Fake News Detection

CS39440 Major Project Report

Author: William Minton (wim12@aber.ac.uk)

Supervisor: Dr Richard Jensen (rkj@aber.ac.uk)

11th May 2020

Version 1.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

Name: William Minton

Date: 11/05/2020

Consent to share this work

Name: William Minton

Date: 11/05/2020

Acknowledgements

I am grateful to my supervisor, Richard Jensen, for providing support via email and skype, especially as a part-time external student

I’d like to thank my family, friends, and everybody who supported me throughout this process

Abstract

Fake news is a prevailing issue in modern society, being spread extensively across news and social media. According to Ofcom (2019) [1], as much as 66% of people use the internet for news, with 49% of people overall who use social media as a method of news consumption. Many social media platforms allow for information to be spread rapidly, without any methods of checking if the information is factually accurate, leading to a rise in deliberate misinformation, hoaxes and conspiracy theories. This can occur for political reasons, to purposefully ruin the reputation of others, or simply as a manner of seeking attention, and can have a major impact on the zeitgeist of the time. The spread of misinformation in this manner has been a major focus of the news cycle in the last few years, as seen after both the 2016 US presidential elections, as well as the 2016 EU referendum in the UK. The detection and classification of fake news is therefore one of the major challenges of the modern era.

In this paper, various different machine learning solutions to this classification problem will be compared and evaluated on three separate datasets. Multiple methods of natural language processing are used to assimilate textual information contained within the datasets into machine-readable language and are then split into training and test datasets. These training datasets are used to train multiple machine learning classification models, with the trained models then being used with the test datasets to predict if the news presented in the test dataset is real or fake, with the results compared and evaluated using various different evaluation metrics. This is a realistic, data-driven supervised approach to fake news detection based around text-classification and serves to succinctly demonstrate the capability of machine learning algorithms in classifying and detecting fake news articles.

Contents

1 Background, Analysis & Process

1.1 Background

Generally, background preparation for this project was primarily done via research and analysis of various different methods of fake news detection, and the implementation of these methods within the machine learning and computational intelligence sectors. The author has a significant interest in fake news, being an avid newsreader with a politically focused mindset. Inspiration for this project was taken from the current public discourse over fake news, with the Brexit vote and Donald Trump's presidency contributing significantly to this. The idea and history behind fake news has always been fascinating, from the dystopian societies presented in George Orwell's 1984 [2] and Aldous Huxley's Brave New World [3], to the supposed "Wave of mass hysteria" reported by the New York Times [4] caused by the unannounced broadcast of HG Wells' War of the Worlds, which in itself is apparently an example of Fake News through anecdotal evidence [5]. Being able to apply machine learning approaches to solving this problem was an interesting opportunity and offered a significant challenge in determining ideal methods to attempt detecting fake news.

1.2 Analysis

When it came to initial preparation, the definition and characterisation of Fake News was seen as an ideal place to start. As a complicated topic that can be defined in a number of ways, a strong understanding of the logic underpinning Fake News was a necessity in understanding and tackling the detection of it.

Rubin, Chen and Conroy(2015) [6] suggested that there were three major types of deceptive news: Serious Fabrications, Large-Scale Hoaxes, and Humorous Fakes. The recognition of these different types, as well as the ability to differentiate between them is important in various different natural language processing models. The idea of Humorous Fakes offers a different reflection on the topic of fake news, serving as a form of mild entertainment satirising the news industry, rather than as an attempt at malicious deliberate misinformation or conspiracy theory. Serious Fabrications and Large-Scale hoaxes were seen as types of fake news distributed with malicious intent and were topics that would be focused on more within this project due to this.

With this providing a basis into how fake news should ideally be classified, investigations into similar research such as Ahmed, Hinkelmann and Corradini (2019) [7] suggested that there were multiple methods of doing so, outlining four major approaches to automated Fake News Detection. These were Stance Detection, Quantifying Metadata, Fact Checking and Text Classification.

1.2.1 Different approaches to Automated Fake News Detection

1.2.1.1 Stance Detection

Stance detection is a form of natural language processing, considered as a subproblem of sentiment analysis, that focuses on the intent of the author towards a specialised target suggested in the text. The estimation of this "Stance" could then be used in a variety of ways. In their investigation of Stance Detection methods for the Fake News Challenge, Chaudhry et al. [8] suggested the use of high accuracy classifiers as a tool to assist manual human fact checkers, or as part of a substantial AI system that would use multiple approaches to identify truthfulness. Ultimately, this method was considered, but would be deemed unviable for a few reasons. The use of this method is generally supplementary to other approaches, working to assist their veracity rather than working to detect fake news by itself. The method of sentiment analysis, whilst being an interesting topic, was therefore too far removed from the initial Fake News Detection project topic. The accuracy of Stance Detection was also questionable at best, with Ghanem, Rosso and Rangel's investigation (2018) [9] reaching an accurate result of 59.6 % Macro F1, leading to the choice of another method of Fake News Detection.

1.2.1.2 Qualifying Metadata

Qualifying Metadata is another method highlighted by Ahmed et al.(2019) [7] for the automated detection of fake news. This involves analysing the metadata of suspected articles, such as the location, time, send frequency, and author and comparing the information to reputable news sites to see if the same news was published, or if deliberate Serious Fabrications or Large-Scale Hoaxes have first appeared in places or times that would signal their falsehood, such as news about American elections first appearing in Russia. Acker (2018) [10] argues that information manipulators can use "Platform activity signals" including usernames, profile handles, bio fields, dates of posted photos, followers and following counts, and hearts on posts to control and manipulate social media systems, mimic legitimacy, and specify and target users of these platforms, whether with advertisements or deliberate misinformation. This was seen as a difficult method of fake news detection, however as it would require a dataset of comparable articles, with the relevant metadata fully available, which do not seem to be readily available. Careful attention should therefore be paid to metadata analysis in the development of a substantial Fake News Detection AI, as it would form an integral addition to the portfolio of different detection methods it could use.

1.2.1.3 Fact checking

A further method of detection comes in the form of Fact Checking. This is a type of Knowledge Engineering that focuses on checking the facts of the news based on the known facts (Ahmed et al., 2019), working to approximate the complex nature of human fact checking via machine learning methods. This is a necessary next step in online news verification, as the volume of new information has exponentially increased with the rise in news and social media based internet usage, requiring a move away from human verification and towards automatic methods of data classification. Ciampaglia et al. [11](2015) suggests the use of triples, a collection of data in the subject, predicate, object format to facilitate this, by forming a knowledge graph with a set of such triples. Nodes in this knowledge graph would denote subjects and objects, while edges would denote predicates. This knowledge graph would therefore form a web of structured data. Given a trusted dataset of true statement triples, a substantial knowledge graph could be queried with a new statement to determine its truth, with it being true if it exists as an edge, or if there is a short path linking the statement's subject to its object within the knowledge graph, with the statement being false otherwise. This approach lead to generally positive results, and also has some range of improvement based around different methods of computing the shortest path. This gave the approach strong consideration for focus in this project, however the lack of significant further study ultimately gave pause to this idea, with only the paper previously mentioned and a scientific journal article (Wu et al.) [12] approaching the topic of computational fact checking. Significant research into sentiment analysis would also have to be done, as well as further Natural language processing and text classification research to be able to automatically extract the subject, predicate and object of the news articles to be verified.

1.2.1.4 Text classification

Text Classification is the automated fake news classification option that was selected for further study in this project. The methodology involves extraction of features from text, and then training various classification models to recognize these features extracted from the text articles present in the dataset. The model should then be able to determine if another test set of data presented is fake or non-fake. The first step of training is feature extraction, which will reduce the raw data (words) into more manageable groups for processing, which will save on computing resources and allow for large amounts of data to be parsed on personal computers. This is a method of dimensionality reduction which works via mapping textual data to real valued vectors, more specifically the words will be classified by their score in a predefined dictionary of words, and will then be numerically represented by a vector, which is known as the "Bag of Words" approach. The machine learning algorithm models will then be fed with the feature-extracted training data. The models will be trained on a training set, tuned using a validation set, and tested using a test set of data. The feature extraction process will be further explained in the design section, as will the selection and representation of the various models used in this project.

Investigation was also done into the various models that could be used for Data Classification. Since a machine learning approach of text classification was decided upon, multiple scientific papers were consulted on the ideal methodology behind this, which are further covered in the Libraries and Languages section.

1.3 Process

The project is research-based, so deciding on a specific software methodology was fairly difficult. Ultimately, Scrum was decided upon, due to having the benefits of an iterative design process similarly to XP with each sprint, although it also allows for non-standard practices in design and testing. This was kept to for the initial stages of the process, although due to the changing circumstances in the world, this eventually devolved into a FDD Hybrid process.

2 Experiment Methods

2.1.1 Dataset Selection

With this basic categorisation in mind, selecting ideal datasets appeared to be the perfect next step in preparation. Following further online research, the Awesome Fake News Github Repository [13] was found, offering a substantial source for various different fake news datasets, as well as numerous further research papers. This led to the selection of three distinct datasets, those being the Fake News Corpus [14], the "Fake or Real News" Dataset [15] , and the "Liar, Liar pants on fire" dataset" [16]. Generally, the smaller "Fake or Real News" and "Liar Liar" datasets were selected as a benchmarking dataset. As smaller datasets than the 28GB "Fake News Corpus", they were used for troubleshooting issues with the machine learning process as the models could be trained much faster than with the larger datasets, and also worked to evaluate the performance of the models on these smaller datasets. It was deemed important to gauge the model's performance on a variety of different datasets, which would show a full metric of the models capability in different situations, and allow a more complete appraisal of the total accuracy of the model used.

