately than the more general fake news.

5.1.7 Hypothesis & Research Questions

Hypothesis: In this research study it is hypothesized that prediction models will offer a greater accuracy when performed on political fake news rather than the more general fake news.

From the conclusion of this research study based off the reviewed results, it can be determined that the hypothesis has been partially proven. Although the accuracy results obtained from the political model shown to have have improved its prediction, the difference does not justify the hypothesis to be a complete success.

Research Questions

- *Can this research achieve a satisfactory result in detecting political fake news while corresponding with similar research studies?*

Answer: The political model was a partial success when compared with models within the same parameters of this research, but failed when it was compared to other similar studies. Although its prediction score was close, it still failed to surpass or equalise the prediction score.

- *Can this research successfully predict the surge of political fake news?*

  Answer: Due to the low accuracy score from both the prototype and from LIAR itself, the models failed to successfully predict a pattern to determine if a surge in fake news is happening.

5.2 Limitations

Throughout the longevity of the research there were numerous limitations that were encountered. The biggest limitation was the construction of the LSTM when trying to integrate it with the proposed NLP techniques. Furthermore, it took alot of time researching its hyperparameters and configurations to properly understand what was causing overfitting. Another big limitation was the dataset. Since the scope of my research had a political alignment, a purely political dataset was required. Since LIAR was the dataset of choice for the general fake news due to its 6-way classification labeling method, a similar dataset had to be found having a similar labeling method. Unable to find such dataset, the LIAR dataset had to be altered to pose as a political dataset. Another limitation was the number of models required to be constructed and document the results each model outputted. This was due to an unforeseen technicality at the time when beginning this research. This made it hard and time consuming to construct and keep track of all the results that needed to be documented.

5.3 Recommendations

For future studies, it is recommended to use fewer algorithms and NLP techniques depending on the study and available time. This will help with keeping the research study more precise and keep realistic expectations about a particular topic. Furthermore, it will also make it easier to troubleshoot or debug a problem which in return would help with building a more efficient model. Unfortunately in this study, only the top four models from each classifier were executed on the political dataset due to insufficient time, therefore another recommendation would be to conduct more tests. If all the models were executed using the LIAR dataset, then it should also be the same for the political dataset. The more results there are to compare, the better the understanding between the correlation of the LIAR dataset and the political LIAR dataset.

Appendix A

Python Libraries Used

Numpy

Pandas

Matplotlib

Seaborn

Gensim

Gensim: Word2Vec

Nltk

Nltk: stopwords

Nltk: punkt

Nltk: wordnet

Nltk: WorkNetLemmatizer

Nltk: PorterStemmer

Sklearn: ensemble, preprocessing, feature_extraction, metrics, classification_report, accuracy_score, precision_recall_fscore_support

Sklearn: svm

Sklearn: train_test_split

Sklearn: metrics

Sklearn: DecisionTreeClassifier

Sklearn: train_test_split

Keras: EarlyStopping

Tensorflow.keras: Sequential

Tensorflow.keras: Dense, Conv1D, Embedding, LSTM, MaxPool1D, Dropout, SpatialDropout1D

Tensorflow.keras: pad_sequences, Tokenizer

References

Alam, S. & Yao, N. (2019), 'The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis', *Computational and Mathematical Organization Theory* **25**(3), 319–335.

Alhindi, T., Petridis, S. & Muresan, S. (2018), Where is your evidence: improving fact-checking by justification modeling, *in* 'Proceedings of the first workshop on fact extraction and verification (FEVER)', pp. 85–90.

Battineni, G., Chintalapudi, N. & Amenta, F. (2019), 'Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (svm)', *Informatics in Medicine Unlocked* **16**, 100200.

Bengtsson, M. (2016), 'How to plan and perform a qualitative study using content analysis', *NursingPlus open* **2**, 8–14.

Bogach, I. & Kovenko, V. (2020), Recommendation system based on NLP techniques, PhD thesis, BHTY.

Burkhardt, J. M. (2017), 'History of fake news', *Library Technology Reports* **53**(8), 5–9.

Corbu, N., Oprea, D.-A., Negrea-Busuioc, E. & Radu, L. (2020), "they can't fool me, but they can fool the others!' third person effect and fake news detection', *European Journal of Communication* **35**(2), 165–180.

Creswell, J. W. & Clark, V. L. P. (2004), 'Principles of qualitative research: Designing a qualitative study', *Office of Qualitative & Mixed Methods Research, University of Nebraska, Lincoln* .

Cușmaliuc, C.-G., Coca, L.-G. & Iftene, A. (2018), Identifying fake news on twitter using naive bayes, svm and random forest distributed algorithms, *in* 'Proceedings of The 13th Edition of the International Conference on Linguistic Resources and Tools for Processing Romanian Language (ConsILR-2018) pp', pp. 177–188.

Farhall, K., Carson, A., Wright, S., Gibbons, A. & Lukamto, W. (2019), 'Political elites' use of fake news discourse across communications platforms', *International Journal of Communication* **13**.

Goyal, R. (2021), Evaluation of rule-based, countvectorizer, and word2vec machine learning models for tweet analysis to improve disaster relief, *in* '2021 IEEE Global Humanitarian Technology Conference (GHTC)', IEEE, pp. 16–19.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2016), 'Lstm: A search space odyssey', *IEEE transactions on neural networks and learning systems* **28**(10), 2222–2232.

Hakak, S., Alazab, M., Khan, S., Gadekallu, T. R., Maddikunta, P. K. R. & Khan, W. Z. (2021), 'An ensemble machine learning approach through effective feature extraction to classify fake news', *Future Generation Computer Systems* **117**, 47–58.

