Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Explaining why news may generate caution

Piyush Mankad
[redacted]

September 7, 2020

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
MS Computer Science (Intelligent Systems)

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: Piyush Mankad Date: 07/09/2020

Abstract

With the advent of technology in our daily lives the proliferation of misleading information has become even more prominent through social media, blogs and online newspapers and thus in turn making us question the credibility and trustworthiness of this online generated content. Our research focuses on classification of fake news articles by extracting out useful text features and then explaining the predictions made by the model in human terms as to why a particular prediction was classified so. With this research we aim to provide a helping hand to the expert fact checkers already working in the domain.

Acknowledgements

I have received a great deal of support and assistance throughout the writing of my dissertation.

I would first like to thank my supervisor and my Professor Dr. Owen Conlan for providing me with his invaluable expertise and guidance in the domain and for believing in me even when i didn't.

I would like to thank Dr. Atif Quershi for sharing his valuable experience in the domain and equipping me with technical knowledge throughout this research.

Contents

List of Figures

List of Tables

1 Introduction

In today's day and age, being surrounded with technology at every step of our life and with most of the work being done over the internet. How much of the information that we consume over a period of day can we actually trust? Some people might say that you should opt for more trusted sources but trusting some certain organizations or even people for that matter is a choice. So even if the information is coming from a "trusted" source, how do we know about its correctness and not consider it as misinformation.

1.1 Motivation

In 1901, David Hanig published a paper that forever changed our understanding of taste (1). His research led to what we know today as the taste map, an illustration that divides the tongue into four separate areas as shown in figure 1.1 which classified the different receptors located at different spots on the tongue to capture different tastes like sweetness, bitterness, sour, salty etcetera.

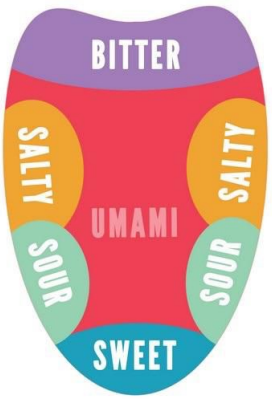

Figure 1.1: Tongue map

Since its discovery the tongue taste map has made its way into newspapers and textbooks taught in the school today. The only problem with the taste map is its false as a matter of fact it's not even an accurate representation of what Hanig originally discovered. The tongue map is a common misconception that is prevalent to this day. So where do misconceptions come from and what makes a fake fact so easy to believe? (1)

The misconception of the tongue map was not known to me before taking up this research topic and hence inculcated my interest in the domain of fake news. What are its features and

characteristics? How can you detect it? What does it look like and can we detect it with the help of modern day technology? And most importantly why do we believe it? Maybe because once we hear a convincingly good story it can be difficult to change how we see that information even in the face of new evidence. So one thing we should all do, the next time we hear a convenient story is to try to maintain a healthy skepticism because fake news can leave a bitter taste at every part of our tongue.

1.2 Research Question

It is a well-researched notion that lies spread faster than truth (2). A study carried out over the years on the popular social networking and microblogging website Twitter found that the top 1% of false news spread to over a range of 1000 to 100,000 people in comparison to the truth which rarely reached to over 1000 people. It was also noted that fake news spreads much faster than the actual truth. With false news spreading at such alarming rates has the potential to shape people's opinions based on a lie and too many people believing a lie can lead to chaos. Studies have shown that many Americans cannot tell what news is fake and what news is real (3). This can create confusion and misunderstanding about important social and political issues.

Fake news can have polarizing effects on people's opinions and beliefs. Disinformation and fake news trap people into believing lies and impairs them of making the correct judgement for themselves. Furthermore, it is done to make profits out of deceiving people.

Due to the aforementioned reasons we can ascertain that fake news detection is a novel task and through the medium of this research we would like to

1. How hard is it to detect fake news?
2. What are the various features of a fake news article?
3. And Explaining why a particular piece of news is said to be fake?

1.3 Data Preparation

When it comes to fake news there's really two ways for data preparation

1. Either you collect it
2. Or you create it

Since machine learning requires a lot of data for the training process and in this scenario most of the data needed to be fabricated from thin air, it made more sense to collect the data from various previously conducted researches.

Since we are collecting the data from the previously conducted research it is imperative for us to understand about the type of data we are looking for. Firstly, we need a dataset containing news articles that are spread across common topics most prevalent in the current scenario like Business, Politics, Education, Health, Science and technology, Entertainment, Sports etc. This type of dataset will help us in differentiating between the various categories and manifest the features that

separate news topics have, in addition to giving us an understanding of the nature of how news looks like in a real-world scenario. For this task we take up Washington Post's dataset (4) collected over the years and parse it into different categories. Washington post is an American daily newspaper that is published in Washington DC. With the first issue being in 1877, it has built its reputation over the years as a trusted news source with over a million consumers and hence is our trusted news source in this research.

Secondly, we need a dataset to compare against how the news varies in various degrees and lengths all around the globe. For the same purpose, we collect 3 articles each in the aforementioned common topics of news from reputed news agencies (Times of India, CNN, BBC, Al-jazeera) across different parts of the world. As news changes from place to place depending on the topics of relevance in that particular part of the world, we have tried to capture specifically those articles that hold relevance on an international level but on a more regional level the topics, events or individuals that those news agencies will talk about might not be the same. For example an Indian media company might not report the events taking place in the assembly elections of another country. For the same reason we must ensure caution while collecting such articles.

Next, we need a counterpart of the first dataset, not necessarily of the same articles that are covered in the Washington post dataset. We can go the traditional way of getting a dataset from the satirical news websites but the shortcoming is that generally these websites only publish in one domain. Another way of collecting fake news is through crowdsourcing the task of generating fake news from real news. We took a dataset (5) that has two types of fake news, the first type contains news that is generated from crowd sourcing techniques using Amazon mechanical turk where a small dataset is collected from legitimate news websites like CNN, FoxNews, CNET, NewYorkTimes etc belonging to different domains (Business, Politics, Education, Health, Science and technology, Entertainment, Sports) and was provided to the workers to generate a fake version of the news. Some contingencies like keeping the length of the news article short were put in place so that the writing style of the people and the journalists were comparable. The second type of news was collected from the entertainment domain with the general idea being that a lot of rumours are spread in this domain which also made it easier to find natural fake news that was actually spread over the masses. Due to the nature of this category, the legitimacy of the articles were evaluated using gossip-checking websites such as gossipCop.com and was cross referenced with information from other news sources.

1.4 Thesis Overview

The remainder of the thesis is organized as follows. Chapter 2 shows the background theories and techniques used in this thesis and discusses the related works. Chapter 3 sets the research scope and path taken. Chapter 4 gives you an in depth detail about the experiments done and the inferences drawn from them. Chapter 5 will discuss the final results reached. Chapter 6 draws the conclusion of the thesis.

2 Background and Related Works

2.1 NLP

With the advent of Artificial Intelligence into every subpart of the human world it is only natural for wanting Artificial Intelligence to understand and communicate with humans in their own language. Natural Language Processing or NLP for short is a field of study that deals with the interaction between humans and computers using natural language. However, NLP is considered a hard problem in the AI domain due to the nature of the human language and the various rules surrounding it that change with the change in language. Also, the rules surrounding the human language may seem abstract to the machine for example when someone uses a sarcastic remark to pass information. With all that in mind let us look at some of the techniques using which we can extract information and meaning from the human language.

2.1.1 Bag of words

The bag of words model (6) or BoW model is a technique of extracting information from the text that can be used to train a machine learning algorithm. A bag of words is a set of text representation that depicts the frequency of the words within the document. It usually involves a collection of words and the frequencies of those words as they appear in the document. We should note here that a Bag of Words model does not focus on the order of words or even how they are structured. Following this approach, a bag of words model is the simplest to use but it has its own limitations since we initially know the number of words, say 10 we know that we would have to make 10 vectors or columns for representing each word in the machine learning model but as the vocabulary size increases so does the vector representation of documents. With a very large dataset as that of the Washington Post itself consisting of thousands upon thousands of different words, the corpus size for the bag of words model increases multiple folds. One may argue that the same words spanning across different articles can be represented by the same vector but the chances of having a lot of common words are very thin. These sparse vectors are resource heavy and require a huge number of memory and computational resources and one thing you can do to make this machine learning process a viable one is to reduce your number of features. One way of reducing the number of features is by removing all the stopwords(commonly used words) from the text.

2.1.2 N-grams model

Another approach for reducing the number of features is by coupling some of the words which then is being represented by a single vector. In this approach, each word is called a "gram" and for the same reason if two words are combined in a pair then it is called a bi-gram model and if three words are combined in a group is called a tri-gram model giving this approach its name, the n-gram model. It should be noted that in a bi-gram model, a combination of two words for example "please turn" does not mean that "turn please" would also be identified as one of the kind. Computational linguistics define n-grams as "a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application" (7) (8). N-grams show a very promising approach for predicting the next word in a sequence in an (n-1) order Markov model. Even though the applications of using n-grams for predicting future words are great but there still remains the flaw of combining two unrelated words with each other only to expect them to occur again some time in the documents what's more is that these words might not even draw out any meaning or give you the context of the text (not all n-grams but most of them won't) plus with the help of maths we can deduce that let's say there are n words to be arranged in a bigram manner, with the help of permutation and combination we can say that it would turn out to be more than n bigrams even if you apply the grammatical rules, but what about increasing bigrams to trigrams and so on. Sure increasing is a way to go but it will greatly reduce the versatility of the features which after some length will only occur once in the entire document. Instead what would be useful is to combine two words that are similar in nature and will occur continuously one after the other in the same order almost every time one such example is your first name and surname and it also gives you context. An n-gram which has been identified as "Barack Obama" would surely tell you that the article is about the US president.

2.1.3 Parts of speech tagging

In order to identify the correct interlinked words occurring consecutively we need to first identify the hierarchy they hold in the linguistic space or simply the part of speech they belong to. This brings us to our next technique, parts of speech tagging which is a process of marking up a word in a text corpus as corresponding to a particular part of speech (9) which is helpful in differentiating between the various categories of words who also happen to have similar grammatical properties. The parts of speech hierarchy is divided into several different subparts but the 11 broad categories consists of noun, verb, adjective, pronoun, adverb, preposition, interjection, conjunction, article, numerical or determiner. One may think that identifying parts of speech is as simple as mapping the words to their corresponding parts of speech but actually it is more complicated than that because it is quite possible for a single word to have different art of speech tags in different sentences based on different contexts (10) (11). Parts of speech tagging is not a solution in itself but it opens up doors to utilise this feature as we see fit, for now returning to our idea of utilising interlinked words and as of now we can identify the parts of speech for two consecutively occurring words and check whether they belong to the same parts of speech. If they do then it can mean that either the intended speech is a part of a noun or is numeric in nature (as far as we can comprehend no two other speeches exist consecutively and be grammatically correct). By identifying the two consecutive

nouns we can combine them and use them as a feature, as we have done for Noun Phrases.

2.1.4 Named Entity Recognition

Named Entity Recognition (often referred to as entity chunking) is a specialized task of information extraction from text by identifying and categorizing key information into predefined categories such as name of persons, organizations, locations, expression of times, quantities, monetary values, percentages etcetera (12). The output of named entity tags can have applications like finding relevant web pages in the search engine and posting ads according to identified texts. A preview of this technique can be seen in the figure 2.1 where all the named entities are recognized and classified into their respective categories.