2.1.1.1 Liar Liar Dataset

The "Liar Liar" dataset was also selected as it did not contain the main body of the text, simply the article headline. This allowed for the evaluation of the models effectiveness on a dataset composed of shorter sentences, although the expectancy of high-quality results is low. The dataset initially contained six different labels for the truth of these headlines, which were:

- True
- Mostly True
- Half True
- Barely True
- False
- Pants-Fire

As it was eventually decided that the project would be focused on as a true/false binary classification problem, the use of six labels would overcomplicate this, so a classification threshold had to be determined. A simple threshold would end up being used, with headlines under the "True" and "Mostly True" labels being categorised as True, and headlines under the "False" and "Pants-Fire" labels categorised as False. The "Half True" and "Barely True" labels effectively functioned to categorise articles within the grey area between credible and fake news, and would be of particular relevance if fake news classification were to be treated as a multilabel classification problem, which may be an avenue for further research. Headlines labelled with either of these two were deleted from the database used in this project. This led to a total of

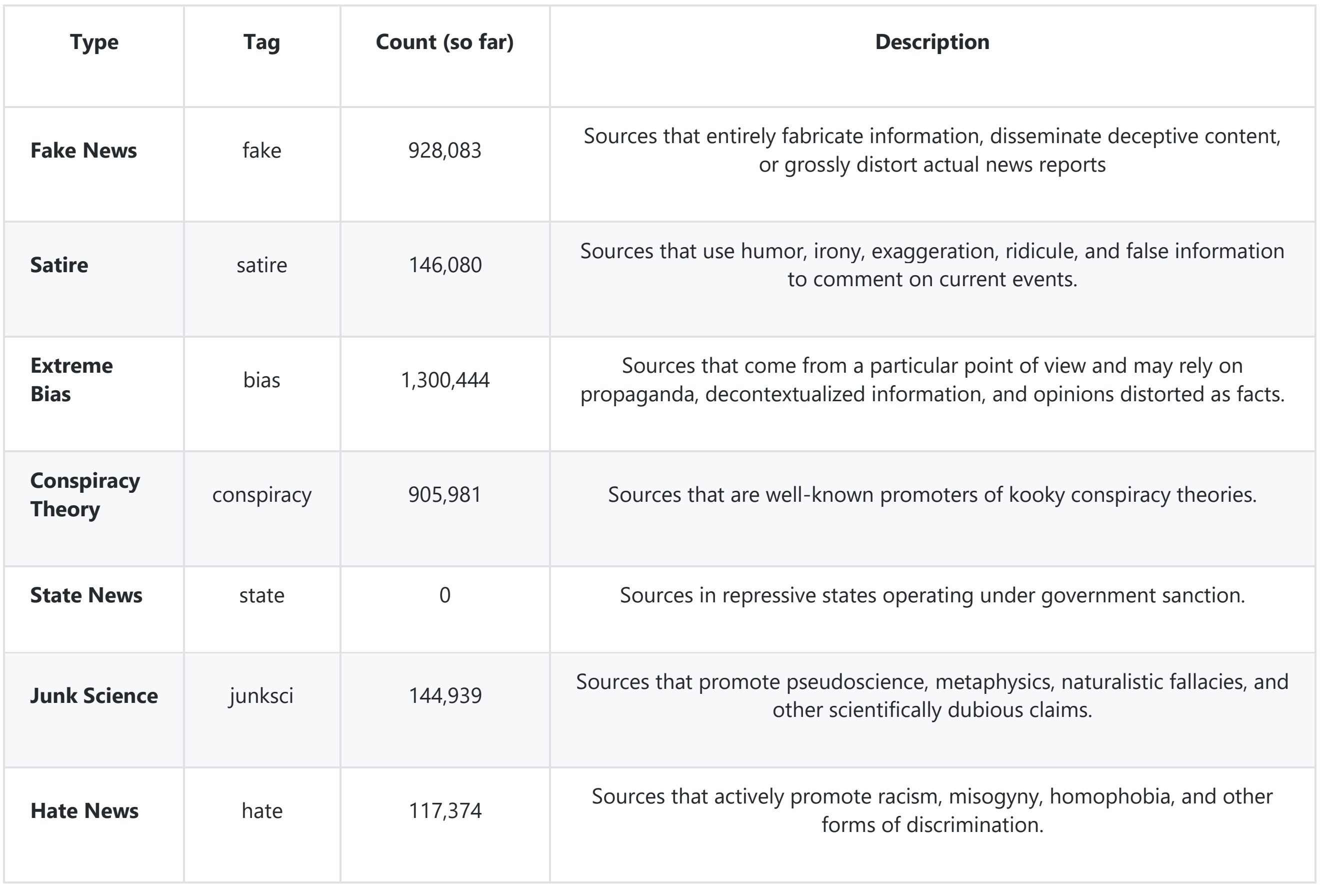
2.1.1.2 Fake News Corpus

The Fake News Corpus was a very large dataset to be appraised, with numerous different labels for the text content of the articles. There were several different fields present in the corpus, which was formatted as a csv (Comma Separated Value) file. The fields present were:

- Id
- Domain
- Type

- URL
- Content
- Scraped_at
- Inserted_at
- Updated_at
- Title
- Authors
- Keywords
- Meta_keywords
- Meta_Description
- Tags
- Summary
- Source (Opensources, NYtimes, or webhose)

These fields offered a complete overview of the data's content, metadata, and how the information was gathered, offering information useful to a variety of different types of data classification. The fields useful to this project would be ID, Type, Content, and Title, as Natural Language Processing would later be determined as the method of classification. The ID signified the identification number of each article, to easily determine which article was being referred to when debugging. The content and title show the headline and written news content of the article itself, whereas the type is where the article fits in the spectrum of real to fake news, as can be seen in figure 1, taken from the Fake News Corpus github repository [14].



Type	Tag	Count (so far)	Description
Fake News	fake	928,083	Sources that entirely fabricate information, disseminate deceptive content, or grossly distort actual news reports
Satire	satire	146,080	Sources that use humor, irony, exaggeration, ridicule, and false information to comment on current events.
Extreme Bias	bias	1,300,444	Sources that come from a particular point of view and may rely on propaganda, decontextualized information, and opinions distorted as facts.
Conspiracy Theory	conspiracy	905,981	Sources that are well-known promoters of kooky conspiracy theories.
State News	state	0	Sources in repressive states operating under government sanction.
Junk Science	junksci	144,939	Sources that promote pseudoscience, metaphysics, naturalistic fallacies, and other scientifically dubious claims.
Hate News	hate	117,374	Sources that actively promote racism, misogyny, homophobia, and other forms of discrimination.

Type	Tag	Count (so far)	Description
Clickbait	clickbait	292,201	Sources that provide generally credible content, but use exaggerated, misleading, or questionable headlines, social media descriptions, and/or images.
Proceed With Caution	unreliable	319,830	Sources that may be reliable but whose contents require further verification.
Political	political	2,435,471	Sources that provide generally verifiable information in support of certain points of view or political orientations.
Credible	reliable	1,920,139	Sources that circulate news and information in a manner consistent with traditional and ethical practices in journalism (Remember: even credible sources sometimes rely on clickbait-style headlines or occasionally make mistakes. No news organization is perfect, which is why a healthy news diet consists of multiple sources of information).

Figure 1 : A Table showing the different types of fake news present in the fake news corpus, with the amount of them and a brief description. [14]

As can be seen from figure 1, there are 11 types of news, real and fake, represented in this dataset. These signify a large scale analyses of the different types of news across the real/fake spectrum, and, other than the four credible and unknown types present across the bottom of the table, can be sorted into the three types of fake news described by Rubin et al. [6], with some categories that fit into both the serious fabrication and large scale hoaxes theory, such as hate news that uses pseudoscience to attempt to seem credible.

2.1.1.2.1 Regularizing the Corpus

The initial download for the Fake News Corpus was as 9 separate 1GB compressed files. When decompressed and extracted, they would be 9 28GB csv files with the amount of data present, with the amount of articles being 8,510,549. To train and test a number of different machine learning models on 8.5 million different articles would not really be possible, especially on hardware not particularly suited to machine learning. Naturally this dataset needed to be cut down. An additional python program was created to facilitate this, as the large csv files could not be opened in Excel, due to it's limit of 1,048,576 rows. It would load the data as a dask dataframe, due to the memory limit being too high to reliably run with a standard pandas dataframe [17]. Dask [18] offered an ideal alternative, as it would allow for "Out of core" processing of the dataframes, as opposed to the pandas dataframe which required fitting into memory. This meant that even with limited virtual memory available on the computer for machine learning, the dask dataframe was able to load the file that wouldn't fit into the virtual memory allocated by the computer, and split it into multiple different excel files that could easily be loaded.

This resulted in 458 excel files, which had to be searched and filtered through to determine which articles would be viable. Many of the types that the dataset was categorized under would generally only serve to overcomplicate the classification process. As it was decided that this would be a binary classification problem, rather than a multi-class classification problem, these classes naturally had to be reduced down. As satire came under the category of humorous fakes [6], articles classified under satire would be removed, as well as unreliable and clickbait articles, as these would generally not tell the model anything significant and would only serve to confuse the model if classified as real or fake. Generally, the political and credible labels were identifiable as real news, whereas the hate, junk science, conspiracy theory, bias, and fake labels could be identifiable as fake news.

Naturally, this presented a choice of options; to utilize the loose grouping of various labels as binary classifiers, offering increased noise and the possibility of overfitting, although also with the advantage of a larger possible dataset, or to simplify the dataset further, and cut out every type other than the credible and fake labels. There would be a smaller dataset, although it would be easier to train and test models with this dataset, due to less memory being used. Would the benefit of more in-depth analyses outweigh the performance costs of data processing and analysis? This method of model constraint and simplification is known as regularization, and often helps reduce the risk of overfitting, as Geron describes [19]. For this report, the decision to focus on performance was made, with the smaller dataset of credible and fake labels used, with articles classed with the other labels deleted. With access to hardware more suited to machine learning, the former choice would be the superior option, although the computer used for this project would suffer Stop Errors (Blue Screen of Death) when testing datasets that were too large. The practical option was therefore decided, with a python file created to search through the 458 csv files and add the articles with credible and fake labels to a new csv file, as well as updating the ID numbers for each article. This new file was ultimately reduced down to a subset of 500000 articles, which was seen as a compromise to avoid overfitting and hardware issues, while having a big enough dataset to comfortably train and test the data.

With the datasets selected and processed, research into Natural Language Processing and Text Classification was the next step that needed to take place.

Natural language processing & text classification

Models and hyperparameters

Research into this was done via published scientific papers,

2.2 Hypothesis

The hypothesis

2.3 Programming Language and Libraries

2.3.1 Language Selection

The selection of a programming language for this project was a relatively simple affair. From prior knowledge, there were four choices of language initially considered; Python, C++, Java, and R. R is a language specialised for statistical analysis, and has a number of libraries available for machine learning. It is also a popular language, ranking at #5 on the IEEE spectrum's 2019 ranking of Programming languages [20] despite being specifically designed for statistical analysis and data wrangling. It is, however, the only language listed which the author does not have personal experience in, and it was seen that the difficulty of learning a new programming language would outweigh the benefits that this language would bring.

Java and C++ were other languages considered, both being object-oriented languages with large amounts of machine learning libraries available, such as Smile [21] and Tensorflow [22] respectively. Where they differ, though, is that Java is an interpreted language, whereas

C++ is compiled. As the code for Java is interpreted during run time, as opposed to C++ which is precompiled to binary, C++ would run more efficiently than Java. The other language mentioned, Python, is also an interpreted language, and therefore slower to run than C++. The advantages that it does offer, however, are in ease of use, garbage collection, and flexibility, allowing for cleaner code to be written and debugged more easily than C++. Python also has a large quantity of machine learning libraries available, including the popular and simple to use Scikit-Learn [23], as well as pandas [17] [24] for the easy management of datasets, NumPy [25] [26] for facilitating mathematical options on higher-dimensional data, and matplotlib [27], which can be used for simplifying the visualisation of this data. Additionally, Python is also the most popular language amongst machine learning repositories on github [28], and is referred to as the "second most loved language" and "Fastest-growing major programming language today" in the 2019 Stack Overflow Developer Survey [29]. With that in mind, Python was the language selected for use in this project.

2.3.2 Libraries used

As Python was the language selected, what followed was the choice of external libraries, allowing for additional functionality to be easily implemented in a simpler and less computationally expensive manner. This would serve to drastically reduce the difficulty of the project, and allow for the hypothesis to be tested in a productive and objective manner.