Hayes, B. K. & Heit, E. (2018), 'Inductive reasoning 2.0', *Wiley interdisciplinary reviews: cognitive science* **9**(3), e1459.

Heale, R. & Twycross, A. (2015), 'Validity and reliability in quantitative studies', *Evidence-based nursing* **18**(3), 66–67.

Heinrich, A. (2019), How to build resilient news infrastructures? reflections on information provision in times of "fake news", *in* 'Resilience and Hybrid Threats', IOS Press, pp. 174–187.

Huang, S., Cai, N., Pacheco, P. P., Narrandes, S., Wang, Y. & Xu, W. (2018), 'Applications of support vector machine (svm) learning in cancer genomics', *Cancer genomics & proteomics* **15**(1), 41–51.

Hussain, M. G., Hasan, M. R., Rahman, M., Protim, J. & Hasan, S. A. (2020), 'Detection of bangla fake news using mnb and svm classifier', *arXiv preprint arXiv:2005.14627* .

Ipeirotis, P. G. (2010), 'Analyzing the amazon mechanical turk marketplace', *XRDS: Crossroads, The ACM magazine for students* **17**(2), 16–21.

Jehad, R. & Yousif, S. A. (2020), 'Fake news classification using random forest and decision tree (j48)', *Al-Nahrain Journal of Science* **23**(4), 49–55.

Kulkarni, A. & Shivananda, A. (2019), Converting text to features, *in* 'Natural language processing recipes', Springer, pp. 67–96.

Levi, L. (2017), 'Real fake news and fake fake news', *First Amend. L. Rev.* **16**, 232.

Marcos-Pablos, S. & García-Peñalvo, F. J. (2020), 'Information retrieval methodology for aiding scientific database search', *Soft Computing* **24**(8), 5551–5560.

Mazzeo, V., Rapisarda, A. & Giuffrida, G. (2021), 'Detection of fake news on covid-19 on web search engines', *arXiv preprint arXiv:2103.11804* .

Murayama, T., Wakamiya, S. & Aramaki, E. (2021), 'Mitigation of diachronic bias in fake news detection dataset', *arXiv preprint arXiv:2108.12601* .

Murray, P. (2017), 'National: 'fake news' threat to media; editorial decisions, outside actors at fault', *Polling Institute, Monmouth University, New Jersey* .

Nasir, J. A., Khan, O. S. & Varlamis, I. (2021), 'Fake news detection: A hybrid cnn-rnn based deep learning approach', *International Journal of Information Management Data Insights* **1**(1), 100007.

Oshiro, T. M., Perez, P. S. & Baranauskas, J. A. (2012), How many trees in a random forest?, *in* 'International workshop on machine learning and data mining in pattern recognition', Springer, pp. 154–168.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A. & Mihalcea, R. (2017), 'Automatic detection of fake news', *arXiv preprint arXiv:1708.07104* .

Qaiser, S. & Ali, R. (2018), 'Text mining: use of tf-idf to examine the relevance of words to documents', *International Journal of Computer Applications* **181**(1), 25–29.

Shi, T. & Horvath, S. (2006), 'Unsupervised learning with random forest predictors', *Journal of Computational and Graphical Statistics* **15**(1), 118–138.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D. & Liu, H. (2018), 'Fakenewsnet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media', *arXiv preprint arXiv:1809.01286* .

Shu, K., Sliva, A., Wang, S., Tang, J. & Liu, H. (2017), 'Fake news detection on social media: A data mining perspective', *ACM SIGKDD explorations newsletter* **19**(1), 22–36.

Siami-Namini, S., Tavakoli, N. & Namin, A. S. (2019), The performance of lstm and bilstm in forecasting time series, *in* '2019 IEEE International Conference on Big Data (Big Data)', IEEE, pp. 3285–3292.

Skafidas, E., Testa, R., Zantomio, D., Chana, G., Everall, I. P. & Pantelis, C. (2014), 'Predicting the diagnosis of autism spectrum disorder using gene pathway analysis', *Molecular psychiatry* **19**(4), 504–510.

Suglia, A., Greco, C., Musto, C., De Gemmis, M., Lops, P. & Semeraro, G. (2017), A deep architecture for content-based recommendations exploiting recurrent neural networks, *in* 'Proceedings of the 25th conference on user modeling, adaptation and personalization', pp. 202–211.

Thavareesan, S. & Mahesan, S. (2020), Word embedding-based part of speech tagging in tamil texts, *in* '2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)', IEEE, pp. 478–482.

Torabi Asr, F. & Taboada, M. (2019), 'Big data and quality data for fake news and misinformation detection', *Big Data & Society* **6**(1), 2053951719843310.

van der Linden, S., Panagopoulos, C. & Roozenbeek, J. (2020), 'You are fake news: political bias in perceptions of fake news', *Media, Culture & Society* **42**(3), 460–470.

Vijayaraghavan, S., Wang, Y., Guo, Z., Voong, J., Xu, W., Nasseri, A., Cai, J., Li, L., Vuong, K. & Wadhwa, E. (2020), 'Fake news detection with different models', *arXiv preprint arXiv:2003.04978* .

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

Wu, D. J., Feng, T., Naehrig, M. & Lauter, K. (2015), 'Privately evaluating decision trees and random forests', *Cryptology ePrint Archive* .

Yang, S., Shu, K., Wang, S., Gu, R., Wu, F. & Liu, H. (2019), Unsupervised fake news detection on social media: A generative approach, *in* 'Proceedings of the AAAI conference on artificial intelligence', Vol. 33, pp. 5644–5651.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F. & Choi, Y. (2019), 'Defending against neural fake news', *Advances in neural information processing systems* **32**.