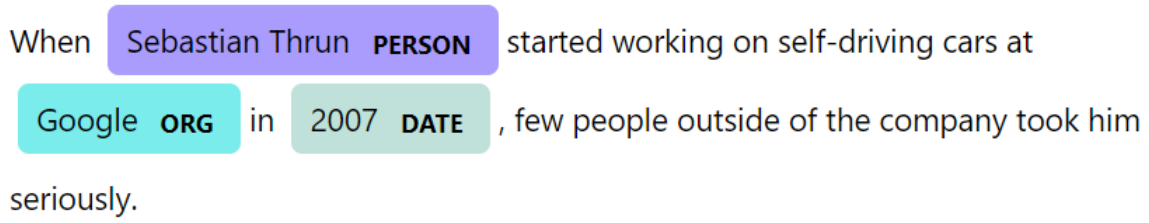

Figure 2.1: Example of named entity recognition

The Named Entity Recognition technique is good at generalizing data and may answer questions like:

- Which organisations have been mentioned in the text?
- Who is the article about?
- How much money has the organisation generated?

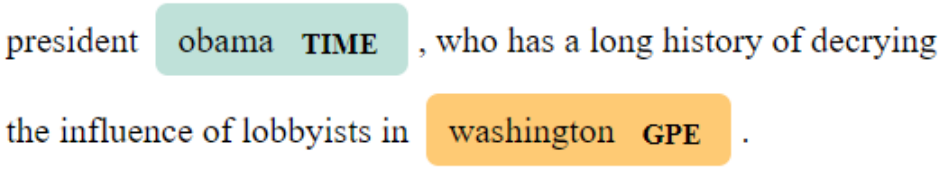

Figure 2.2: Example of named entity recognition from dataset

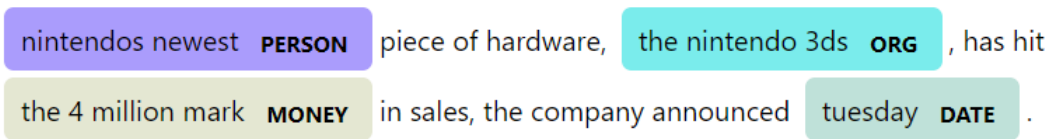

Figure 2.3: Example of named entity recognition from dataset

But using this technique in the process of fake news classification seems futile especially when such a technique has not been developed completely as you can see the misclassifications of person

categories of the two small lines extracted out from our dataset as shown in the figures 2.2 and 2.3 which covers both the scenarios of False Negatives and False Positive respectively.

2.1.5 Readability

The Readability index or Readability score tells us how technical a given manual is along with the education level required for the person to understand that documentation. Different Readability indices were introduced throughout the 19 th century and have found applications in many different areas like in the military where they are used for assessing the training manuals used throughout the training of the soldiers or in the modern day where the governments use these readability indices to impose constraints on the insurance policy makers to keep in check that their policy guidelines do not get too technical for the general public to not completely understand them.

Readability formulas usually look at factors like sentence length, average word length, word counts, number of syllables and word familiarity as a part of their calculations.Different readability scores will be discussed in the Experimentation chapter below.

2.1.6 Syntactic Analysis

Syntax refers to the arrangement of words in a sentence and Syntactic analysis is the process of analyzing natural language with the rules of grammar. Rules of grammar are always applied to some particular set of words and not individual words (13). It is important to for a sentence to be grammatically correct as it would directly impact the meaning the text is trying to convey, a grammatically incorrect sentence would fail to impart any knowledge that being said nobody in the fake news domain would likely make deliberate mistakes in their cooked up stories but we can still take the liberty to reach the conclusion that a general person will not be as well versed as a professional would be in the field of grammatical rules and is bound to make mistakes. By analysing the grammar of a sentence we are likely to figure out the difference between a professionally written article and the one that is not. Other ways for syntactic analysis include

- Stemming: This process simply reduces the words to their simple forms by chopping off the ends of words. For example "playing" to "play" but it may also reduce them to words that have no dictionary meaning like reducing "universal" to "univers".
- Lemmatization: It is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. It groups together words that have similar meanings and represents them as a single word.
- Word Segmentation: It involves dividing a large piece of continuous text into distinct units.
- Morphological Segmentation: It involves dividing words into individual units called morphemes. A morpheme is the smallest unit of meaningful language that cannot be further divided.

2.1.7 Sentiment Analysis

A sentiment analysis system combines the natural language processing and machine learning techniques to assign weighted sentiments scores to entities, topics, themes and categories within a sentence or a phrase to determine whether a piece of text is positive, negative or neutral in nature (14). Sentiment analysis is a useful tool for big brands to assess public opinion and product reach by analysing the tweets, comments and reviews left on their social media platforms in order to deliver useful insights and keep their brand image in check. In the domain of fake news as well, Sentiment analysis can play an important role by marking out extremely polarising news consisting of heavy language aiming to drive people's emotions and spread hate speech and reporting it as inappropriate. Sentiment analysis focuses on scrutinizing written texts for people's attitudes, sentiments and evaluations with analytical techniques (15).

2.1.8 Word Embeddings

Word embeddings is an umbrella term used for identifying natural language processing techniques that are used for representing the words and phrases in the vocabulary in the form of vectors of real numbers. Word embeddings are an essential part of NLP and required for converting text to numbers that can be easily understood and processed inside the machine learning algorithm. Next we look at some of the word embedding techniques:

- Count Vectoriser: For performing count vectorisation on a particular document we first need to separate all the documents into individual words, this process of separation is called Tokenization. The count vectoriser provides a way of building a vocabulary of known words and using that vocabulary to encode new unseen documents (16). An encoded vector consists of the number of occurrences of all the words in the document and has a size of the entire vocabulary. While converting text to numbers, the count vectoriser converts all the uppercase letters to lowercase and hence those two words are considered as one. In case a document to be encoded consists of new words then these words are ignored as they do not form the part of the initial vocabulary and are not calculated.This is done to preserve the feature space on which the machine learning model will be trained.
- TF IDF: Counting words is a good starting point but the biggest limitation is that some words like "a" will appear quite often, increasing the value of counts but on the same hand is also not quite meaningful. We can surely provide the stopwords parameter in the count vectoriser so as to not consider these types of frequently occuring words but an alternative to calculating just the word frequencies is to penalise the words that occur more often(because these words do not add meaning to our feature space, they do not make it unique) by using TF IDF. Term Frequency- Inverse Document Frequency is a method that evaluates how relevant a word is in a collection of words by calculating the term frequency of a word in a document(which means the number of times that word has occurred in the document) and the inverse document frequency of that word(calculated by taking the total number of documents and dividing it by the number of documents containing the word and taking the logarithm of the word) across the set of documents and multiplying them as shown in the figure 2.4. And thus the words that appear more often have an inverse document frequency

closer to 0 while the ones appearing rarely have an inverse document frequency closer to 1. Using this method you are not only keeping a tab on what is appearing in the document but also on how important that word is in consideration of all the documents (16)(17).

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 2.4: Equation for TF IDF Vector

2.2 Machine Learning Model

There are a number of machine learning models out there that are able to get the task done for classifying fake news but in this domain we are also focussing on the explainability part. Machine learning models are a bunch of complex algorithms that crunch numbers and carry out millions if not billions of calculations to predict the final outcome and leave the task to us for accepting that output without an explanation which is also the reason why machine learning is frequently considered as black box- data goes in, solution comes out but the processes between input and output are opaque. In practice, most of the machine learning processes will be able to highlight the best possible features that contributed to those world class predictions and even attribute a score with it but nowhere does it say how that came to be. To break the for said culture of black box machine learning models we would be primarily using Decision Tree Classifier that will give us a detailed insight into all the decisions that it made to reach that classification in the form of a graph and in text format as well. Just to compare because decision trees have a reputation of overfitting and being non-smooth in their fitting process, we also use another Classifier, Ridge Classifier which was chosen because it was the best performing classifier in a certain experiment (discussed below) as a measure of comparison and with ridge classifiers limited explaining capabilities.

2.2.1 Decision Tree Classifier

A Decision tree is one of the most popular approaches for representing classifiers and is simply defined as a recursive partition of the instance space. A decision tree builds a flowchart like tree structure (as shown in figure 2.5) and consists of three parts: root node, branch node and the leaf node. A root node has no incoming edges and acts as the entry point into the decision tree. Every branch node has an incoming edge which it further divides into two outgoing edges by making a decision. Every leaf node acts as a final classification and all the data points that end up there belongs to a particular class. In general, it asks a question (like, is this variable < 0) in every decision branch and then classifies the data point based on the answer but it should be noted that

only one variable is considered at every decision point which is why the decision tree graph is not smooth in nature (see figure 2.6)(18).

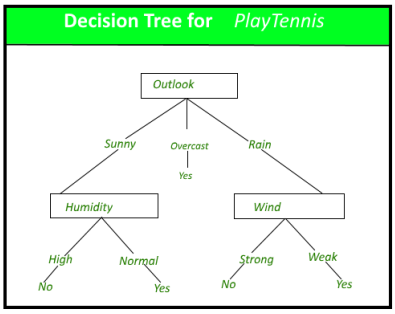

Figure 2.5: An illustration of a Decision Tree

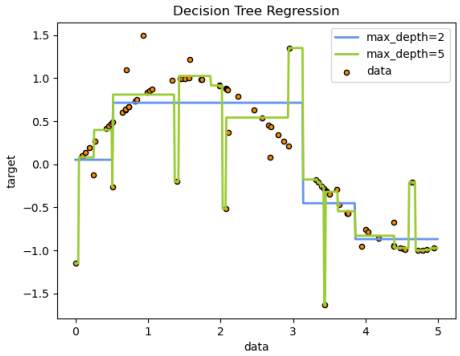

Figure 2.6: Illustration of a fitted Decision Tree

Let's now understand how a decision tree selects its features for making a binary classification for example. A decision tree tries to find a feature (from the available set of features) that divides the tree into two pure classes, with the left node consisting of all the data points from class 1 and right node comprising of class 2 but since a perfect split is not always possible it tries to find a feature that maximises the Information Gain (see figure 2.7). The formula for gini score (a metric to calculate the impurity at a node) is in the figure 2.8 and this process of dividing the set into further subsets based on attribute value is repeated on each derived subset until a decision is made that is able to divide the set into pure nodes or if the gini impurity of that node (without division) is going to be less than the gini impurity of that node after the division takes place with the best possible attribute of a feature in which case that node is also converted to a leaf node as shown by the right child node in the figure 2.9. In case a separation in the data set leads to an improvement in the gini score then the separation with the lowest impurity value is chosen.

2.3 State of the art

There is not a lot of research in the field of explaining the process of fake news detection however a good amount of research has been done in the field of fake news detection so this section would be

$$IG(D_p) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

where I could be **entropy**, **Gini index**, or **classification error**, D_p, D_{left}, and D_{right} are the dataset of the parent, left and right child node.

Figure 2.7: Equation for Information Gain

$$Gini = 1 - \sum_j p_j^2$$

Figure 2.8: Equation for Gini Impurity

divided into two parts: one dealing with the explainability off fake news and the other dealing with just the detection of fake news.

2.3.1 Explainable machine learning model

This research(19) in the field of Explainable machine learning in fake new domain focuses on how hard it is to detect fake news and what are the features of the fake news. It utilises the power of XGB models which is a gradient boosting technique that uses an ensemble of weak prediction models for classification and regression purposes making it a fast and effective method for building prediction models. On further analysis of various machine learning classifiers we learn that only 2.2% of the models achieve an detection performance above 85% in terms of area under curve thus establishing the fact that even the previous researches have not born a ripe fruit in this domain of detecting fake news however our research is more focussed on the explainability side of things. The research turns to fact checking websites, social media sites and established media houses for its collection of data. Further analysis revealed that only a handful of features were useful again and again in the classification of fake news. The selection of fake news features in the research (19) includes:

- 31 syntax level features ranging from calculating the number of words and syllables per sentence to identifying and calculating various parts of speech categories like nouns, verbs, adjectives etcetera.
- 59 lexical features ranging from first person pronouns and demonstrative pronouns to punctuations count etcetera.
- An analysis of the semantic structure gives you the perceived toxicity of various text and comments made with the help of a Google API.
- It also utilises the most popular dictionary based text mining software LIWC (Linguistic Enquiry and Word Count) which has been used by many researches for extracting 44 features ranging from proportions of words that fall into psycholinguistic categories(eg positive emotions, perceptual processes) to words appearing in individual categories(eg analytical thinking, emotional tone).