As the datasets had already been decided upon, one of the initial necessities was the determination of a method of data manipulation, which would serve to assist in data analysis. Pandas [17] [24] is the third most popular machine learning library on github [28], and is the preeminent library for data analysis and manipulation. As this project involved feature extraction and storage of large datasets, there was generally no competition in determining how the data would be stored and wrangled. The use of pandas dataframes would serve as the backbone of this project, and would be invaluable in their ability to store and manipulate data in a consistent, quick, and easily manageable fashion, effectively functioning similarly to excel spreadsheets.

In attempting to read through and utilise the "Fake News Corpus" dataset, it soon became apparent that the big data presented, at 28GB, was too large to efficiently use Pandas to manipulate. One of the few issues with the Pandas dataframes is that all data has to be stored in memory, which causes issues when using datasets with large filesizes, especially on machines with limited RAM. The Dask Library [18] was used in this circumstance. This library is similar to the Pandas library, with slightly more limited functionality due to its lazy execution. It does, however, allow for out of core data streaming from disk. This allows for the processing of data that doesn't fit into memory, such as the "Fake News Corpus" dataset. Dask also has limited functionality with Scikit-learn, although this was not used, as it was only compatible with a small amount of scikit-learn's classification models. Graphlab is another Python library built to efficiently handle large datasets, however Dask was chosen due to it's similarity to Pandas, being able to read and manipulate dask dataframes in a similar, although more limited, manner to Pandas dataframes.

Matplotlib [27] was another library utilised, and functions to simplify the visualisation of the data. This was used to assist in plotting the confusion matrices, ROC curves, and precision-recall curves, allowing for the results to be easily compared and evaluated. Many other visualisation libraries in python are built on top of matplotlib, or use it in some way to assist with data analysis. Due to being integrated with the Pandas plotting function, Matplotlib was the first choice in efficiently plotting the results. The slugify library was also used to facilitate the naming of files. It would condense the given classifier name strings into lowercase strings without special characters, allowing them to be used as filenames, also known as a slug in RESTful API's. This helped when automatically naming the different graphs that were plotted with matplotlib, and saved the creation of a specific regular expression.

The final library used, and the most important, was Scikit-Learn [23]. This library provides support for a wide range of classification, regression and dimensionality reduction algorithms, and is generally considered one of the best libraries for data modelling and classical machine learning algorithms. The extensive selection of classification models offered by this library presented a variety of results to compare and evaluate, broadening the scope of this research project, There are also multiple methods of feature extraction available, extensive tuning methods for the hyperparameters, and multiple evaluation metrics available. Additionally, Scikit-Learn contains integration with both Matplotlib and Pandas, simplifying the process of both managing and visualising the data. Scikit-Learn was integral to the functionality of this research project, and expedited the creation and increased the functionality of feature-extracted textual data, and the classification models and metrics said data was used with.

Two more libraries were considered for use, these being Tensorflow [22]and Keras [30]. These libraries are designed for developing deep neural networks, which can be used for a variety of purposes, including the classification of fake news with feature extracted text articles. Kaliyar et al. [31] developed a deep Convolutional Neural Network utilising discriminatory feature extraction at multiple hidden levels of the neural network, achieving accuracy results of 98.36% on their test data. These are extremely promising results, and show the potential of fake news classification using neural networks. The use of Tensorflow and Keras to develop neural networks in a similar manner could provide an ideal comparison benchmark, and would be an ideal topic for further research and evaluation, although would not show results as positive as Kaliyar et al [31]. Unfortunately, the use of these libraries were beyond the scope of this project, and would not be used due to time limitations. The choice was made to focus exclusively on models available within the Scikit-learn library.

2.4 Feature Extraction

Feature extraction is the first step of training after loading the dataset and splitting the data, which involves reducing the raw data (words) into more manageable groups for processing, which will save on computing resources and allow for large amounts of data to be parsed on personal computers. As explained in the dataset selection section, the datasets are split into four categories; ID, type, content, and title. (Three with the liar dataset, as the title would also be classifiable as the content). The content, being the body of the article containing the actual text content of the news, is what is focused on for Feature Extraction.

Before continuing, it is best to elaborate on what was previously mentioned about the text classification and feature extraction process, which will give context to how the project will ultimately work. From a linguistic standpoint, there is a substantial difference in the methodology of writing fake news, compared to writing real news. Fake news is created for a number of reasons, but ultimately all, with the exception of humorous fakes, are written to influence the consumer reading it. To be able to draw people in to stories that aren't true, and that aren't supported by facts or verifiable, this necessitates the use of different language to attract readers. For example, look at figure 3 on the next page.

3951	Germany's Merkel backs tighter refugee rules amid sex assault protests	As demonstrations erupted in Cologne on Saturday over New Year's Eve sexual assaults and robberies blamed largely on foreigners, Chancellor Angela Merkel called for stricter laws regulating asylum seekers. Merkel, who has been particularly outspoken in welcoming refugees to Germany, told a two-day meeting of the Christian Democrats in Mainz that tighter restrictions would be "in the interest of citizens, but also in the interests of the great majority of the refugees who are here, Deutsche Presse-Agentur reported. "When crimes are committed, and people place themselves outside the law ... there must be consequences," she told reporters after the meeting, the BBC reported. Party leaders agreed on a proposal to strengthen the ability of police to conduct checks of identity papers, and also to exclude foreigners from being granted asylum who had been convicted of crimes and sentenced to terms even as light as probation. The measures would require approval by parliament. The sharper tone follows reports in Cologne that some 1,000 men, many intoxicated, robbed, sexually assaulted and in some cases raped women during celebrations on New Year's Eve. "Refugees, asylum seekers, migrants, foreigners, friendly or evil, new or long-time residents: It doesn't matter," the news magazine Der Spiegel said in an editorial. "It seems as though the time has come for a broad debate over Germany's future and Merkel's mantra 'We can do it' is no longer enough to suppress it." Police initially identified the suspects as up to 1,000 men "of Middle Eastern origin, but later backtracked as public officials cautioned there was little information on whether those involved were migrants. Of the 32 suspects identified by police in Cologne, 22 are asylum seekers, the German Interior Ministry said. One suspect was carrying a document with Arabic-German translations of sexist phrases and threats, which mass-circulation tabloid Bild published on Saturday. As the uproar over the assaults gathered strength, thousands of demonstrators, from the left and right, turned out in Cologne for a day of protests Saturday. The protesters included the anti-Islamic PEGIDA movement and the right-wing extremist Pro Cologne party. At one point, police fired tear gas and used water cannon to break up a PEGIDA rally after protesters threw firecrackers and bottles at officers, Agence France-Presse reported. One group carried signs saying, "No violence against women." Another banner read, "Migrants out." Demonstrators from the political left chanted, "Say it loud, say it clear, refugees are welcome here," Buzzfeed reported. Some 1,700 police, many in riot gear, were on hand to control the crowds, the DPA news agency reported.	REAL

6304	Set Staff Horrified At What Hillary Is Caught Doing After Brutal Interview	Set Staff Horrified At What Hillary Is Caught Doing After Brutal Interview Posted on October 27, 2016 by Amanda Shea in Politics Share This Set of the Commander-in-Chief Forum (left), Hillary Clinton at the Forum (right) It was only a matter of time before the glue that keeps Hillary Clinton together cracked and her true evil self came out from behind her forced smile. Looking visibly uncomfortable in front of the camera during a particularly brutal interview, Hillary exited stage left as quick as she could when a horrified staff member caught what she did backstage. Hillary probably felt safe going into the interview by liberal-leaning Matt Lauer, who conducted a televised question and answer session with the Democratic candidate in September at the Commander-in-Chief Forum. As she's apparently accustomed to, Hillary allegedly received a list of questions before the event, so she could prepare her answers (lies) and prevent being caught on the spot to come up with what to say, making her come across flawlessly to the public. Thinking everything was worked to her advantage as is typically the case, Lauer threw the candidate a curve ball that Hillary couldn't hide from. According to The Watch Towers, a Comcast official (the parent company of NBC Universal) stated that Lauer went rogue when he asked one legitimate question about the FBI investigation concerning her homemade server and the unsecured emails. The alleged informant said at that point "we could see she (Hillary) was beginning to boil." It's evident by her expression and tone caught in the recording of the Forum interview that Hillary wasn't happy about being forced to answer a question she wasn't prepared for. Holding her rage inside the best she could, the person claiming to be a producer on the set, accused Hillary of launching an explosive verbal assault on her staff backstage after the interview. "She was in a full meltdown and no one on her staff dared speak with her, she went kind of manic and didn't have any control over herself at that point", the anonymous informant claimed in an unconfirmed report by Watch Towers . Hillary proceeded to pick up a full glass of water and throw it at the face of her assistant, and the screaming started", the description of the alleged events also said. The staffer claimed that they heard Hillary turn her rage on Donald Trump, saying, "If that f ***ng bastard wins, we all hang from nooses! Lauer's finished and if I lose it's all on your heads for screwing this up." While the reports of this incident have not been authenticated, it definitely doesn't seem out of character for Hillary, and with what happened to Lauer after the interview certainly shows that the Democratic candidate complained. Matt Lauer was massively criticized for the rest of the week on air by the Clinton campaign and the rest of the MSM as having conducted ˜an unfair and partisan attack on Clinton" Watch Towers reported. As we saw in the court hearings over Benghazi, Hillary is a loose cannon when being called out on her illegal, dishonest, and deadly actions that she's not used to being confronted on. Whether this explosive tirade happened or not " to any degree of severity" her angry actions against the American people and our heroes she disrespects, speak volumes about who she is, which isn't someone we need leading our country.	FAKE

Figure 3: Shows the content of one randomly selected real news article and one randomly selected fake news article, from the "Fake or real news" dataset

This shows the significant difference between the content of the fake news articles and the real articles. The fake article uses incendiary language to start the article, motivating a strong emotional reaction from the reader and making them more likely to share the information. The article also uses unconfirmed reports presented as fact, at least for the first half of the article. The article does later state that these reports are unconfirmed, but does this after the cutoff point, or fold, that the Nielson Group mention in this study [32]. According to Fessenden, 80% of the user's time is spent on the initial page, before having to further scroll. Past this point causes User Attention to be lost, meaning that the amount of information that users quickly scan, rather than actually read, increases. Eyetracking research from Google [33] supports this, positing both scanning in an F-Shaped pattern, as well as a “Golden Triangle” where the users eyes and attention is usually focused. It also means that the reader is more likely to close the article without reading the full extent of it, and that the reader is more likely to share information without closely studying the content. Talwar et Al. [34] suggest five reasons why readers do this, linking to; Online Trust, Self-Disclosure, Social Comparison, Fear of Missing Out, and Social Media Fatigue. This article preys on different aspects of this to attempt to get the user to share it, with the incendiary language using aspects of Self-Disclosure and Social Comparison, whereas the unconfirming of reports using the reader’s social media fatigue against them, which is an increasingly more common issue in recent times.

The Real News article, however, uses more formal language, substantiated sources, and generally follows the commonly seen journalistic style of prose. The Linguistic differences between these two articles are what is used to define the difference in classification between Real and Fake News, for this project’s scope at least. The Identification and classification of these differences is what classifies it as natural language processing, with the identification and extraction of these features from the textual data presented being what allows it to be defined as Feature Extraction.