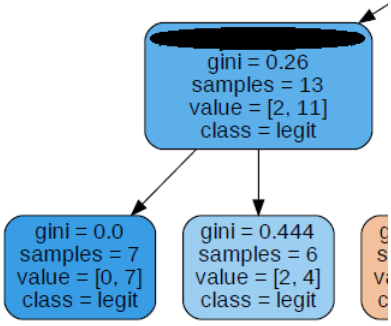

Figure 2.9: A branch of a Decision Tree

- The subjectivity of the text was also analysed and sentiment scores of a text were computed.

Apart from all the features extracted from the text, the research also focuses on meta features found from different media platforms like:

- The credibility and the trustworthiness of the news source.
- The location of the post from which it was posted with an understanding that people from certain cities are much more likely to spread misinformation.
- User engagement with the posts for example in facebook it looks at the number of likes, comments and shares of the post.
- The time interval between the several engagements of the posts in order to identify the virality of a certain post at a particular time.
- The political bias of the news articles as claimed by the fact checking websites.

After the establishment of all the models, it trains the gradient boosting model on all the possible combinations of features to achieve a 0.86 AUC to 0.88 AUC for the top 10% performing models and infers that there are only a small number of features for which the predictive accuracy is significantly higher. From the explainability point of view the paper takes the top performing models and compares the common features in them to infer that these features are the best performing features for detection of fake news, a feat almost every model can give by providing you with feature importances. A look into the most important features tells us that the number of shares and the reaction counts were more relevant by 40% and 29% respectively along with location and credibility, page reactions from users(not that particular posts'), IP of domains(a feature that is completely random) scored 29%,39% and 28% more relevant than others.

The explainability of the model basically revolves around the feature analysis of how a certain feature appearing in more top performing models is more likely to be a dominant feature and does not offer us any quantitative inference rather it offers only qualitative results like high page fan counts has a larger impact on the positive or negative prediction of fake news and how they are less likely to share fake news. Also, the research focuses more on the meta features of number of likes, comments and shares (which are inferred to be a larger contributor for fake news detection) rather than the analysing the characteristics of fake news like how polarising, toxic and subjective a piece

of article is. Despite all this, the research offers some great insights on all the features that are potentially useful in your classification of fake news. One approach we are likely to replicate in our experimentation is taking a huge number of features(that make some logical sense) and run over their combinations to see what works. Also, a decision tree is considered as a weak classifier with respect to the other models but it is much more explainable than any other machine learning algorithm out there so it is really a trade off between the misclassification rate and the degree of explainability it offers with a focus on maximising the later on a priority basis.

2.3.2 Deep learning model

Deep learning models or neural networks first came to light when they proved to be really useful in extracting out complex features from the text and have since been used in a number of other applications like Google's Search engine and amazon's voice assistant . Neural networks have been long known for their complex nature and solving the curse of dimensionality and prior to their appearance we used hand crafted features for text classification (20). On such research(21) that utilises deep neural networks in the domain of fake news focuses on classifying misinformation by performing 3 different techniques of converting textual information into numerical embeddings that are to be used as final features:

- By using the Bag of Words approach where it separately converts the headline and the content into their numerical representation using Count Vectoriser.
- By using TF IDF vectoriser and the cosine similarity to numerically represent the headline-article pair.
- By using a pre-trained word2Vec vectoriser that will represent the headline-article pair in a 300 dimensional vector space embeddings.

Unlike the previous explainable model, this one uses only the features extracted directly from the text for classification and attains an accuracy of 94% on the model using TF IDF embeddings as features. However good the accuracy may be, this deep neural network would not be able to explain the reasons behind its classification since vectors obtained after using TF IDF have been passed to the deep neural networks are represented in a hidden dimension only to be used by other layers in a seamless fashion to implement a complex non linear mapping from the input to the output as well as the top layer of the neural network is abstract(we cannot look at it) rendering the neural network unexplainable. However we can conclude that neural networks really champion the art of finding complex patterns in the most simplest of the features but we are more interested in the explainability as a model that performs exceptionally well in a particular dataset might not give the same results in all the other datasets and when that happens we would like to have the answers for why this model performed poorly and which again places the emphasis on explainability of a machine learning model.

3 Methodology

3.1 Focus

Before proceeding with the experimentation we must set the spectrum of our research as the current domain of fake news is too broad to be completely solved in the given timeframe. Fake news can exist in the form text or it could be an edited image with a tagline or a combination of the two. Our research will not focus on any pictures shared with the article, only the text and for the sake of our research question (point 2) we would also not be focusing on any social media engagement metrics like the number of likes, comments, shares, retweets or any other related feature that does not directly originate from the text.

3.2 Experimental framework

Unlike other researches that directly delve into the fake news dataset to extract out features we will first try to classify different categories of news and take a look at what are the features that separates legitimate news into different categories and try to use those features to draw out some patterns that will also possibly be able to separate the fake news dataset. If we are able to successfully do this then we would be effectively looking at a set of features that were not trained on a fake news dataset but are able to classify one and so in theory these features should also be able to categorise any other fake news dataset.

4 Experimentation

Each and every experiment carried out asks a question which forms our hypothesis for that experiment and then we re-iterate over that experiment until we reach a plausible outcome taking which we can draw some inference and that also serves as the starting step for our next experiment. Only those iterations of the experiments have been shown here that led to some concrete results or the ones that fitted in the developing process of classifying and explaining fake news.

4.1 Experiment 1

Hypothesis

Can we develop a machine learning classifier that can differentiate between the various categories (Business, Politics, Education, Health, Science and technology, Entertainment, Sports) in the Washington post dataset.

Aim

The basic aim of this experiment is to check whether a machine is even able to differentiate between different categories of text (Business, Sports etc) or not.

Methodology

There are two ways we can do this, first either we take the whole dataset and single out the labels and then divide all the labels in two categories of positive class and negative class for example we take the politics label and divide all its corresponding columns on its basis, so all the data that's not labelled as politics would be labelled as non politics and hence we have two categories politics and non-politics. The second method is to classify all the data as it is in their original labels that is business, sports, politics etc

Dataset

We take the sub data parsed from the Washington Post dataset which is a total of approximately 20000 rows. The data set contains three columns namely ,labels, normalised text, cleaned text. The different columns are explained below:

1. Clean text : This is basically every article that is extracted from the whole corpus of Washington post dataset in plain english. The preprocessing follows something like this, after

parsing the labels, the rest of the article associated with that data (headlines and accompanying text is combined) is stripped of all html/xml tags that came with the washington post dataset file along with the removal of any accented text characters that do not lie in the english alphabets. To go a step further all the contracted words like "don't" and "won't" were expanded to "do not" and "will not" respectively plus any of the uppercase headlines or words that might be there for enunciation were converted to lowercase letters. Lastly, the most important step of all, in order for it to make sense to the machine similar meaning words were reduced to a single agreed upon word that is the text was lemmatized (instead of stemming) which is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meaning to one word. For eg the word better would be lemmatized to good, the word rocks would be lemmatized to rock. This is done to keep the base meaning of the word unchanged which can turn out to be helpful if the machine encounters any other permutations of the word that mean the same. This feature is kept to give us a plain english alternative of the text

2. Normalize text : A normalized text consists of clean text but with more preprocessing involved like removal of all the special characters which clears out all the punctuations (periods, commas, semicolons, hyphens and colons) in a sentence. To further reduce the text, we remove the set of words called stop words from the text. In terms of Computer science, the stop words generally refer to the most common words (is, an, the, for etc) in any language whose removal might not/will not completely change the meaning of the sentence or the inherent knowledge of the text. The NLTK library consists of 100s of words that are pre-identified as stop words, it is an effective method to reduce the length of parameters in a sentence which in turn make the computations easier. Also, an abundance in stopwords in a text may lead to the finding of a false correlation between the final result and those particular set of stop words. Finally, being able to individually identify each word in a sentence into its separate entity type is a feat to achieve in itself. In simple terms, every word in a sentence can be roughly identified into nouns, pronouns, verbs, adjectives, prepositions etcetera but due to the huge number of words in the english vocabulary It is a difficult task to account for each and every word and hence we remove any word whose entity type we fail to identify. This removes any unfamiliar words whose occurrence might be so rare they may end up as an outlier in the combination of features. In a way, this process removes the possible outlier features to some extent.

Steps Involved (Experiment 1.1)

1. We load the dataset and separate it into dependent variables and independent variables.
2. The machine does not understand text so we convert the text in the labels (Discussions, Education, Entertainment, Europe, Health & Science, Sports, Technology, World, Travel) to individually assigned numbers starting from 0 and ending at the number of categories - 1 using the ordinal encoder.
3. Then we divide the variables into training set and test set with 10% percent of the data for

testing and the remaining one for training rather than going by the norm of 20% dataset for testing because of the lack of data available at hand for training also the option for shuffle is set to False which means that the data will exist in its natural form of indices rather than randomly placing rows, this could be an important factor in the training process affecting the overall accuracy as the most of the rows containing one particular class of data might naturally fall at the end of the dataset which, however will end up in the test set. As most of the data for one class will be in the test set, the model might not get aptly trained for identifying that class and thus ultimately affecting the accuracy. But currently we only have only two classes (so either the features will be for the class label or against it) which is a more favourable circumstance.

4. The next step should be to convert these words into numbers (for it to be interpreted by the machine and make the computations), we use TF-IDF for that purpose. How this works is we first put the whole array of "normalized text" sentences into the object of TFIDF Vectorizer. Here each index of the array consists of a sentence which is first separated into its individual words thus forming a corpus of all the words that were entered into the TF-IDF object. The term frequency of every word is calculated which means the number of times that word has occurred in the document along with its inverse document frequency which indicates how commonly a word is used. Inverse document frequency helps us in determining the importance of a word as words that naturally occurs more frequently in a document would not have a significant impact on differentiating between different classes and would generally be assigned a lower score.

5. We have primarily used two methods of the TF-IDF object, the fit transform method which is used for calculating the TF-IDF scores of every word in the training set and the transform method which is used for projecting the scores of previously calculated words that were in the training dataset. The transform method is used only on the test set.

6. We use two different objects of TF-IDF Vectorizer for fitting normalized text feature and clean text feature so that their encoding data remains preserved.

7. Then we concatenate all the features (reading score, normalized text, clean text) and convert them into pandas DataFrame data type.

8. For machine learning model we use the LGBM or Light Gradient Boosting Machine framework which was developed by Microsoft and is a tree based learning algorithm. Gradient boosting is a technique that produces a prediction model based on the ensemble of weak prediction models, typically decision trees. It works like the principle of Wisdom of the crowd, here several decision trees (each of which is able to make one prediction based upon a single feature with a certain accuracy) are combined in levels and works on the intuition that the next possible model which is able to minimizes the overall prediction error would be chosen. Thus by a combination of these weak models we get a stronger working model.

9. Now over to some of the parameters that were used in this model prediction. The parameter objective determines whether the problem is a regression or a classification problem, we have selected binary here as it is a two-class classification problem as of now. The parameter max

depth is used to control the growth of tree levels and we have set it to 30. The parameter learning rate determines how fast the model will converge to its optimal minima and is set to 0.1. The parameter metric (loss function) is used for the determining the error between the predicted and original value according to which the model converges further, we have chosen binary logloss for quantifying the performance of our model as this loss function not just depends on the predictions made but also considers their prediction probabilities. The parameter feature fraction randomly selects the set percentage of features on each iteration, this was used as a means to prevent overfitting and also as a means to select and eliminate features that are there but of no use as initially there is no way of finding out what features could be extracted out. It is currently set at 80%. The parameter boosting is used on default value where it uses gradient boosted decision trees. The parameter bagging fraction is used to subdue the amount of data resampled at every iteration and is used to decrease the training time of the model while preventing overfitting, it is set to 80%. The parameter bagging freq determines at what intervals the bagging will take place and is set to every 30 iterations. The parameter verbose determines the number of lines of output that will come with every epoch of the training process and we set it to 0.