2.4.1 Preprocessing

The initial step for Feature Extraction in this process involved preprocessing of the textual data. This involves performing operations on the textual article content from the dataset to transform the text data into a form more appropriate for the machine learning classifier models, which is a simple method of increasing performance of the models. There were three methods of text preprocessing used in this project, which were:

2.4.1.1 Normalisation

This involves modifying the complete dataset to be transformed into a single, canonical form. Usually, this involves mapping near-identical words to be considered in the dictionary as the same word, which was not used in this project due to the difficulty of implementation with a large, unknown dictionary of words extracted from the news article datasets. What was used was a simpler method of normalisation, involving mapping all of the letters to lowercase. This allows for the text to be registered in the dictionary regardless of case-sensitivity, and is great for increasing the population of sparse datasets. It is a useful general purpose method of preprocessing, and should be standard practice for the majority of text classification problems.

2.4.1.2 Noise Reduction

Noise Reduction is probably the simplest type of text preprocessing, and involves the removal of extraneous characters, text segments, spaces, digits, and punctuation, which can impact the performance of feature extraction. As an example, noise reduction was used on the textual data presented in the datasets here to remove the URL from datasets, which would be useless information to a feature-extracted dataset, and could cause the mistaken word commonality of ‘http’ or ‘www’, which could impact the performance of the classification models. It was also used in the project to remove extraneous spaces, punctuation, and special characters. This was done via the use of regular expressions.

2.4.1.3 Stop Words

Stop words are the final method of preprocessing used in this project, involving the removal of commonly used words in a language, which would not add anything to the project and might just confuse the classifier. For the English language, these would be words such as ‘a’, ‘the’, ‘but’, ‘or’, ‘is’, or ‘are’. Scikit-Learn includes its own list of designated stop words, which are passed to the vectorizer upon creation, meaning that these stop words are ignored during feature extraction and removed from the dictionary. The Scikit-Learn list numbers to 318 separate words, and is generally an imperfect solution to a highly-specific problem containing controversial words and surprising omissions, as critiqued by Nothman et Al. [35]. The decision to use this list was decided after viewing, as the issues with the list were seen as negligible for this project, and the simplicity of inclusion was another notable advantage.

2.4.2 Bag Of Words

The Bag of words model is the most common method of feature extraction, and is used to represent text data with a numerical vector. This can significantly cut down the computational resources required to store this data, as computers are generally better at handling numerical data than textual data.

Initially, a list is created out of the words listed in the dataset’s content. For a news article, that would be the body of the article itself. For this list, or dictionary, each unique word is categorised with a unique vector, which is used to identify each word later when the dictionary object is queried. This is called vectorization. The structure of the words in the dictionary is not categorised, simply their presence. If the content of two documents are similar, it can be reasonably assumed that the documents themselves are similar. Meaning if the list of vectors representing two separate news articles are relatively similar, it can be reasonably assumed that the classification of the two news articles is also similar. Due to large vocabulary sizes, particularly as the number of articles in the dataset increases, the length of the dictionary vector can exponentially increase. This causes the dictionary vector to naturally contain lots of 0 scores, logically becoming a sparse dataset.

Once the Bag of Words is utilised and a dictionary created, there needs to be a method of scoring the vocabulary to describe the occurrence of words within the document. This is how the words are actually converted into the vectors needed. Two methods of this have been used, with their results to be compared

2.4.3 Count Vectorizer

The Count Vectorizer works simply on term frequency. This involves counting the number of times that the individual words appear in each article. Each word is known as a Token. The occurrences of tokens is recorded, and a sparse matrix of articles (also known as documents when referring to feature extraction) × Tokens is created. This is then what the classifiers use to be able to classify the documents.

2.4.4 TF-IDF

Term Frequency – Inverse Document Frequency, or TF-IDF may also be used. Term Frequency scores the frequency of the word in the current document, while Inverse Document Frequency scores the rarity of the word across all documents. This offers a natural filtration for common words such as “the”, “of”, and “and”, penalising these common words, and serving to highlight distinct words across the documents. This offers another option other than the bag-of-words approach and allows for further testing and comparison to see the most optimal strategy. The importance of each word increases according to how many times it appears in the document, but the importance is then offset by the inverse document frequency. This is a method of weighting the word scores. TF is divided by the document length to normalize the frequency. A max differential threshold of 0.7, or 70% was also used, which eliminates words that appear in more than 70% of all documents. This could have been tested more with different thresholds, but 70% managed a decent job as a baseline.

$$TF = \frac{Number\ of\ appearances\ of\ the\ word\ in\ the\ document}{Number\ of\ words\ in\ the\ document}$$

$$IDF = \log_e \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ specific\ word\ in\ it}$$

$$TFIDF = TF * IDF$$

2.5 Models

The classification models described here are all part of the Scikit-Learn library, and were chosen due to being strong supervised learning methods,

2.5.1 Multinomial Naïve-Bayes

This is generally one of the simplest and most intuitive classification algorithms, although is one that tends to work well in a majority of cases, especially given it being one of the "Classic" classification algorithms. The multinomial naïve-bayes classifier works according to Bayes theorem of probability, which generally operates on the principle that every classified feature is an independent value, that is each feature contribute independently to the probability of classification. This principle is relatively strong when it comes to text classification, due to the encoding of text data as a series of vectors. The generic Naïve Bayes algorithm works well for classification, however the multinomial naïve bayes' probability selection follows multinomial distribution methods, which works well with the word count and TF-IDF vectorizers, improving performance over the standard naïve-bayes theorem.

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

With Probability P, A representing the vector representation of article content, and B representing the class it is being tested with. The goal therefore, is to find the probability of A belonging to a certain class B.

Given a basic news sentence, such as "Soundgarden Releasing New Album", and given a training set of 200 articles, 100 real and 100 fake, with 25 real articles mentioning Soundgarden, 50 real articles mentioning a new album, 45 real articles with the word "Releasing", 50 fake articles mentioning Soundgarden, 60 fake articles with the word "Releasing" and 75 fake articles mentioning a new album, the naïve bayes classifier would look like this:

$$\begin{aligned} &P(\text{"}Soundgarden\ releasing\ New\ Album|REAL) \\ &= \frac{P(Soundgarden|REAL)\ \times P\ (Releasing|REAL)\ \times P(New|REAL)\ \times P(Album|REAL)}{P(REAL)} = \frac{0.25\ \times 0.45\ \times 0.5 \times 0.5}{P(REAL)} \\ &= 0.028125 \end{aligned}$$

Which would then be compared against the probability of the news sentence being fake

$$\begin{aligned} &P(\text{"}Soundgarden\ releasing\ New\ Album\text{"}|FAKE) \\ &= \frac{P(Soundgarden|FAKE)\ \times P\ (Releasing|FAKE)\ \times P(New|FAKE)\ \times P(Album|FAKE)}{P(FAKE)} = \frac{0.5\ \times 0.6\ \times 0.75 \times 0.75}{P(FAKE)} \\ &= 0.16875 \end{aligned}$$

As 0.16785 > 0.028125, the article would therefore be classified as fake.

2.5.2 Linear SVC

Linear SVC (Support Vector Classification) is a type of Support Vector Machine, and is a type of classifier that works well in high-dimensional spaces. Linear SVC was used over the regular SVC provided in Scikit-Learn, due to the choice of Pandas dataframes to store the data. Pandas stores large datasets as sparse data, which regular SVC's cannot handle within Scikit-Learn, although linear SVC's can. This does limit it to the choice of a linear Kernal, although this is ideal for the large dataset of the Fake News Corpus anyway.

Support Vector machines work on the concept of representing data as points in space, which as the textual data is represented as vectors after feature extraction, makes it well suited to this classification problem. As points (Vectors) on this virtual space are classified as Real or Fake based on the training data it was given, they are naturally grouped with others of the same classification. SVC's create a hyperplane, which functions as a dividing line between the classes, which works to linearly separate and classify the data. The further away from the hyperplane a point is, the more certain the SVC is that it has been classified correctly. If a clear hyperplane cannot be made in a dataset, the SVC maps the data to a higher dimension, and attempts to create a hyperplane in that dimension if possible, and will continue until a hyperplane can be made. It therefore works to find the optimal hyperplane, which has the furthest distance from the nearest data points, which functions as an optimisation problem.
Lorent [36] mathematically defines it as:
"Given pairs of features-label $(x_i, y_i), y_i \in \{-1, 1\}$, it solves the following unconstrained optimization problem.

$$\min_{w} \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + \mathbf{C}\sum_{i=1}^{l} \xi\ (\mathbf{w}; x_i, y_u)$$

Where ξ is a loss function, in this case $L2$ loss function has been used, and $\mathbf{C} > 0$ a penalty parameter. Class of new examples are assigned by looking at the value of $\mathbf{w}^{\mathbf{T}}\mathbf{w}$. The class 1 is assigned if $\mathbf{w}^{\mathbf{T}}\mathbf{w} \geq 0$ and the class −1 if $\mathbf{w}^{\mathbf{T}}\mathbf{w} < 0$."

This equation is just a mathematical notation for the explanation given before. It attempts to optimize the set of data $(x_i, y_i), y_i$ via cutting through a dimensional hyperplane $\min_{w}\frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w}$, whilst reducing ξ loss function and $\mathbf{C}$ penalty as much as possible, before inevitably classifying the data at the end of the function as either 0 or 1 $(\mathbf{w}; x_i, y_u)$ depending on the result of $\mathbf{w}^{\mathbf{T}}\mathbf{w}$

2.5.3 K-Neighbours Classifier

K-Neighbours classifier is another machine learning classifier that, like Linear SVC, represents data as points in space. This is much simpler than the Linear SVC, though, and operates under the assumption that the vectors in the virtual space are naturally grouped with other vectors classified as the same label. Generally meaning that the distances between points has much more importance with this classifier. Classification is done via a majority vote of the k nearest neighbours of each point, due to uniform weights of the neighbours being chosen, with k as 5 throughout this project based on the hyperparameter tuning. The point to be queried is decided by the data class with the most representatives within the nearest neighbours of the previous point. This generally does not perform well with large or complicated datasets due to it's simplicity, and high levels of performance are not expected from this model.

2.5.4 Ridge Regression Classifier

This classifier works differently to the other classification models, as it treats the problem as a regression problem. Regression problems differ to classification problems, as they examine the influence of independent variables on dependent variables. The Dependent variable is the variable to be understood or predicted, whereas the independent variables are the information that should impact the dependent variable. This is a type of linear regression, so only one independent variable is used to explain or predict the dependent variable. The use of a binary (-1 to 1) threshold as determined by scikit-learn is what quantifies it as a classifier. Similar to Linear SVC, it functions as a linear classifier defined by a hyperplane separating the points of data. Ridge regression is, however, better at avoiding overfitting due to regularisation of the data. A penalised least squares loss function is one of the most unusual methods of data classification, and this model was chosen out of interest of the performance, although it is expected to be similar to LinearSVC in performance.
As defined by the Scikit-learn API [23]:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

With a positive predicted class if Xw is positive and a negative predicted class if Xw is negative.