10. Next we will talk about the parameters used inside the lightgbm.train method. Apart from the "parameters" variable(discussed above) being passed as argument we pass the training and test sets as a list in the parameter valid sets. The parameter num boost rounds determines the number of iterations that will take place over the training process. The parameter early stopping rounds will determine if the model isn't converging any further in the said early stop rounds.
11. After the training process is complete we finally make our predictions using the predict method. And calculate some metrics for our observation.

Result

Currently we calculate the accuracy and find out the confusion matrix of the testing and prediction data sets. The accuracy comes out to be 95% in this case of binary class classification

Observations

While looking at the confusion matrix we safely conclude that this is an example of a skewed dataset. In the figure 4.1, Here C0,0 is True Negative representing the number of rows consisted in the test dataset correctly classified as "Not politics". C0,1 is False Positive representing the number of rows in the test dataset that are incorrectly classified as "Politics". C1,0 is False Negative representing the number of rows in the test dataset incorrectly classified as "Not Politics". C1,1 is True Positive representing the number of rows in the test dataset correctly classified as "Politics".

We can deduce that this dataset is skewed as the total number of actual rows labeled as "Not Politics" is $91 + 1 = 92$ while the total number of actual rows labelled as "Politics" are $4 + 4 = 8$. So, if a model incorrectly classifies all the rows as "Not politics" then in C0,0 we would have 92 and

confMatrix - NumPy object array

	0	1
0	91	1
1	4	4

Figure 4.1: Confusion matrix

8 in C1,0. and thus if we calculate the accuracy would still come out to be 92% without the need of any training as shown in figure 4.2.

cm2 - NumPy object array

	0	1
0	92	0
1	8	0

Figure 4.2: Confusion matrix

The changes in the next experiment would be to not reduce the multiclass labels to two classes and let it be set at the original set of 85 distinguished classes.

Steps Involved (Experiment 1.2)

With this experiment, we are effectively testing out our second method from our methodology in this section

1. We repeat the steps 1-8 of the experiment 1.1.
2. Because it is a multi-class classification we need to change some of the parameters in LightGBM like the parameter objective is set to multiclass for multiclass classification rather than binary classification in the previous experiment. The parameter num class is set to the number of distinct classes in the current dataset that is 85. The parameter metric is also changed to multi logloss for correctly converging the model to its minimum.
3. And finally train the model based upon the change in parameters described in the previous step.

Result

We calculate the accuracy and find out the confusion matrix of the testing and prediction data sets. The accuracy comes out to be 16% in this case of multi-class classification and the multi log loss comes out to be 2.89.

Observations and Changes

This time the accuracy drops from a whopping 95% to a mere 16%, conforming our argument that nothing fruitful was being derived in the previous experiment for differentiating between the classes.

This can also point out towards the inability of a particular machine learning model for not deriving any valuable features so we isolate the current features while changing the machine learning model.

By isolating the metrics of the experiment and changing one metric at a time will help us reach closer to figuring out the optimal solution. After establishing that the features are the problem we will move towards making changes in the features.

For the next experiment, we choose a Decision Tree Classifier which is a less complex version of the LightGBM

Steps Involved (Experiment 1.3)

1. We repeat the steps 1-7 of the experiment 1.1 for our multiclass classification.
2. For this experiment, we are using Decision Tree Classifier for classifying multiple classes. A decision tree Classifier is one of the most powerful tools for classification problems. It is a flowchart like tree structure where each branch represents an outcome of the test, each internal node represents a test on an attribute and each leaf node represents a class label. A Decision tree is able to fit to the dataset using a process called recursive partitioning in which the source dataset is divided into subsets based on an attribute value test. As the name suggests, this process of recursive partitioning is done repeatedly until all the nodes in a splitted subset have the same class label or when splitting no longer adds value to the predictions. In general, a decision tree classifier has good accuracy and is able to handle high dimensional data. Also, one does not need to have the domain knowledge for using a decision tree and is therefore appropriate for exploratory knowledge discovery.
3. There are a lot of parameters in the Decision tree space but for the initial experiment we have only set the parameter criterion which essentially helps you in determining a good split point for the root and various other decision nodes that will be there in the decision tree model. The split point on a feature is decided by the maximum information gain for the selected criterion (gini or entropy).
4. Since there are a lot of parameters to play with in the decision trees, we take the help of Grid Search CV which is the go to tool for finding the optimum hyper parameters in a machine learning process. As we know that the hyper parameters are not learned by the machine learning model because of which we need to keep on experimenting with different values of it. While using grid search CV you could input all of the values of that parameter you want to test into an array format and the algorithm will test all the possible combinations of inputted parameters for you while returning the best matched parameters for your task. In addition to searching the parameters, a grid search CV employs a sample testing technique known as K-fold cross validation where the whole dataset is divided into k-subsets and the estimator is trained on k-1 subsets while testing is done on the remaining subset and this process is repeated k times so that each and every data point gets to be in the test set exactly once which gives us an average on how our estimator will perform all while excluding the chance of deliberately training on a data set that would yield better results. The only disadvantage of

such a process is that it is computation heavy. In order to run a Grid search of all the parameters you need to provide it with an estimator (Decision tree here), a parameter space, a method for searching, a K-fold cross validation value and a scoring function which will determine the best parameters.

5. As the decision trees suffer from an overfitting problem by making over-complex trees that do not generalise the data as well. The parameters entered for the Grid Search CV are necessary for preventing this kind of problem.

    - Max depth: It determines the maximum number of nodes that would be allowed to make a decision tree for the given dataset. One reason for limiting the depth of a decision tree is not to make it a complex decision tree and helps overcome overfitting.
    - Min samples leaf: It determines the number of samples required on one of the sides to render it as a valid split(decision). Having a minimum sample prevents decision trees from splitting when there is only one sample at a particular decision in a tree. This helps the tree to generalize well on the overall data.
    - Min samlples split: It determines the minimum number of nodes required to split them into left and right sides even if a split point exists. This feature inadvertently counters the maximum depth parameter when a tree split at the node branch doesn't lead to comparable samples on both sides and reduces the depth of one side that had lower samples to begin with despite the specified maximum depth parameter.
    - Max features: It determines the number of features to consider from the dataset when looking for the best split. It is important to note that the features considered are not restricted to this parameter while looking for the best split.
    - Min weight fraction leaf: It is the fraction of the total sum of weights required to be at a leaf node. The weights of the individual samples are determined by sample weights, here sample weights are the same/equivalent for every class unless otherwise specified. This parameter is a way to deal with class imbalance problems that may arise at a leaf node. Class balancing may be done by normalizing the weight values for each class. Also note that this parameter is less biased towards the dominant class situation that will arise at the leaf node than other parameters like min samples split which do not consider class frequencies while considering features for the best possible split.

6. Then we train the model on the default values of the Decision tree classifier and on the various values of the said parameters using the Grid search cross validation method to find the best hyperparameters for our Decision tree classifier. The best fit found for the parameters is shown in the figure 4.3 below:

Result

We use the same metric of accuracy to check how our model performs. The accuracy from our default Decision tree Classifier comes out to be 22% and the accuracy from our Decision Tree classifier using Cross validation comes out a little better at 26%.

Key	Type	Size	
criterion	str	1	gini
max_depth	int	1	100
max_features	str	1	sqrt
min_samples_leaf	int	1	2
min_samples_split	int	1	6
min_weight_fraction_leaf	int	1	0
random_state	int	1	42

Figure 4.3: Best hyper parameters of Decision tree

Observations

We know that both of our metrics aren't good enough to classify anything in the real world scenario but there are some observations that are to be drawn from it

Since the accuracy of both of our decision trees (22% and 26%) comes out to be better than the experiment 1.2 performed with LightGBM classifier (12%), we may safely conclude that for starters a simpler Decision Tree classifier works better than a complex ensemble of Decision Trees in a gradient boosting model so maybe we will use the gradient boosting in the future to only improve accuracy but there would be a tradeoff between the explainability and accuracy.

Cross validation is able to reduce the overfitting of our default decision tree to some extent and hence using one is beneficial for us in our experimentation.

During the experimentation we also find that there are fewer no of classes that make it to the test dataset and even more eccentric is the fact that some of the labels in the dataset of 20,000 rows only occur once in the entire dataset (as pointed out by the decision tree in the form of a warning).

On a closer look we find that much of the labels like NBA, MLB, NFL could be clubbed together into a single umbrella category known as sports in order to reduce the complexity.

It was found that most of the labels could simply be deleted like the ones that only have a single occurrence, the ones under the category None and the ones that do not deal with facts like the articles under the category of discussions and opinions.

One possible reason why the decision tree is failing could be because we aimlessly put all the features that is the complete article (normalizeText) as a single feature. In the hope of getting better results we will put more specialized features like nouns, proper nouns and Noun Phrases.

Inference

An important thing that hasn't been pointed out in the Steps Involved is that we are not using the feature "clean text" in any of our iterations of Experiment 1. The reason being that "clean text" when expanded out TFIDF vectoriser gives us a total of 100,127 features, effectively putting the system out of memory.

Even when we are using just the "normalize text" as a feature, the feature counts about 36,979 after vectorisation (but the system does not go out of memory). Also for the same reason drawing out the explainability graph for this decision tree is unintelligible.

4.2 Experiment 2

Hypothesis

Extracting out specific features (like nouns) from the dataset will be beneficial

Aim

A reduction in the number of features would lead to improved accuracy and better explainability. An emphasis would also be placed on further cleaning of the data as pointed out by the decision tree.

Methodology

We would start by cleaning the data and then train on the cleaned data for three of our features namely Nouns, Proper nouns and noun phrases. Keeping in mind the accuracy provided by the decision tree and the fact that there still could be a high number of features produced after vectorisation we would use one more model for training that could also be explainable in nature

Dataset

Initially the data is cleaned of useless labels that made their way into the dataset probably because not all the articles in the Washington post were of the same format and hence the parsing program couldn't correctly parse them. As there is no way of telling what label these data rows are associated with, it would be best to identify these data rows that all the rows having less than 10 occurrences in their "labels" column are to be completely removed from the dataset. As a result, the total number of distinct labels dropped from 85 to 51. Talking about the features in our cleaned dataset, we have:

1. Nouns: A noun is any word used to identify any of a class of people, places, or things. We choose this feature because nouns are a specific set of words that are generally used in a particular context and some meaning can be associated with them rather than using verbs, pronouns or adjectives which can independently exist anywhere in text.
2. Proper Nouns: A proper noun is a specific name of any person, place or thing. This feature is a specialised case of nouns and helps us identify the various actors/places or regions around which our article might be based. For eg the term "president" is most likely used in Politics and hence the article may be categorized as politics.
3. Noun Phrases: This feature is a specialised case of two proper nouns existing together adjacently. Usually the vectoriser separates all the terms individually due to which when two proper nouns occur adjacently we join them into one string using an underscore(president_

obama). This is helpful in deriving even more meaning out of the words. For eg the terms "new" and "year" when existing separately in the feature space have an individual meaning but when they are combined to form "new_year" marks an occasion.