2.5.5 Decision Tree Classifier

Decision trees are another unusual method of classification, functioning as a type of non-parametric model. They are flowchart-like structures, consisting of both nodes and branches. It is essentially just a massive series of if…else decisions. The values of the features are split into two groups at each node, with each group on a separate branch. At the end of each branch is another node, with the ultimate goal involving the repetition of this until the groups are separated as much as possible to a single classification. These final classification nodes are known as leaf nodes. These are a simple algorithm to program and utilise, and are easy to interpret and use, usually having good results for classification. They do, however, often take a long time to actually run, and suffer with higher dimensional data such as the data used in this project.

2.5.6 Forest of Randomised Decision Trees

The Forest of Randomised decision trees is an ensemble classifier, functioning extremely similar to decision trees. The main difference is that multiple decision trees are used on a subset of the training data. The decision tree estimators are built independently, and their results averaged. This usually gives improved performance when compared to standard decision trees, due to their variance being reduced

2.5.7 Gradient Boosting Classifier

Gradient Boosting classifiers are another ensemble classifier, taking advantage of the use of multiple decision trees to create a more powerful final classifier. The idea for gradient boosting initially came about from seeing if a weak learner could be modified to become a better learner [37]. An initial decision tree is first fitted to the data. A second model is then built, with the target outcomes set based on the information from the last model, to attempt to minimise the overall prediction error. The prediction error is a loss function, with the minimisation of such being the goal of optimisation for GBC. Logistic regression is the loss function to be optimised via Gradient boosting, due to being a binary classification problem. Another variable is the weak learner to be used, which is limited to decision trees in Scikit-Learn. Decision trees would be the ideal choice anyway, as the Gradient Boosting requires that the weak learners are to remain week, but can be constructed greedily as an additive model.

2.6 Hyperparameter Tuning

Hyperparameter tuning, or hyperparameter optimisation is an important part of machine learning algorithms, involving the modification of parameters to achieve the optimal result from the classifiers. Hyperparameters are like regular parameters, although ones that are set before the learning process begins. They are passed as arguments to the constructor of the model. Generally, there are a lot of different options for each classifier, which can cause difficulty for inexperienced data scientists when attempting to utilise the optimal configuration settings for classification models. Luckily, Scikit-Learn includes methods for Grid Search, a traditional method of optimising the hyperparameters.

To utilise grid search, a list of different possible hyperparameters is created, and sent to the grid search method along with the classification model itself. The Grid search works via searching through all possible combinations of these different hyperparameters, eventually returning the optimal combination of hyperparameters for the problem, dependent on which evaluation metric is selected to optimise.

Grid search is usually done on a validation set of data, which is a small subset of the dataset that is often created when the training and test data are split. For this project, the grid search was done on the full dataset, in order to truly show the optimal configuration settings so that the classifiers can be compared properly.

The hyperparameter combinations that were used for grid search are located in the appendix, as well as the optimal combinations that were eventually decided upon.

2.7 Evaluation Metrics

The choice of metrics is an important one in any machine learning task. The type of metric chosen can affect how the performance is evaluated, with the functionality and effectiveness being measurable in a number of different ways. As a classification problem, many of the methods commonly used for monitoring and measuring the performance of these models were used, which would work to show a complete representation of each models performance. These were provided via the Scikit-Learn library, which handily includes metrics for each of the classification models available.

As a binary classification problem, positive and negative refer to the prediction of the classifier (Whether the news is classified as real or fake by the model), with true and false referring to the conformity of the classifier model's prediction with the true classification of the subject (Whether the news article is actually real or fake). As it is binary classification, meaning that the option is between two separate classes (Real or Fake), either option can be considered as the positive or negative depending on judgement. For the rest of this section to aid with the explanation of the different metrics, the classification of the Real News will be used as the positive, although these options could easily be substituted (Correctly classified real news being the true Positive, with Correctly classified Fake News being the true Negative).

2.7.1 Accuracy

Classification accuracy, also known as accuracy, is the most common classification metric, consisting of the ratio of the correct number of predictions to the total number of input samples. It would usually be multiplied by 100 to give a percentage

$$Accuracy = \frac{Correct\ number\ of\ predictions}{Total\ number\ of\ input\ samples\ (Predictions\ Made)} \times 100$$

Accuracy is used to generally show how often a classifier is correct. This is useful to initially gauge the performance of a model, but also isn't a complete evaluation of a model's capabilities. With imbalanced datasets, it is possible for models to prioritise picking one option over the other, which would skew and artificially inflate the accuracy rate even though the model would only be picking one option.

$$Accuracy = \frac{\boldsymbol{True}\ Positive + \boldsymbol{True}\ Negative}{\boldsymbol{True}\ Positive + \boldsymbol{True}\ Negative + \boldsymbol{False}\ Positive + \boldsymbol{False}\ Negative} \times 100$$

For this text classification program, the accuracy can be defined in simple terms as seen above. The True Positive and True Negative Rates are the results that are correctly classified as being real news and fake news respectively, whereas the False Positive and False Negative rates are the opposite, being the results that have been incorrectly classified.

$$Fake\ News\ Accuracy = \frac{Correctly\ Classified\ Real\ News + Correctly\ Classified\ Fake\ News}{\begin{array}{c}Correctly\ Classified\ Real\ News + Correctly\ Classified\ Fake\ News + \\ Incorrectly\ Classified\ Real\ News + Incorrectly\ Classified\ Fake\ News\end{array}}$$

2.7.2 Precision

Another important evaluation metric is Precision. This shows the proportion of True Positives divided by the total number of elements labelled as positive (Including elements incorrectly classified as positive, while actually being negative). It is generally used to determine how many of the positive identifications were correctly classified.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True}\ Positives}{\boldsymbol{True}\ Positives + \boldsymbol{False}\ Positives}$$

For this Fake News Classification problem, that would mean the number of real news articles correctly labelled divided by the total amount of articles labelled as Real news.

$$Precision = \frac{Correctly\ Classified\ Real\ News}{Correctly\ Classified\ Real\ News + Incorrectly\ Classified\ Fake\ News}$$

High Precision is a positive result, as it would signify that the algorithm would return much more of the relevant results, rather than irrelevant results. For this example, a high Precision would mean that the algorithm is effective at identifying Real News Articles.

2.7.3 Recall

Recall shows the proportion of True Positives divided by the total number of elements that correctly belong to the positive class. This is generally used to determine the proportion of the positives that were correctly classified.

$$\boldsymbol{Recall} = \frac{\boldsymbol{True}\ Positives}{\boldsymbol{True}\ Positives + \boldsymbol{False}\ Negatives}$$

Simplified, that would signify the number of correctly classified Real News articles, divided by the total amount of actual Real News Articles, which includes articles classified by the model as Fake News, while actually being Real news.

$$Recall = \frac{Correctly\ Classified\ Real\ News}{Correctly\ Classified\ Real\ News + Incorrectly\ Classified\ Real\ News}$$

High Recall signifies that the classification model would return most of the total correctly classified relevant results.

2.7.4 F1 Score

When discussing Precision and Recall, it is important to be aware of the natural trade-off between the two. Improving the precision for a machine learning model will decrease therecall for the same model, and vice-versa. For fake news classification, high precision will indicate less false positives, meaning that there were less fake news articles classified as real news, whereas it would similarly have a lower recall, showing that there were a significant number of real news articles that weren't included in the positive (Real) identifier. Low precision would therefore indicate that there were a greater amount of fake news articles incorrectly classified as real news, although there would be higher recall, signifying that there would be more real news articles included in the positive (Real) label.

For many machine learning tasks, either the precision or recall is focused on, dependent on whether relevant results or a full range of results is important. There are machine learning problems, such as black box optimization, where the choice of whether to prioritise precision or recall is not known, and others where a balanced incorporation of both precision and recall are important. One of the potential solutions to this issue is the F1 Score, which is a method of combining precision and recall into a single metric. F1 Score is the harmonic mean of precision and recall, with a higher score generally showing a better model, with the joint benefits of both high precision and recall. The contributions of precision and recall are equal. It is defined as thus:

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}\ or\ F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

There are other metrics of the F-measures series, which F1 is a part of, which offer different variances of the weighting of Precision and Recall, compared to F1 score's equal weighting of both metrics. These were ultimately left out of the project but may prove useful in further evaluation in similar studies, particularly with more imbalanced datasets.

2.7.5 Averages

Additionally, for the F1 Score, Precision, and Recall, multiple methods of averaging the data are used. These all show different aspects of the metrics, with all recorded for posterities sake and to show a complete overview of the performance of the classifiers. As defined in the Scikit-Learn API:

- The Micro Average "Calculates the metrics globally by counting the total true positives, false negatives and false positives"
- The Macro Average "Calculates metrics for each label, and find their unweighted mean. This does not take label imbalance into account."
- The Weighted Average "Calculates metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall."

These all offer different characteristics of the metrics to judge the results by, and were seen as a useful addition.

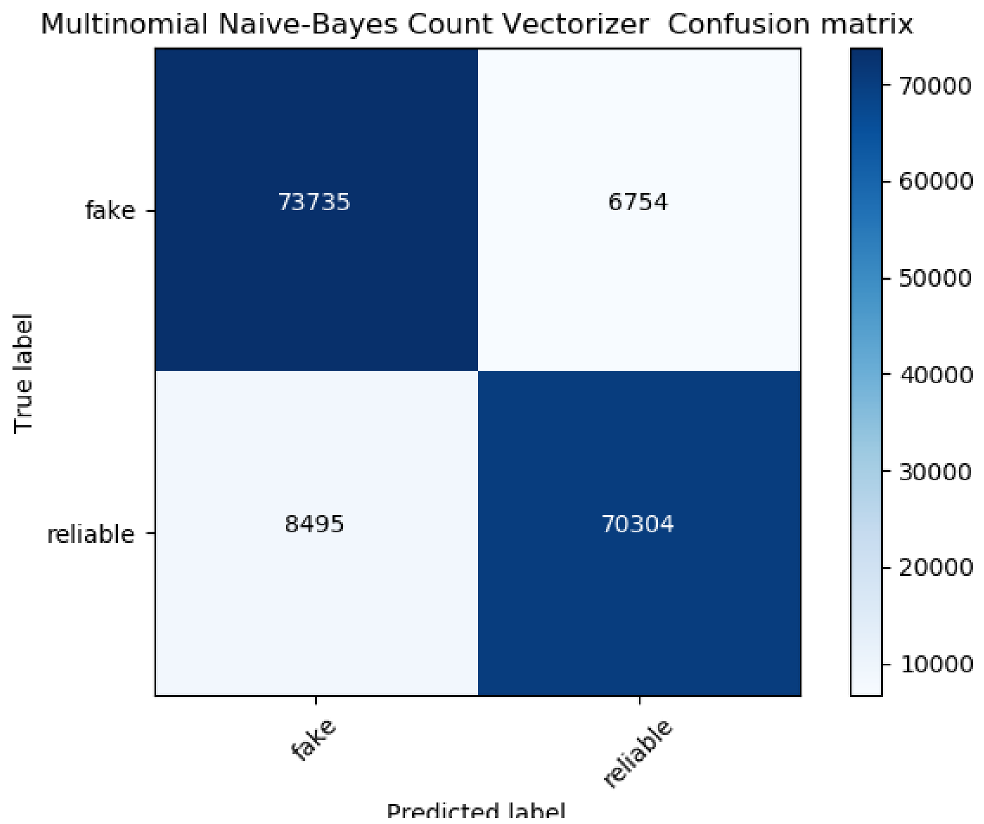
2.8 Visualising the data

Along with the evaluation metrics highlighted in the previous segment, scikit-learn and matplotlib offer multiple methods of data visualisation, which can facilitate the viewing and understanding of the data, and make it easier for humans to interpret the data. There are three main methods of doing this for classification problems in the scikit-learn API, and each are specifically formatted to be useful presentations of the specific metrics presented in the previous segment.

Tables, Box plots, Line Graphs, Scatter Charts, and Bar Graphs are also utilised to visualise data throughout this report. As these are common methods of data visualisation, it is assumed that the reader has previous experience in understanding these methods, so an in-depth explanation will not be provided.

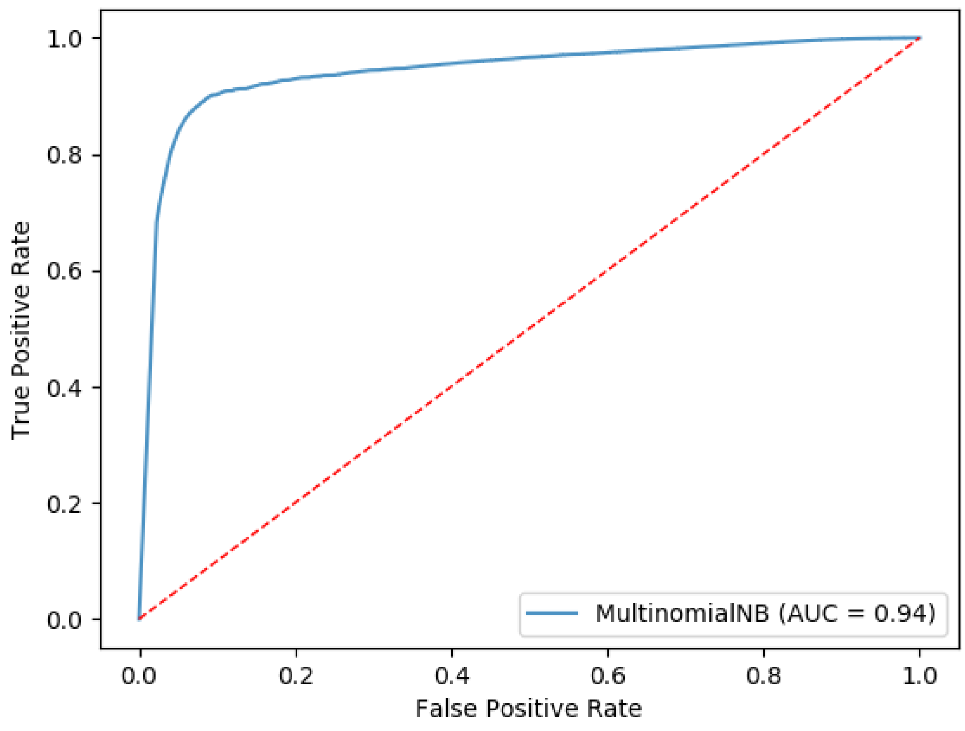
2.8.1 Confusion Matrices

The confusion matrix is a common metric for accuracy measurement and visualisation in classification problems. It is a simple way of displaying the different metrics used for Accuracy, those being True Positive, True Negative, False Positive, and False Negative as defined earlier. This allows for a simple method of visualising the accuracy of the classification model, as well as seeing the terms that the various metrics can be computed from.



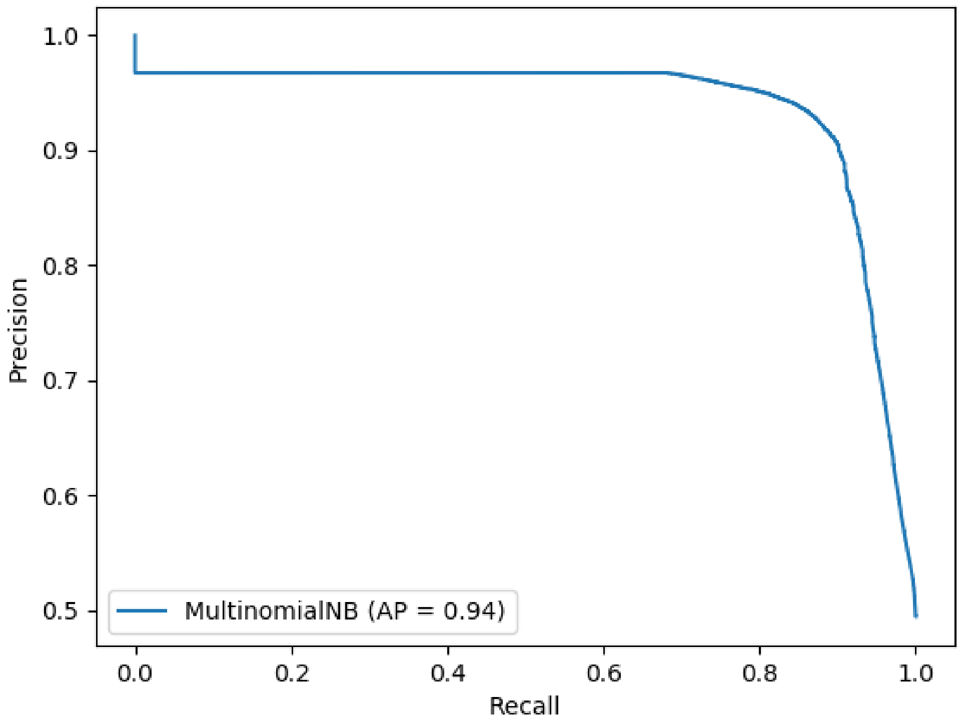
2.8.2 ROC Curves

ROC Curves, or Receiver Operating Characteristic curves, are another metric used for illustrating characteristics of Binary Classification problems. It plots a number of discrete points on a graph, with a True Positive Y axis and a False Positive X axis. Each discrete point represents a different threshold setting of the model, and determines a curve by joining these different points together. This allows measurement at different threshold settings of how much the model is able to distinguish between classes by determining the AUC, which stands for the Area Under Curve. The higher the AUC level is categorised as (Between 0 and 1), the better the model is able to distinguish between classes, or the better the model is able to distinguish between real news and fake news. Visually speaking, that means that the closer the classifier is to the top left corner, the better the performance of the classifier. A random classifier is also included as a baseline across the diagonal (True Positive = False Positive)


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
MultinomialNB (AUC = 0.94)

2.8.3 Precision-Recall Curves

Precision-Recall Curves are the final method of data visualisation, operating similarly to the ROC curve in that it measures discrete points at different thresholds and establishes a curve by connecting these different points. A major difference, though, is that the precision is plotted on the y axis, while recall is plotted on the x axis. Models with higher skill ceilings are closer to the 1, 1 end of the graph, or top right. Saito & Reimsmeier [38] argue for their usefulness for using models trained on imbalanced datasets due to lacking True Negatives, whereas ROC Curves are generally useful for balanced datasets. These are rough guidelines, though, and are more contextually dependent than anything elses. Precision-Recall curves are generally useful for visualising both the precision and the recall at the same time, and allows for proper threshold selection to find the ideal balance between Precision and recall for further tuning of the classification model.



3 Software Design, Implementation and Testing

3.1 Design

Many aspects of the design process and theory were covered in the other sections, so this will be used to explain the functionality of the code itself. The code consists of multiple python files, with each configured to focus on allowing classifiers to learn and predict a specific dataset, and some python files used to configure and modify the datasets to assist in classification. A file called Comp&Tune.py was also used to grid search the classifiers for hyperparameter tuning.

3.1.1 Classifier file Architecture

Three files were used for this, SKProcess.py for the corpus dataset, f_or_nProcess.py for the fake or real news dataset, and liarProcess.py for the liar dataset. These are generally similar, with some minor differences in the names of the classifiers based on the datasets column names.

These generally consisted of code to load the data from the dataset into a pandas dataframe. Regular expressions were then used on the data in these dataframes to preprocess the text data, as well as remove duplicates. Scikit-learn code was then used to split the dataframe into training and test sets of data, and then to configure two copies of both of these datasets for the Count Vectorizer and TF-IDF Vectorizer. There was also y_test and y_train datasets created, which contained the true classification label of each article to allow for blind testing.

The list of classifiers were then looped through, fitted, and their predictions evaluated with the datasets of each vectorizer. The data was gathered from these classifiers, saved to a .txt document, and confusion matrices, Precision-Recall curves, and ROC Curves were then visualised and saved based on the data from these classifier's predictions. These classifiers would loop through until the loop ended, then the program would finish.

3.1.2 Dataset management

A python file named Corpussplit.py was used, which would be used to load in the dataset from the fake news corpus, which was too large to realistically use on the machine learning classifiers. This file would load in the dataset and gather 500000 articles from said dataset, 250000 real and 250000 fake, which it would then save as a csv file. This was a much more manageable dataset to work with, and allowed for the information to be loaded and classified properly without any of the hardware issues that would arise with the original corpus dataset.

3.1.2.1 Hyperparameter tuning

Another python file named Comp&Tune.py was used. It would function similarly to the dataset classifier files, except it would load the data and use grid search with a number of different hyperparameters to optimise them. The data gathered from this file would be used to modify the hyperparameters of the classifier files, as it would be the optimal parameters for the classifier models.

3.2 Implementation issues

Generally, the only issues that could be found were issues with my personal hardware. This made it difficult to analyse the corpus dataset without eventually deciding to focus on a smaller subset of the data, as memory problems would cause issues with significant slowdown, and crashes. Otherwise, the only issues were with my personal understanding and breakdown of the fake news classification problem itself. It is a complicated topic, with many different ways of dealing with it. Attempting to understand how the classifiers worked past the basic level of scikit-learn implementation was difficult, and required a large amount of research. Understanding the optimisation and metrics was also problematic.

3.3 Testing

As a research project, testing was done quite unusually as compared to a regular project. Principally, the focus was on the results and on the evaluation of the results, so as long as they seemed to be within expected parameters, the results were okay. Generally, rather than a formal testing strategy taking place, a common-sense approach was used. If the results gained could be reasonably expected from the classifier, then the model was working as intended. This worked as an adequate way of testing the performance, and left the final project with no major errors.

Testing came as a result of running the models and analysing their performance. There were a few issues involved, such as the results for all classifiers being unexpectedly high when initially testing the fake news corpus dataset. This came as a result of the training data and test data being mixed, and containing a complete set of duplicates of the data within the dataset. This would naturally make those results useless, as the classifiers were trained on the same data that they would be tested on. Initial scans through the data before fitting the classification models were then added, curtailing this issue.

Some minor Unit Testing took place, involving the debugging of certain aspects using Pycharm's built-in debugging software. This allowed for testing of certain aspects to ensure that they performed as expected. This took place on the vectorizers, the models, and the methods used to save and print the information. There were generally no major issues reported with the Unit testing, other than some minor syntax issues.