Steps Involved

1. We load the dataset and seperate the dependent variable(labels) from the independent variables.
2. The independent variables(noun, proper noun, noun phrases) are concatenated into one single feature in order for it to be vectorised by a single TFIDF vectoriser object.
3. The dependent variables are encoded using the Ordinal Encoder which assigns a seperate number to every label present in the dataset.
4. The next step is to divide the complete dataset into training set and test set with 80-20 split meaning 80% of the data is in training and 20% of data is in test set.
5. We then pass the training set and test set for vectorization. The TFIDF vectorizer object fits itself onto the training set while using the parameters max_df = 0.5 which ignores the words that have a higher document frequency or simply the words that occur more frequently than the set threshold. Another parameter we use sublinear_tf which is set to true and goes beyond calculating the term frequency in tfidf, in fact it replaces the concept of term frequency with weighted frequency as in figure 4.4.

$$\mathrm{wf}_{t,d} = \begin{cases} 1 + \log \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Figure 4.4: Weighted frequency

It has two important impacts, first it replaces the otherwise less useful term frequency with weighted frequency and it also acts as a normalization tool as words can occur a lot of times in any document. Using the same object we then transform our testing set.

6. We also extract out our feature names that come out after the vectorisation process. These feature names are nothing but essentially the individual words originally present in the articles.
7. As specified in the methodology section, we plan on using two classifiers one being the decision tree and the other is Ridge Classifier. The selection of Ridge Classifier wasn't random, a set of classifiers (Ridge Classifier, Logistic Regression, K-Neighbors Classifier, Perceptron, Decision Trees) were taken in which the Ridge Classifier turns out to be the best at classification.
8. Apart from being the best classifier, Ridge classifier is also the fastest at training because of its parameter solver = sag that is the Stochastic Average Gradient Descent. This solver is particularly faster when it has larger features and a larger sample size, which is exactly the case here.

9. We then proceed to fit the ridge classifier on the training dataset and make its predictions on the test dataset.
10. The accuracy metric is calculated on the predictions made and the original labels.
11. The better part of the ridge classifier is that we can extract the top n key features from every label which most prominently classifies that particular label. We do this by extracting all the coefficients listed corresponding to their features in the feature space of the classifier using clf.coef_ that returns an (n_classes, n_features) and then we loop over all the classes to target a particular class' feature space. The coefficients are of less importance here, it's more important to extract their indices and then sort the indices according to their corresponding coefficients(weights). This gives us the indices of all the features sorted according to their coefficients in ascending order meaning that indices in the front would be of less importance than the ones at the back.
12. A temporary features list is made that consists of all the feature names of that class whose coefficients were just extracted. This temporary feature list is extracted using the indices of the coefficients, these indices are used to extract the corresponding feature names that were originally extracted by the TFIDF vectoriser but this is already sorted in an ascending order of feature space.
13. A variable is set to represent top n features that you want to extract here we set it to 10 keywords at a time. This variable is then used to extract 10 features from the last of that temporary feature list. Like the ones listed in the figure 4.5 below

```
Politics
: robert_barnes postpolitics justices congress gop santorum campaign davidson romney joe_davidson
```

Figure 4.5: Politics class Keywords

14. We then proceed to train the Decision Tree that follows the same processes of training and testing and then predicting the accuracy metric.
15. After training the decision tree we also produce the explainable graph although it's not very interpretable as the total number of features are about 338805 which makes it too big of a graph to be humanly understandable.

Results

The accuracy of Ridge classifier comes out to be 71% and the accuracy of our Decision Tree comes around 50%.

Inference

We see that the accuracy for the decision trees increase significantly which indicates to the fact that the new features of Nouns, proper nouns and noun phrases extract much better information than directly inputting the whole text which is of lesser value. We see that another classifier is able to perform a better job at classifying different categories of the articles than the decision tree itself

which confirms our previous suspicions of decision trees being the bottle neck on the performance metrics.

Now we will try to answer the question as to why our decision tree is performing as such

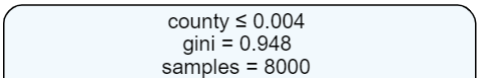

Figure 4.6: Root node

In the above figure 4.6, as we cannot show the whole graph, we are able to show the branch node that initially is the whole tree. The feature is the word "county" and in order to make a decision for a data row initially we look at the TF IDF score of the word "county" in that sentence. In case this word exists in the data row and has a score of greater than 0.004 the decision tree looks towards the right side for further decisions and in case the data row does not have a word "county" in it (which would directly render the TFIDF value =0) or if it does and the TF IDF score is lesser than 0.004, the case holds true and the decision tree looks towards the left side for further analysis. The above feature segregation does not work very well because of the high value of gini impurity which tells us how impure is the remaining set after a decision and the first decision in any decision tree accounts for the maximum information gain over any of the other available features and any other possible value for the separation, meaning that the best possible split in the course of 338805 features this was the best split and offers a high impurity as well. In the below figure 4.7, we see some leaf nodes among several other leaf nodes that have a decision being made on a single row in order to classify it between any of the labels (different colors represent different colors). The decision tree is making a decision for one particular sample to exclusively fit it in a particular class and is hence overfitting. The above mentioned points could be the possible reasons for the decision trees to perform poorly.

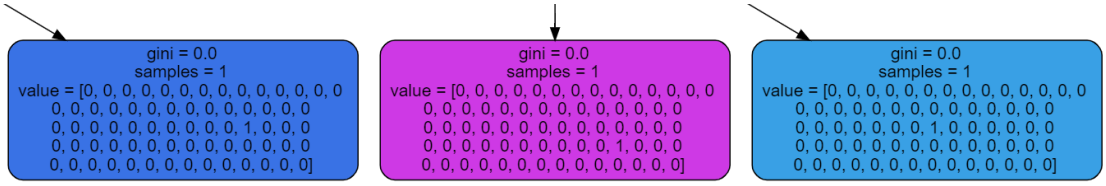

Figure 4.7: A glimpse of the leaf nodes

We carry on with our current results for our next experiment since we have a classifier (Ridge) that has good enough accuracy metric and even consists of extracted features that will take into account the things on the explainability side.

4.3 Experiment 3

Hypothesis

A well trained classifier should be able to give a good accuracy on unseen data.

Aim

To check how our classifier performs in recognising the various categories of news that is published around the world

Methodology

Since we were using a Washington post dataset it would not be wrong to assume that a lot of the news covered in it would be regional and hence the knowledge gained from that resource may or may not be useful. In other words it might not produce the same result when looking at an article that comes from a news source situated in another part of the world. To test this we take 4 reputed news sources (Al Jazeera, BBC, CNN and Times of India) from varying parts of the world (Middle east, South-east Asia, British, American), picking up 3 articles each in 3 popular domains of health, sports and tech and checking how our classifier performs. We extract the same 3 features(Nouns, Proper Nouns and Noun Phrases) from these articles and test it against our pre-trained classifier. Apart from making predictions, our focus would also be on telling why the classifier has made that prediction whether it be correct or an incorrect one.

Dataset

The dataset has been parsed carefully and manually through the various news websites. The Data purposefully features conflicting keywords or words that may fool the classifier to predict the article in another class than the one it originally belonged to. We take a look at some of the conflicting articles like this one from BBC health section which describes how Trump halted the US funding for the WHO when it praised China's efforts for stopping the early spread of the virus even when they silenced the medics in their country for showing early concerns regarding the virus; which however could very easily be classified as a political fight of power between two nations. Or the article from BBC tech section talking about health and fitness in relation to their fitness tracking app and how the data from their app could be misused by health insurance companies when their clients' claim health insurance denouncing them for not completing their targets; could easily be mistaken as an article related to health. Or it might be this article from CNN health section where Dalai lama talks about how to find inner peace, ending violence and inequality and the ill effects of using a smartphone and the ways technology affects us in the modern day leading us to experience a lack of focus, inadequate sleep etc; just might get classified as a tech article. Or what happens when Tiger Woods receives the Presidential Medal of Honor from Trump or the news of drones helping in the fight against coronavirus by air dropping medical supplies. It would be interesting to see these classifications as it would give us a better understanding about how the classifier performs. After collecting all the textual data of 37 articles in these well classified fields, we parse the data from its original text format and then extract the three features (Nouns, Proper Nouns and Noun Phrases) from it.

Steps involved

1. We start by loading the estimator and the vectoriser object (with the help of pickle library) of our previously high performing Ridge classifier that was able to classify 51 labels with a 71

2. We then load the dataset into a variable and then proceed to separate out the three different features with the help of the Parts of Speech tagging feature as previously discussed.
3. After extracting the features, we need to concatenate the features into a single row to make it easier for the TFIDF vectoriser object to carry out its operations and also because we don't want to use separate objects for vectorisation.
4. As the concatenated features have a "next line" symbol in it every row ends up as a 2D matrix which we take care of by putting the spaces in places of the "newline" characters for each row in the dataset.
5. Now we must separate the dependent variables from the independent ones.
6. Since this is essentially a testing experiment there is no training process involved so we start by transforming the independent variables (features) into their vector forms while using the vectoriser object used in the training the multi class classifier in the previous experiment. This is done to effectively keep the feature space of the of the testing dataset equal to the one the estimator was trained upon because in case we introduce a new vectoriser object, it's very likely that the vectoriser will have lesser or more number of distinct words than found in the training space and since each extra word actually ends up as a new feature which will change the dimension of the feature space leading us to an error in the prediction process.
7. After vectorisation, we use the predict method of the classifier for predicting the labels in the testing dataset.
8. We then manually compare the predicted labels from the actual ones because the Washington post dataset had separate labels for major sports leagues like NFL, NHL, MLB etc which could also be directly classified under Sports.
9. Just because we had a decision tree classifier readily available at hand, we load that as well and make the predictions to check whether its performance gets better or worse on the unseen data.

Results

In comparison we found that 30 out of 37 articles were correctly classified for the Ridge classifier which is about 81% accuracy but we shouldn't be using that metric here as the dataset is too less to quote an actual test accuracy. Nonetheless, our decision tree classifier performed comparatively similar to its previous experimental results with 18 out of 37 articles being correctly classified which is roughly half of the predictions being correct. Note : in this manual calculation of corrected predictions we have considered articles classified as "NHL, NFL, Allmet Sports" under the Sports category only.

Observations

We see that both the classifiers perform comparable to their performances in the test dataset as they did in the validation dataset irrespective of cleverly choosing the articles to be interpreted as

more than one class. This in turn solidifies our belief system that the classifier is very well suited when performing in a real world test case scenario.

Inference

With the focus of our project being the Explainability we will now look at some of the positive and negative results obtained from the classifier(Ridge classifier) and will try to explain why the classifier has predicted so.

Our first misclassification comes from the technology section in Times of India, the article talks about the new digital tax laws that are soon to be introduced in India and will be levied on all the foreign e-commerce companies in the near future. This article is primarily classified in the technology section by the Times of India but our classifier has misclassified this article to be of "Asia & Pacific" section primarily because when looking for the top keywords in the "Asia & Pacific" label of our classifier, the keyword "India" comes at fifth priority and has 10 keyword search matches (as shown in figure 4.8)in the entire article which overshadows the 2 keyword search matches of the word "google" which is at the third priority (as shown in figure 4.8)for classification of an article into the technology category. With the misclassification of a Technology article, let's move on to the

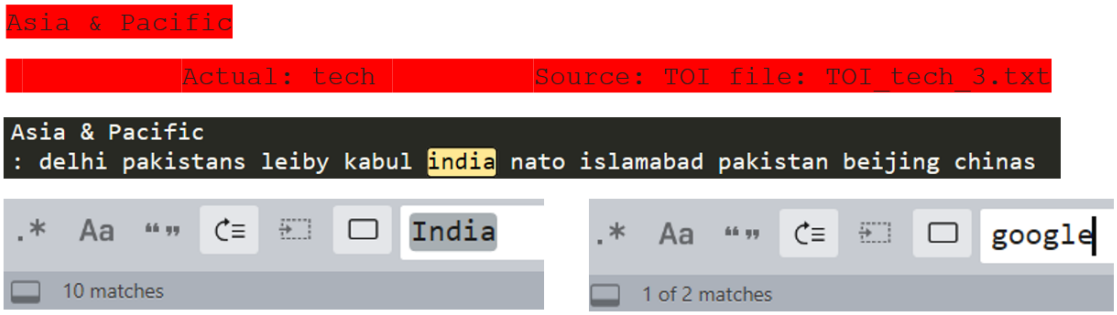

Figure 4.8: An example of misclassification

correct classification of the same. In this article by CNN in the technology section they talk about these smart shopping carts that will let you skip the grocery store line by automatically charging you as you move out. This article is correctly classified by our classifier in the technology section because when you look at the top 3 keywords (in the figure 4.9) listed as "company", "apple" and "google", all of them have a positive keyword search match in the article with the keyword "google" matching 1 time and the keywords "apple" and "company" both matching 2 times.