Generally, the author's opinion is that the testing was good within the confines of this research-based project. More testing ideally could have taken place, but the major issue with testing the fake news corpus was found and solved, and there were no other issues with collecting of the results, the main focus of this project. The argument could be made that there also wasn't much to test, and as the components of the project worked as intended, the testing was done to an optimal degree.

4 Results and Conclusions

4.1 "Fake or Real News" Dataset results

4.1.1 Count Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.890961	0.906743	0.788618	0.759923	0.821616	0.871353	0.899091	0.884266
Weighted F1 Score	0.890718	0.906745	0.788599	0.758819	0.821673	0.871389	0.89912	0.884193
Macro F1 Score	0.890449	0.906626	0.788617	0.759269	0.821569	0.871336	0.899034	0.884247
Micro F1 Score	0.890961	0.906743	0.788618	0.759923	0.821616	0.871353	0.899091	0.884266
Weighted Precision	0.892375	0.906747	0.790892	0.759923	0.822362	0.872462	0.899406	0.888344
Macro Precision	0.893155	0.906613	0.789826	0.767693	0.82168	0.871647	0.898934	0.886845
Micro Precision	0.890961	0.906743	0.788618	0.759923	0.821616	0.871353	0.899091	0.884266
Weighted Recall	0.890961	0.906743	0.788618	0.759923	0.821616	0.871353	0.899091	0.884266
Macro Recall	0.889653	0.90664	0.78972	0.762775	0.822091	0.872064	0.899391	0.885869
Micro recall	0.890961	0.906743	0.788618	0.759923	0.821616	0.871353	0.899091	0.884266

From these results, it can be seen that the SGD classifier performed the best, with the highest scores in all evaluation metrics. SGD is an efficient and easy classifier to implement that depends on the hyperparameters being tuned correctly, which had of course taken place via grid search. Ridge Classification performed unusually badly in this, which is an unusual result, which may be due to bad tuning of the hyperparameters for this dataset, or simply not being trained as much as it should have been due to the lower total amount of articles in the dataset. Ensemble datasets and MNB performed well as expected, but the gradient-based SGD performed optimally in this case.

4.1.2 TF-IDF Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.851267	0.935438	0.527977	0.92922	0.803922	0.935916	0.899091	0.884266
Weighted F1 Score	0.84829	0.935456	0.40797	0.929219	0.803934	0.935933	0.899123	0.884264
Macro F1 Score	0.847421	0.935429	0.417089	0.92922	0.803695	0.935908	0.899047	0.884266
Micro F1 Score	0.851267	0.935438	0.527977	0.92922	0.803922	0.935916	0.899091	0.884266
Weighted Precision	0.872898	0.936587	0.761504	0.931676	0.803953	0.937109	0.89955	0.886601
Macro Precision	0.876054	0.935742	0.752632	0.930465	0.803659	0.936248	0.898995	0.885449
Micro Precision	0.851267	0.935438	0.527977	0.92922	0.803922	0.935916	0.899091	0.884266
Weighted Recall	0.851267	0.935438	0.527977	0.92922	0.803922	0.935916	0.899091	0.884266
Macro Recall	0.846662	0.93623	0.544321	0.930435	0.803738	0.936726	0.899494	0.885422
Micro recall	0.851267	0.935438	0.527977	0.92922	0.803922	0.935916	0.899091	0.884266

With TF-IDF, the K-Neighbours classifier performed extremely poorly. This may have been due to the small amount of vectors available for this, which may have threw off the neighbours distance metric due to being further away from each other. The incredibly low recall shows that not many relevant instances were actually received, meaning that a lot of articles were simply incorrectly classified, with the higher precision implying that many articles were simply classified into one category, rather than both. This also could explain the incredible performance of Ridge and Linear SVC, which probably did not have much difficulty in creating the necessary hyperplane, due to there being less vectors to work around. The ensemble classifiers performed similarly to the count vectorizer previously, and the decision tree would have ideally needed a higher depth to be able to further categorise properly. SGD , Ridge, and Linear SVC performed very well, with high marks in all metrics.

4.2 “Liar Liar Pants on Fire” Dataset results

4.2.1 Count Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted F1 Score	0.604862	0.590267	0.493011	0.59341	0.607343	0.59183	0.618109	0.563459
Macro F1 Score	0.593061	0.582197	0.504915	0.586076	0.600515	0.584563	0.607187	0.539615
Micro F1 Score	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted Precision	0.605355	0.590042	0.584441	0.593877	0.608121	0.592414	0.61836	0.603073
Macro Precision	0.599904	0.582306	0.570366	0.585939	0.600297	0.584413	0.613159	0.602387
Micro Precision	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted Recall	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Macro Recall	0.593443	0.582109	0.554623	0.586279	0.600913	0.584815	0.606961	0.56428
Micro recall	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493

The Liar dataset generally performed poorly, due to only being given the article title to classify. Given the limited dataset available, many of the classifiers registered a poor performance. Decision trees and MNB performed well due to their simplicity of classification, although Random forest had the highest accuracy and F1 Score, due to simply being an optimisation of the decision tree classifier.

4.2.2 TF-IDF Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted F1 Score	0.604862	0.590267	0.493011	0.59341	0.607343	0.59183	0.618109	0.563459
Macro F1 Score	0.593061	0.582197	0.504915	0.586076	0.600515	0.584563	0.607187	0.539615
Micro F1 Score	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted Precision	0.605355	0.590042	0.584441	0.593877	0.608121	0.592414	0.61836	0.603073
Macro Precision	0.599904	0.582306	0.570366	0.585939	0.600297	0.584413	0.613159	0.602387
Micro Precision	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Weighted Recall	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493
Macro Recall	0.593443	0.582109	0.554623	0.586279	0.600913	0.584815	0.606961	0.56428
Micro recall	0.611319	0.590512	0.520183	0.593009	0.606742	0.591344	0.623387	0.605493

With TF-IDF, it performed similarly to the count vectorizer. This is probably due to there not being that much of a dataset to really work with.

4.3 “Fake News Corpus” Dataset results

4.3.1 Count Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.904268	0.96534	0.865093	0.907601	0.946556	0.954811	0.955602	0.92921
Weighted F1 Score	0.904245	0.96534	0.865048	0.907589	0.946557	0.954812	0.955604	0.92918
Macro F1 Score	0.904223	0.965339	0.865013	0.907596	0.946553	0.954807	0.955602	0.929192
Micro F1 Score	0.904268	0.96534	0.865093	0.907601	0.946556	0.954811	0.955602	0.92921
Weighted Precision	0.904439	0.965402	0.865334	0.908165	0.946582	0.954814	0.955696	0.930448
Macro Precision	0.904522	0.965343	0.865426	0.907999	0.946543	0.954799	0.955624	0.930199
Micro Precision	0.904268	0.96534	0.865093	0.907601	0.946556	0.954811	0.955602	0.92921
Weighted Recall	0.904268	0.96534	0.865093	0.907601	0.946556	0.954811	0.955602	0.92921
Macro Recall	0.904141	0.965395	0.864931	0.907782	0.946587	0.954817	0.955671	0.929482
Micro recall	0.904268	0.96534	0.865093	0.907601	0.946556	0.954811	0.955602	0.92921

The Fake News corpus results were the strongest overall, with almost all of the classifiers achieving scores in all metrics over 0.9. These results are incredible, showing how the classifiers managed incredibly well with a large dataset, with a large amount of articles. SGD, Linear SVC, and Random forest were the highest-scoring, with all metrics scoring over 0.95. This is due to the classifiers being optimised to large, higher-dimensional datasets.

4.3.2 TF-IDF Vectorizer

Classifier	Multinomial NB	SGD Classifier	K-Neighbours Classifier	Ridge Classifier	Decision Tree Classifier	Linear SVC	Random Forest Classifier	Gradient Boosting Classifier
Accuracy	0.895435	0.947667	0.880908	0.962973	0.946412	0.970531	0.956488	0.931231
Weighted F1 Score	0.895396	0.947668	0.880856	0.962973	0.946412	0.970532	0.956489	0.931207
Macro F1 Score	0.895368	0.947667	0.880876	0.96297	0.946407	0.97053	0.956487	0.931218
Micro F1 Score	0.895435	0.947667	0.880908	0.962973	0.946412	0.970531	0.956488	0.931231
Weighted Precision	0.895738	0.947805	0.882071	0.962985	0.946417	0.970566	0.956611	0.93233
Macro Precision	0.895849	0.947718	0.881841	0.962958	0.946397	0.970521	0.956529	0.932095
Micro Precision	0.895435	0.947667	0.880908	0.962973	0.946412	0.970531	0.956488	0.931231
Weighted Recall	0.895435	0.947667	0.880908	0.962973	0.946412	0.970531	0.956488	0.931231
Macro Recall	0.895268	0.947751	0.881178	0.962993	0.94642	0.970572	0.956567	0.931487
Micro recall	0.895435	0.947667	0.880908	0.962973	0.946412	0.970531	0.956488	0.931231

Most of the classifiers performed slightly better with a TF-IDF vectorizer than with a count vectorizer, with the exceptions of MNB and SGD, which performed slightly worse. Linear SVC was the standout, however, reaching a high of 97% accuracy, which is an incredible result. Ridge Classifier also performed incredibly well, at 96%, a significant improvement of 6% with all metrics over the count vectorizer. The F1 score of these classifiers signifies how many results they managed to correctly classify, with the confusion matrices and ROC Curves of these classifiers showing their incredible performance

4.4 Results Conclusion/Analysis

The results of this project manage to show how the differences in the datasets used can show incredible variation in the performance of these classifiers, and highlight the importance of choosing the right classifier for the job, depending on the dataset selected. The strongest performance overall was the TF-IDF Linear SVC model on the corpus dataset, which shows the strength of this classifier on large sparse datasets, due to the higher dimensionality available. SGD and Linear SVC are datasets that can be recommended if trying to optimise the results on these datasets. Ensemble classifiers such as Random Forest and Gradient Boosting show a consistency in their results across all datasets, although the recommendation would be to avoid these if planning on optimisation, or if working within a limited time frame, due to the high cost of computation and time complexity they take to operate. MNB is what should be recommended instead for ease of use, consistency, and speed in that case. K-Neighbours is the consistent worst performing classifier, and would consistently take the longest due to having to brute-force the text classification problem. This can be useful in different circumstances, just not for fake news classification, or any text classification problem. There were also issues with precision with this classifier. Decision trees are another unusual choice, performing averagely across most of the results. They also take a longer time to actually run, although offer some simple usability due to being easy to understand and interpret, given that the full algorithm can be printed and seen visually. Ridge Classifier is another consistent choice that can be recommended, although not to the same level as Linear SVC and SGD. The Confusion Matrices, ROC Curves, and Precision Recall curves are useful for visualising this data for personal comparison. Generally, larger datasets are what should be preferred for this type of classification problem, although

5 Critical Evaluation

Generally, the experiment went quite well. The requirements of fake news classification showed a consistent result that allows for much analysis and comparison, although managed to allow recommendations for a number of different scenarios. The choice of many classifiers allowed a lot of choice for adjustment to specific datasets and circumstances, and allowed a complete appraisal of many different options. The design decisions chosen were practical and allowed for implementation of many different aspects of fake news text classification, and the choice and usage of the different python libraries facilitated this.

As previously mentioned, ideally comparison against neural networks utilising Keras or Tensorflow would have been an interesting idea. It is something recommendable for further study, but would have allowed for the study into an entirely new area of data classification. Further study into different types of automated fake news classification would have also been something that could have improved this project. A general overview of the different methods used was given earlier, but the use of another one of these methods could have allowed for a full fake news classification system, rather than just the study and comparison of a single part of it. Given more time and further research, it could have really added to this project, and provided a further full insight into the topic. Image classification is also another area of fake news that could have been utilised, as it would have shown a fuller visualisation of the topic.

Ideally, more time and detail could have been put into the results discussion, as well as showing a complete variation of the different metrics. The general similarity of the accuracy, precision, recall, and F1 score in the results didn't lend itself to much detail in specified discussion, which was unfortunate. It did manage to show how competently tuned and fitted classifiers can operate at a high level though.

As an aside, the current coronavirus crisis made this project exponentially more difficult. Working as a key worker during this crisis meant that I was unable to put aside as much time as would be ideal for a project such as this, although given that, I feel that I did well in creating and researching this topic, the report, and the programming required. This did mean that the initial life cycle method presented was unfortunately not kept to, which could have added a sense of structure that would have simplified this project and led to a better result overall. Working as an external student also meant that I was limited to my personal hardware, which definitely made the project more difficult, and led to a number of problems with implementation and running the classification models.

Overall, the requirements for the project were met, and I feel that the project was a success. There were notable limitations with it, but it was a genuinely interesting topic, and is something I would very much like to return to in future.

6 Bibliography

[1] Ofcom, Jigsaw Research, "News Consumption in the UK: 2019," 2019. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0027/157914/uk-news-consumption-2019-report.pdf. [Accessed 20 April 2020].

[2] G. Orwell, 1984, Paris: Hatier, 1948.

[3] A. Huxley, Brave New World, New York: Harper Brothers, 1932.

[4] New York Times, *Radio Listeners in Panic, Taking War Drama as Fact, Many Flee Homes to Escape `Gas Raid From Mars'--Phone Calls Swamp Police at Broadcast of Wells Fantasy,* New York, 1938.

[5] W. J. Campbell, Getting It Wrong: Debunking the Greatest Myths in American Journalism, Oakland, California: University of California press, 2017.

[6] V. L. Rubin, Y. Chen and N. J. Conroy, "Deception Detection for News: Three Types of Fakes," in *Association for Information Science and Technology Annual Conference*, St. Louis, Missouri, 2015.

[7] S. Ahmed, K. Hinkelmann and F. Corradini, "Combining Machine Learning with Knowledge Engineering to detect Fake News in Social Networks - A Survey," AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering, University of Camerino, Italy, 2019.

[8] A. K.Chaudhry, D. Baker and P. Thun-Hohenstein, "Stance Detection for the Fake News Challenge: Identifying Textual Relationships with Deep Neural Nets," Stanford University, Stanford, California.

[9] B. Ghanem, P. Rosso and F. Rangel, "Stance Detection in Fake News: A Combined Feature Representation," in *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, Brussels, Belgium, 2018.

[10] A. Acker, "The Manipulation of Social Media Metadata," Data & Society, Texas, 2018.

[11] G. Ciampaglia, P. Shiralkar, L. Rocha, J. Bollen, F. Menczer and A. Flammini, "Computational Fact Checking from Knowledge Networks.," *PLOS ONE 10(6): e0128193.,* pp. 1-13, 17 June 2015.

[12] Y. Wu, P. K. Agarwal, C. Li, J. Yang and C. Yu., "Computational Fact Checking through Query Perturbations," *ACM Transactions on Database Systems,* vol. 42, no. 1, pp. 1-41, 2017.

[13] S. Kumar, "Awesome Fake News Github Repository," 2020. [Online]. Available: https://github.com/sumeetkr/AwesomeFakeNews. [Accessed 12 February 2020].

[14] M. Szpakowski, "Fake News Corpus Github Repository," 25 January 2020. [Online]. Available: https://github.com/several27/FakeNewsCorpus. [Accessed 24 February 2020].

[15] J. Salminen, "Fake or real news Github Repository," 24 February 2017. [Online]. Available: https://github.com/joolsa/fake_real_news_dataset. [Accessed 17 February 2020].

[16] T. Fernandes, "Liar: A benchmark dataset for fake news detection," 23 April 2019. [Online]. Available: https://github.com/thiagorainmaker77/liar_dataset. [Accessed 25 February 2020].

[17] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Austin, Texas, 2010.

[18] Dask Development Team, "Dask: Library for dynamic task scheduling," 2016. [Online]. Available: https://dask.org. [Accessed 2 April 2020].

[19] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, Sebastopol, California: O'Reilly Media Inc., 2017.

[20] N. Diakopoulous, M. Bagavandas and G. Singh, "Interactive: The Top Programming Languages of 2019," IEEE Spectrum, 2019. [Online]. Available: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019. [Accessed 23 February 2020].

[21] H. Li, "Smile: Statistical Machine Intelligence and Learning Engine," 2014. [Online]. Available: https://haifengl.github.io/. [Accessed 2 March 2020].

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenburg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenburg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: http://tensorflow.org/. [Accessed 28 February 2020].

[23] G. V. A. G. V. M. B. T. Fabian Pedregosa, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher and M. Perrot, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[24] T. p. d. team, "pandas-dev/pandas: Pandas 1.0.3," Zenodo, 2020.

[25] T. E. Oliphant, "A Guide to NumPy," Trelgol Publishing, USA, 2006.

[26] S. v. d. Walt, S. C. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering,* vol. 13, no. 2, pp. 22 - 30, 2011.

[27] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[28] T. Elliott, "The State of the Octoverse: machine learning," Github, 24 January 2019. [Online]. Available: https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/. [Accessed 1 April 2020].

[29] Stack Overflow, "Developer Survey Results 2019," 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019. [Accessed 22 April 2020].

[30] F. Chollet, "Keras," 2015. [Online]. Available: https://keras.io. [Accessed 2 March 2020].

[31] R. Kaliyar, A. Goswami, P. Narang and S. Sinha, "FNDNet – A deep convolutional neural network for fake news detection," *Cognitive Systems Research,* vol. 61, pp. 32-44, 2019.

[32] T. Fessenden, "Scrolling and Attention," 5 April 2018. [Online]. Available: https://www.nngroup.com/articles/scrolling-and-attention/. [Accessed 3 May 2020].

[33] L. Granka, M. Feusner and L. Lorigo, "Eyetracking in Online Search," Google Inc., User Experience Research, Mountain View, California, 2006.

[34] S. Talwar, A. Dhir, P. Kaur, N. Zafar and M. Alrasheedy, "Why do people share fake news? Associations between the dark side of socialmedia use and fake news sharing behavior," *Journal of Retailing and Computer Services,* vol. 51, pp. 72-82, 2019.

[35] J. Nothman, H. Qin and R. Yurchak, "Stop Word Lists in Free Open-source Software Packages," *Proceedings of Workshop for NLP Open Source Software (NLP-OSS),* pp. 7-12, 2018.

[36] S. Lorent, "Fake News Detection Using Machine Learning," University Of Liège Faculty Of Applied Science, Liège, Belgium, 2019.

[37] M. Kearns, "Thoughts on Hypothesis Building," 1988.

[38] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLoS ONE,* vol. 10, no. 3, 2015.

[39] A. All-Tanvir, E. M. Mahir, S. Akhter and M. R. Huq, "Detecting Fake News using Machine Learning and Deep Learning Algorithms," in *7th International Conference on Smart Computing & Communications (ICSCC)*, Curtin University, Miri, Sarawak, Malaysia, 2019.

[40] S. Bird, E. Klein and E. Loper, Natural Language Processing with Python, Sebastopol, California: O'Reilly Media Inc., 2009.

7 Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the ’future conversations’. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

A. Third-Party Code and Libraries

If you have made use of any third-party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third-party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

Apache POI library – The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation **Error! Reference source not found.**. The library is released using the Apache License **Error! Reference source not found.**. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

Scikit Learn library – Facilitated most of the functionality of the project, as previously detailed

Dask library – Used to load large dataset, and split up into smaller readable chunks

Pandas library – Was what enabled the loading and modification of the datasets

Matplotlib – Assisted with plotting the visualisations

https://www.datacamp.com/community/tutorials/scikit-learn-fake-news - Tutorial that assisted with initially creating the classifiers, loading data, and plotting confusion matrices.

B. Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

17/02/2019
For your information, please find below a copy of your
recently completed online ethics assessment
Next steps
Please refer to the email accompanying this attachment for details on the correct ethical
approval route for this project. You should also review the content below for any ethical
issues which have been flagged for your attention
Staff research - if you have completed this assessment for a grant application, you are not
required to obtain approval until you have received confirmation that the grant has been
awarded.
Please remember that collection must not commence until approval has been confirmed.
In case of any further queries, please visit www.aber.ac.uk/ethics or contact
ethics@aber.ac.uk quoting reference number 12196.
Assesment Details
AU Status
Undergraduate or PG Taught
Your aber.ac.uk email address
wim12@aber.ac.uk
Full Name
William Minton
Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar
Please enter the aber.ac.uk email address of the person responsible for reviewing
your assessment
rrz@aber.ac.uk
Supervisor or Institute Director of Research Department
cs
Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
MMP Evolving robots within the Torcs open racing car simulator
Proposed Start Date
28/01/2019
Proposed Completion Date
03/05/2019
Are you conducting a quantitative or qualitative research project?
Mixed Methods
Does your research require external ethical approval under the Health Research Authority?
No
Does your research involve animals?
No
Are you completing this form for your own research?
Yes
Does your research involve human participants?
No
Institute
IMPACS
Please provide a brief summary of your project (150 word max)
This project involves the use of evolutionary algorithms to evolve self-driving cars
within the TORCS framework. This involves using evolutionary algorithms on basic
robots to develop automatically guided vehicles, as well as to optimise racing lines,
which will be compared and contrasted against other methods of intelligent control.
Where appropriate, do you have consent for the publication, reproduction or use of
any unpublished material?
Not applicable
Will appropriate measures be put in place for the secure and confidential storage of
data?
Yes
Does the research pose more than minimal and predictable risk to the researcher?
Not applicable
Will you be travelling, as a foreign national, in to any areas that the UK Foreign and
Commonwealth Office advise against travel to?
No
Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a
DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes
Declaration: Please tick to confirm that you have completed this form to the best of
your knowledge and that you will inform your department should the proposal significantly change.
Yes
Please include any further relevant information for this section here:

C. Additional information

The confusion matrices, ROC Curves, and Precision Recall curves can be found in the results sections of the technical submission. These notably assisted in visualising the data and allowing for said data to be easily explained.

Datasets used can be found on https://github.com/wim12/Datasets - The recommendation from my supervisor was hosting them on an external repository, as the file size is too large for regular submission

Annotations for the bibliography can be found with the technical submission