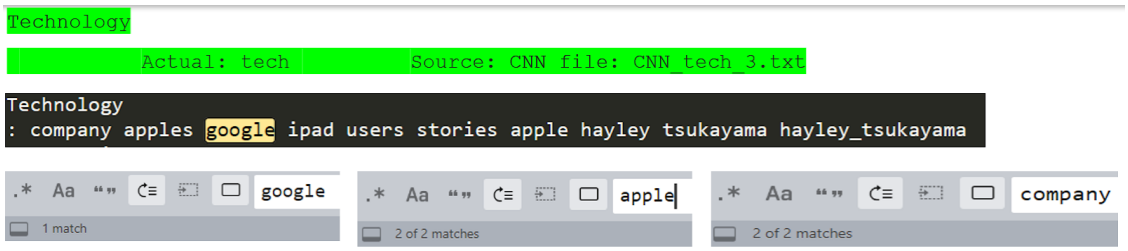

Figure 4.9: An example of correct classification

As for the examples given during dataset section of the this experiment:

1. The one where Trump cut the US funding for WHO was correctly classified in the health section with the keywords "virus" occurring 5 times, "disease" occurring 3 times, "study" and "research" occurring 1 time and also the word "health" appearing 7 times(as it should in an article about health).
2. The one which talks about health and fitness in relation to their fitness tracking app was also correctly classified in the technology section with the supporting keywords "devices" occurring 3 times, "apple" appearing 2 times and the words "company" and "privacy" each appearing 1 time. (Yes, We had to look at the top 50 keywords to explain this one meaning that the article had more chance of getting misclassified).
3. However, The article where Dalai lama talks about the need to reduce the usage of technology got misclassified to Local category while it was from Health section because our classifier found the keywords like the abbreviation "dr" appearing twice and the keyword " mccartney"(which is the second priority keyword of the local label) despite having health section keywords like "study" and "Scientists" both appearing once.
4. The article where Tiger woods receives the Presidential medal of honour is correctly classified as sports with the keywords "golf" appearing 8 times and "champion" appearing once.

We haven't classified Fake news yet but all the experiments conducted above setup the ground works for essentially carrying out our knowledge transfer in the fake news experiments.

4.4 Experiment 4

Hypothesis

Can you transfer knowledge from the real world data to classify fake news?

Aim

Classify the individual rows of fake news dataset into previously predicted categories and use those predictions as features to train a classifier on the fake news dataset.

Methodology

This experiment will unfold in two steps. The first step includes the preparation of special individual classifiers in their individual categories that is a specific classifier for Politics, Business, Health & Science, Sports, Education, Entertainment and Technology. Because there are these 7 categories only we would also need to reduce the dataset that includes only these 7 categories and no other label. We have chosen these categories because they had the largest number of occurrences in the whole Washington post dataset.

The second step is to make the predictions for every row in the fake news dataset with respect to every special classifier and use its prediction probabilities as a feature to train a new classifier (stacked upon the results of previous classifier) on the fake news dataset. This method will essentially allow us to transfer the knowledge from the washington post dataset to this Snopes fake

news dataset (22). We are expecting some bizarre results on the prediction probabilities on the rows labelled as fake news, something like all the prediction probabilities coming out to be equal or getting misclassified or having a very high probability for each of the classes. What we need to understand here is that these special classifiers cannot predict fake news for real one simply because they were not trained on one but they can however be used for transferring knowledge between what we know is completely legitimate news and the other dataset that consists of fake news which is classified by a fact checking website (Snopes).

Here, one might argue that why was the single TFIDF vectoriser not used for the training of the stacked classifier as it could also be used as a resource for transferring knowledge and their reasoning would be right since the special classifiers have been trained on the same keywords that are stacked classifier would be trained upon but since one technique isn't differentiable than other we are using special classifiers and also it gives us more features but leaving no stone unturned we irrespectively went ahead and trained a classifier on the vectoriser output as well.

Most of the research before this time has focused only on extracting out only meta information from text articles like number of words, noun count, preposition count, type token ratio etc. This Experiment takes a different approach from those conventional approaches and checks whether this approach of transferring knowledge is possible or not.

Dataset

While first looking at the Washington post dataset in cohesion with our approach, one thing is clear that we need to reduce the number of labels from 51 to 7 as we would only be training special classifiers on Politics, Business, Health & Science, Sports, Education, Entertainment and Technology labels only because these are the most abundant in our current dataset as shown in figure 4.10. The Labels in the original dataset are transformed into BroadLabels which comprises

```
BroadLabels
Sports              2252
Politics            1982
Business            1954
Technology           583
Health & Science     520
Education            213
Entertainment         72
Name: BroadLabels, dtype: int64
Index(['Business', 'Education', 'Entertainment', 'Health & Science',
       'Politics', 'Sports', 'Technology'],
      dtype='object', name='BroadLabels')
```

Figure 4.10: Classes with high frequencies

these 7 categories only and we essentially remove all the other labels. The next step is to keep all the possible labels and convert all the remaining labels into negative labels for the training of every classifier in each category for eg when we are training the Ridge classifier (because it gave a better accuracy metric in the previous experiment) in Politics category, we keep all the label values equalling to "Politics" and convert all the remaining values to "other". The conversion of the labels is done while training the individual labels' classifiers. To give an overview, the dataset contains columns that go by the name 'Noun Phrases', 'Nouns', 'Proper Nouns' and 'Labels'. For executing

the second step of the experiment we require a fake news dataset and here we have two at our disposal, both having their own significance. The first dataset for fake news is collected by Snopes Website which is a fact checking website that debunks myths, hoaxes, urban legends and rumours appearing on the internet since 1994. Since this is a fact checking website, every news article that is labelled as "Fake News" in this dataset has actually been spread around the internet and possibly had some people believing in it which is where this dataset is significant. The original dataset involves fields like "url", "article_title", "article_text", "article_category", "article_claim" among 20 other fields before we dissect it into our own features of 'Noun Phrases', 'Nouns', 'Proper Nouns' and 'Labels'. One interesting field in the original dataset is "article_claim"(which would have been a very good feature in cautioning people from whatever agenda that lied inside the twisted words of the article by directly displaying the claims made inside), on some evaluation it was found that the values in this column did not directly match the text "article_text" hence confirming our suspicion that the values in this column were written by a human not a machine.

The second dataset for fake news was itself collected in two parts: first by crowd sourcing and second by collecting rumours spread around in the entertainment industry. We would just be focussing on the data collected by crowdsourcing. It was started by collecting a dataset of legitimate news belonging to 6 different domains (Sports, Business, Entertainment, Technology, Politics and Business) unlike the fact checking websites that usually kept their focus on one domain (Politics) and then by successfully crowd sourcing these news articles to make their fake versions of it through amazon mechanical turk. Even though that now we have a much balanced dataset with both the classes (fake and legit) being equal in numbers from 6 different domains there are still some caveats in this dataset that should be pointed out like normal people might not have the same writing style as of journalists which may unknowingly impact our classifier performing poorly in the general world.

Steps Involved (Experiment 4.1)

1. We start by loading the Washington post dataset into pandas dataframe object. The dataset already consists of all the required features as well as 51 labels that are to be reduced.
2. We make a separate list for capturing the broad labels, assembling the label values of "health" and "science" into a single label of " Health & Science" plus the label values of "NBA", "NHL", "MLB", "NFL", "Soccer" into a single label of "Sports".
3. Another list(useful labels) specifies all the names of the labels on which special individual classifiers will be trained based on which we filter out all the labels that are not on the list and their rows are removed from the dataset. We name this newly formed dataset as "categorised dataset".
4. We loop over the useful labels for training individual categories and convert the labels into positive classes and negative classes. For example, when we are training the Ridge classifier in the Politics category, we keep all the label values equal to "Politics" and convert all the remaining values to "other".
5. Then we divide the labels into independent and dependent variables as well as further

splitting them into training set and test set with an 80-20 split with the shuffle option being true. With all the independent variables at one side we start to vectorise all the combined key features using the TF IDF vectoriser and ordinally encode the dependent variable(labels).

6. The conversion into two classes also gives us an imbalanced dataset meaning comparatively more number of occurrences for a particular class (like 1982 of Politics vs 5594 of Others). This imbalance in the dataset will cause the classifier to give skewed predictions like in experiment 1. To mitigate this problem, during the training process we set the parameters for the RidgeClassifierCV (cross validated Ridge classifier because cross validation is also one of the ways to mitigate imbalance in the dataset). The first parameter being class_weight = balanced which uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as

$$Nsamples/(Nclasses * np.bincount(y)) \tag{1}$$

7. The second parameter being cv = 10 for cross validating the entire dataset 10 times to reduce any training error left due to the imbalance of the dataset.
8. After the training process, we predict the accuracy, precision and recall of the positive labels (labelled as 0) along with their confusion matrix to check whether the individual classifiers perform better or not.
9. Finally, we save all the 7 Individual classifiers and the TFIDF vectoriser to be used in further experimentation using pickle library.

Observations

In table 4.1, we see that all the classifiers perform a great job at classifying their positive labels but since all of them are binary classification and that on an imbalanced dataset it is important for us to look at more than just the accuracy metric. To actually know whether the classifier is able to learn the difference between positive and negative labels we need to look at the recall values which is the percentage of positive classes correctly classified and the precision values which tells us the probability that a positive class is correctly classified. While the precision and recall values for all the labels are satisfactory indicating that the classifiers have learned the differences between the labels we see the recall of the Entertainment Classifier quite low at 0.25 however its corresponding precision is at 1 meaning all of its predicted results are correct but it still leaves out a lot of positive labels, this could be because it has only 72 positive classes vs 7505 negative category but nevertheless it makes this classifier not very desirable.

Irrespectively, we move towards our next iteration for classifying Snopes dataset with our Individual Ridge classifiers.

Step Involved (Experiment 4.2)

1. We start by loading the Snopes Dataset consisting of the three key features and the labels having a total of 313 rows. We should note that because it is a different dataset the labels

Table 4.1: Performance metrics of Individual Classifiers

Classifier	Accuracy	Precision (Positive Class)	Recall (Positive Class)
Sports	0.98	0.99	0.96
Business	0.94	0.93	0.84
Technology	0.97	0.86	0.81
Health Science	0.97	0.89	0.73
Politics	0.95	0.90	0.91
Education	0.98	0.93	0.68
Entertainment	0.99	1	0.25

will be different.

2. So we start by extracting out the required labels('Politics', 'Health Science', 'Fake', 'Sports' , 'Entertainment', 'Business', 'Technology') from the dataset by converting label 'Politicians' into 'Politics', combining labels 'Science' and 'Medical' into "Health & Science' and removing the rows containing the remaining labels that were not in the required labels to get a total of 200 remaining rows whose distribution is shown in the following figure 4.11 and you can see here that the fact checking websites focus more only on one domain.

```
BroadLabels
Politics            127
Health & Science     32
Fake                 20
Business              9
Entertainment         7
Sports                4
Technology            1
Name: BroadLabels, dtype: int64
```

Figure 4.11: Class count in Snopes dataset

3. Using the joblib module we load the TFIDF vectoriser and all the individual classifiers.

4. Next we combine the three key features('Noun Phrases', 'Nouns', 'Proper Nouns') into one and pass them into the TFIDF vectoriser for transformation. Note we only use the transform method here as the vectoriser was previously fitted on the key features of Washington post dataset.

5. After transforming, we predict the probability score for each of the individual classifiers and add that list as an extra column in the Snopes dataset, having a total of 7 additional columns consisting of prediction probabilities from the Individual classifiers.

6. Then we separate out all the 7 prediction probabilities' columns and the newly distributed labels column from the rest of the dataset and divide the dataset into dependent and independent variables.

7. After which we perform our classic split of the dataset into a training set and test set with 10% of data in the testing set and shuffle being true. We do so because we did not have enough fake dataset to properly train our classifier, which is a major problem in the domain of

fake news detection.

8. For the training task, we use a Decision Tree classifier here coupled with GridSearchCV for better results inputting parameters

```
parameters = {'criterion':['entropy','gini'],'max_depth':[20,50,100],
'min_samples_split':[2,4,6],'min_samples_leaf':[3,2,1],
'max_features':["log2","sqrt"],"min_weight_fraction_leaf":[0.5,0,0.2],
'random_state':[42]}
```

9. And then fitting the training sets for dependent and independent variables with a 10 fold cross validation.
10. After making the prediction on the test dataset, we display the accuracy and the confusion matrix given by the Decision Tree Classifier.
11. As a complete measure we take an extra step to train the Decision tree Classifier only on the TF IDF vector values and get their predicted accuracy and confusion matrix.

Observations

Although we achieve an accuracy of 45% in the given 6 labels. In the given figure 4.12, when we look at the confusion matrix we see that all the predictions are made in the "politics" label. This result comes when we are using only the predicted probabilities of the individual classifiers. The obvious conclusion is that the classifier is broken and in order to maximise in its prediction accuracy marking all the labels in the Politics label as it contains the largest number of samples. However,

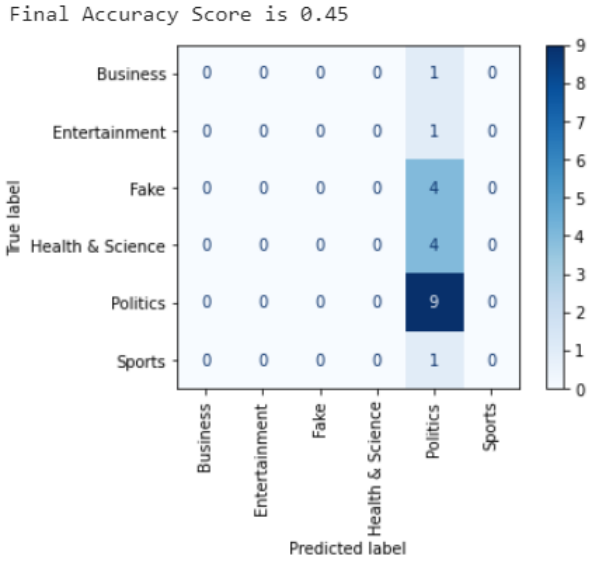

Figure 4.12: Confusion matrix (Prediction Scores-Snopes dataset)

when we train the model only on TF IDF vector features ignoring all the prediction probabilities we get a result in the figure 4.13 depicted below. Note how both the classifiers are giving us similar accuracy of 45% but the classification range of the classifier when given more features to work with

increases and it tries to predict other classes as well depicting a learning in progress. We may even say that a larger pool of features is able to contribute a better understanding of the labels to the machine learning classifier. A closer look at the metrics of the fake news labels tell us that the classifier is able to classify 25% (recall) of the total available fake news but when it does say that a certain piece of article is fake, then it says so with a 100% confidence (precision) which in turn might be useful in the general world. It should be noted here that our initial plan to obtain knowledge from the probability score fails and the two methods however are different with using TF IDF values as features being the superior of the two.

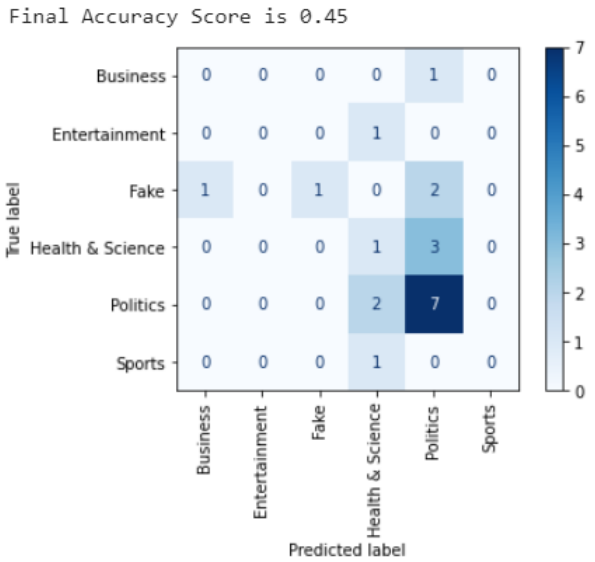

Figure 4.13: Confusion matrix (TFIDF only-Snopes dataset)

With the main crux of the problem being the imbalance in the dataset, we now move towards a more balanced fake news dataset that would be able to provide us with a fuller overview of things at hand.

Steps Involved (Experiment 4.3)

1. We start by loading the fake news dataset and then extract the three key features and the labels from the dataset.
2. Next we load all the modules of individual classifiers and the TF IDF vectoriser using the joblib library. The three key features are then combined and passed into the vectoriser as a single parameter that transforms the variable into a shape of (480, 2239755) on which we then perform individual classifications to obtain 7 columns of prediction probabilities that we append to the dataset.
3. Since we have only two classes here (Fake and legit) we don't need to broadly classify and remove them plus the two classes have an equal number of occurrences(each class containing 240) inside the dataset.
4. We then proceeded to split the data into training set and test set with a 90-10 split with shuffle being true rather than the usual 80-20 split because we did not have enough fake

dataset to properly train our classifier.

5. In this training process as well we are using a Decision tree Classifier coupled with a 5 fold cross validation (due to less number of classes) and then fitting the training sets for dependent and independent variables.
6. After making the prediction on the test dataset, we display the accuracy and the confusion matrix given by the Decision Tree Classifier.
7. Because the score features performed poorly in the previous experiment there is even more motivation to go ahead and perform the training process using only the TF IDF vector values. After which we perform the prediction and get our accuracy metric and confusion matrix.

Observations

We get a 41% accuracy (figure 4.14) from the predictions obtained using only prediction probabilities from the individual classifiers further materialising our inference that just using the probabilities is not a good feature for knowledge transfer and is certainly not a good feature for predicting fake news. But this time we did not get all the predictions in the same class meaning that it was the imbalance in the data that led to such biased results in the previous experiment. The results from the training process from when we only used the TF IDF vector values were used

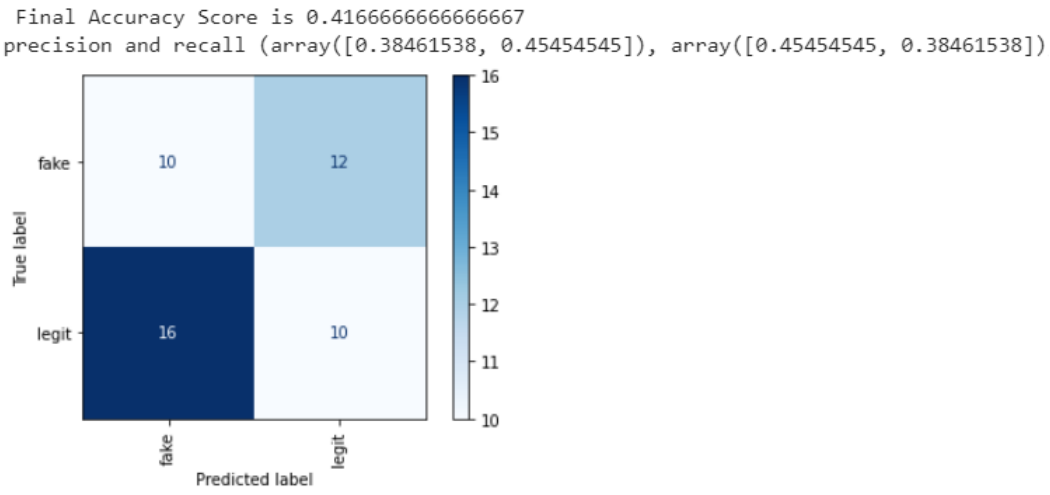

Figure 4.14: Confusion matrix (Prediction Scores-Fake dataset)

gives us an accuracy of 50% (figure 4.15) which is comparable to the accuracy received by our decision tree classifier in Experiment 3. However we would not say the results are reliable because even though our classifier is able to classify 90% (recall) of the positive classes, any class it says falls into the fake category has a less than 50% (precision) chance of actually being fake. On the other hand, this classifier is also not able to reliably predict the correct number of legitimate news as it misclassifies more than 80% of the legitimate news. One may say that we kind of can see the same pattern of mostly classifying all the classes into one category and achieving around 50% accuracy simply because the dataset is quite balanced and so would be the testing data so predicting all the labels into one class only should give you that. But our primary concern here for

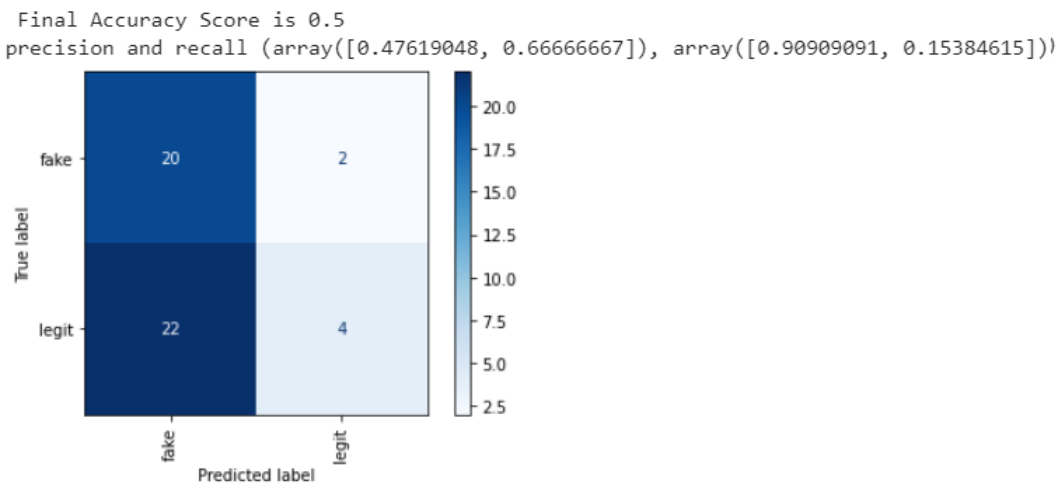

Figure 4.15: Confusion matrix (TFIDF only-Fake dataset)

rejecting this classifier is because it is not able to differentiate between the two classes or in other words have a higher precision and recall value for one of the classes.

Inference

From the above experiments and observations it seems pretty clear that we will not be able to transfer our knowledge simply through prediction probabilities or vector values fetched from a very large, organised and legitimate data source. The idea was to find out some patterns in the probability scores the pattern could be that all the scores were very high very low or zero or any other incomprehensible pattern but as we saw with our results in the Snopes dataset that is not the case and even if we don't consider that case due to imbalanced dataset we still have our results from the fake news dataset where using the scores for training gave us completely misclassified results. On the other hand, if we talk about the usefulness of vector values in knowledge transfer, we infer that while they were better suited to classify the Fake news in Snopes dataset and the fake news dataset, the final results were not promising enough to completely classify the whole experiment as a success. The vector values did do a better job of fake news classification and the reason to include these values was because legitimate news has a non-polarising, calm and neutral nature and it presents facts without taking any sides due to which we just might have found some keywords that were mutually exclusive.

We understand that detecting fake news is a hard task but this time we will be focussing on extracting out features that are universally available and can be extracted out from any news article. These sets of features are known as meta features and can be found from any text based on our general understanding of English Language as such.

4.5 Experiment 5

Hypothesis

Can fake news be classified using meta features extracted from the text

Aim

To find the correct set of meta features in the dataset to maximise the classifier performance and explain all the decisions behind each prediction made

Methodology

This time we would be choosing the features that can be universally extracted from any news article available. We would be looking at syntax of the text and use all the grammatical errors as a feature. The veracity of the content can also be assessed by analysing its readability indices and by including features like number of words, unique words, complex words, number of syllables and many others. We can assess the tone of the language by assessing whether the text is said in positive, negative or neutral emotion. There could be separation between the analysis of text and that of the headline. We can identify non-linguistic cues like sentiments by identifying syntactic patterns of content that can be evaluated to identify emotions from factual arguments by analysing patterns. All the above features can be somehow in one form or another be extracted and used for fake news classification.

Dataset

This dataset initially consisted of text and labels that specified the news to be fake or legit. With some experimentation it was devised that the headings should be a separate feature(as the headings have a higher importance than its content) in the feature domain and with some more trial and error method we reached our finally selected list of 19 features that are used in the classification of the fake news dataset:

1. Flesch-Kincaid Readability Score: From a variety of sources for assessing the readability index of an article, we use the widely recognised Flesch-Kincaid readability test for determining the readability of the article. The Flesch-Kincaid readability test is devised to indicate how difficult it is to understand a paragraph in English. It consists of two methods to assess the difficulty level namely Flesch-Reading ease and flesch-kincaid Grade level, which differ with each other slightly. We only choose the Flesch-Kincaid ease for our assessment, which depends on the word length and sentence length. In Flesch-Kincaid reading test ease, higher scores demonstrate the material that is easier to read and lower scored articles are harder to understand. The formula for Flesch-Kincaid reading test ease is (refer figure 4.16) The average score of a newspaper is about 60-80 which falls on the grade level that any person who cleared the 7th grade could understand. Law reviews and technical papers such as Harvard's are typically much lower in reading scores (around 30s). Readability scores is considered as an important feature in our experimentation as it revolves around the general

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

Figure 4.16: Formula of Flesch Kincaid Readability Score

text understanding capability of the public as it is highly considered by the governing bodies to enforce the difficulty standard of the text that goes into the making of insurance policies. An interpretation of the scores is given in the table below (figure 4.17):

Score	School level	Notes
100.00–90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	6th grade	Easy to read. Conversational English for consumers.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–10.0	College graduate	Very difficult to read. Best understood by university graduates.
10.0–0.0	Professional	Extremely difficult to read. Best understood by university graduates.

Figure 4.17: Flesch Kincaid Readability Table

2. Gunning fog Readability index: Is yet another readability standard for English language that estimates the years of schooling required for a person to understand the text written in English. For example an index of 11 means that the text is understandable by a high school senior. By the same logic an index of 8 is generally universally understandable. The Gunning Fog index is calculated by calculating the number of words and then dividing it by the number of sentences to get the average sentence length after which you need to count the number of complex words that have more than three syllables and divide it by the number of words to get the percentage of complex words which is added to average sentence length and multiplied by 0.4. The formula for Gunning Fog looks like figure 4.18.

$$0.4\left[\left(\frac{\text{words}}{\text{sentences}}\right) + 100\left(\frac{\text{complex words}}{\text{words}}\right)\right]$$

Figure 4.18: Formula for Gunning Fog score

Although it is a good readability score there are some limitations like not all words are complex for eg the word “interesting” is not really a difficult word but consists of three syllables and will be considered as complex by this readability score and on the other hand a short term like “bruit” can be considered as complex for it is not used very frequently by the general public. This feature may not seem to be adequate to be used for classification due to its limitations but a disability can also be considered as a special feature. An interpretation of the scores is given in the figure 4.19 below:

3. Automated Readability Index: It is yet another readability index for the English language

Fog Index	Reading level by grade
17	College graduate
16	College senior
15	College junior
14	College sophomore
13	College freshman
12	High school senior
11	High school junior
10	High school sophomore
9	High school freshman
8	Eighth grade
7	Seventh grade
6	Sixth grade

Figure 4.19: Gunning fog table

designed to capture the understandability of the text. Just like the Flesch Kincaid and Gunning Fog readability indices, this also offers a US grade level to comprehend the text but unlike them rather than counting the syllables in a word it counts the characters in a word with the general idea being that the more the characters in the word the less easy it is to read. The formula for calculating automated readability index is (see figure 4.20)

$$4.71\left(\frac{\text{characters}}{\text{words}}\right)+0.5\left(\frac{\text{words}}{\text{sentences}}\right)-21.43$$

Figure 4.20: Formula for Automated readability index

It was designed for military use in 1967 and was intended for real time monitoring of readability on electric typewriters. The Automated Readability Index was useful to them because calculating the characters per word is quicker than calculating the syllables per word and for more technical documents it seemed that efficiency was the key. Luckily, now we can automate all the readability indices but in that era a study conducted among different readability indices to find the efficient one in the lot, found that the Automated Readability Index was slightly more reliable for military purposes than Flesch Kincaid Score. An interpretation of the scores is given in the figure 4.21:

4. Errors: This feature counts the number of grammar errors and performs a series of spell checks on the given text. To find errors in the sentences we take the help of a library called LanguageTool which makes use of large n-gram datasets to detect errors with words that are often confused like "their" and "there" and its spell checker utilises 3 text files consisting of a million words, each specifying whether the said word is correct, incorrect or if it should be ignored. Because of the current nature of the dataset and fake news in general which is not usually produced by the journalists in a professional setting we are confident to find some strong correlations between the number of errors made by the general public vs that of the professional journalists.

5. Suggestions: This feature comes from a library called proselint which does not solely focus on

Score	Age	Grade Level
1	5-6	Kindergarten
2	6-7	First/Second Grade
3	7-9	Third Grade
4	9-10	Fourth Grade
5	10-11	Fifth Grade
6	11-12	Sixth Grade
7	12-13	Seventh Grade
8	13-14	Eighth Grade
9	14-15	Ninth Grade
10	15-16	Tenth Grade
11	16-17	Eleventh Grade
12	17-18	Twelfth grade
13	18-24	College student
14	24+	Professor

Figure 4.21: Automated readability Index Table

grammar and looks at more things like redundancy, jargon, illogic cliches, misspelling, inconsistency, misuse of symbols etc. It also provides you with suggestions(one of its features) for making the text more suitable for the reader which we use as an extension of the above feature "errors". We use this feature here as it not only points out the location of the error but also gives you the nature of the error so that you can correct it, which also might help in our explainability endeavour. An example of a suggestion is given in the figure 4.22 where we can see that the suggestions offered are not for spelling mistakes but are so trivial that they could have been easily mitigated if the writer was more careful while writing once again drawing the line between a professional and a normal person.

```
[('consistency.spacing', 'Inconsistent spacing after period (1 vs. 2 spaces).',
0, 149, 149, 152, 3, 'warning', '[\\.\\?!]  [A-Z]'),
('typography.symbols.curly_quotes', 'Use curly quotes “”, not straight quotes
"".', 0, 585, 585, 587, 2, 'warning', '“ or ”')]
```

Figure 4.22: Grammar suggestions

6. Sentiments: Finding the sentiments or the opinions with which the texts were written is considered equally important as to extracting the syntactic errors. Sentiment analysis aims to measure the sentiments, attitude and emotions of the speaker. This feature utilises the nltk.sentiment module from NLTK library along with the Liu and Hu opinion lexicon which consists of all the positive and negative words in separate files and by broadly studying a sentence it could be classified into positive, negative or neutral class based on the number of polarising(positive/negative) words in an article. Getting the overall sentiment of the news article seems like a good feature as we expect most of the professionally written articles should be falling in the neutral category.

7. Subjectivity: This feature categorises whether the given text is subjective or objective in nature meaning that whether it expresses its opinions in an open and maybe a polarising way, influenced by personal feelings or whether it expresses true facts and figures without resorting

to any kind of bias caused by personal sentience or external influence. This feature utilises a pre trained Naive Bayes Classifier that has been trained on 10000 tweets for classifying them as subjective or objective.

8. Positive Polarity: This feature gives you the normalized sentiment strength of the positive emotion that is captured throughout the given text. It tells you the percentage of text that conveys positive emotions. This feature helps breakdown the overall sentiment into its compound features and could be helpful in the articles that have been classified as negative by a small margin.
9. Negative Polarity: This feature gives you the normalized sentiment strength of the negative emotion that is captured throughout the given text. It tells you the percentage of text that conveys negative emotions. This feature helps breakdown the overall sentiment into its compound features and could be helpful in the articles that have been classified as positive by a small margin.
10. Neutral Polarity: This feature gives you the normalized sentiment strength of all the neutral words that are captured throughout the given text. It tells you the percentage of text that is neutral in nature. Generally everything that does not fall into a positive or negative bag of words, is considered to be neutral.
11. Compound Polarity: It is the normalized sum of the actual positive and actual negative polarities(not the normalized ones). This feature is highly correlated with the overall sentiments feature but the difference occurs in the neutral territory where the compound polarity might be leaning towards the positive or the negative side but the overall sentiment is still classified as neutral as both of these features are calculated using different formulas for eg a compound polarity of -0.92 and 0.82 should have an overall sentiment of negative and positive but they are actually classified as neutral (normalized values only range from -1 to 1).
12. Positive Polarity of heading: This feature gives you the normalized sentiment strength of the positive emotion captured in the Heading of the article. It was found that people might use excessively positive or negative emotions in the headlines to grab the attention of the usual reader so the headlines were evaluated separately for their polarising words.
13. Negative Polarity of heading: This feature gives you the normalized sentiment strength of the negative emotion captured in the Heading of the article.
14. Neutral Polarity of heading: This feature gives you the normalized sentiment strength of the neutral emotion captured in the Heading of the article.
15. Compound Polarity of heading: It is the normalized sum of the actual positive and actual negative polarities captured in the headings of articles.
16. Word Counts: As indicated by all the readability scores that utilises the word counts as variable in calculating the scores we felt that this feature should be included as a separate feature also because it was determined that a liar uses more words to get their points across while the truth is readily believable and does not require much convincing.

17. Unique Words: This feature counts the number of words in a text that have more than three syllables in them while the general understanding being that people who are not in the writing business don't have a vast vocabulary of words to choose from and hence may use "wise" instead of "knowledgeable".
18. Number of Capital words: We are looking for all the words written completely in capital letters which is probably written to grab the eye of the viewers and could possibly consist of some polarising content inside.
19. Number of punctuations: Some of the punctuation characters include commas, periods, question marks and exclamation marks. Here exclamation marks hold significance because when used right after a word that holds a positive or a negative emotion greatly increases the meaning of that word. Also previous work on fake news detection has suggested that the use of punctuation might be useful to differentiate between deceptive and truthful texts.

Steps Involved

1. We load the dataset and then separate the dependent and independent variables. Since we have the complete text as our independent variable we need to further divide it into Headings and Article content.
2. With the headlines and content being separate, we start by extracting out the grammatical and syntactical errors in the content with the help of our LanguageTool library and count the number of errors to be used as a feature.
3. Next we analyse the text to extract some suggestions relating to syntax and grammar with the help of a library known as proselint which acts like a computer program that scans through the document like a spell checker to identify errors and give us their corresponding suggestions and we use the number of suggestions given for a text as another feature.
4. Then we perform the sentiment analysis on both the heading and the content with the help of a three corpus namely "opinion_lexicon", "vader_lexicon" and "subjectivity" provided to us by the NLTK library. The opinion Lexicon consists of two files containing all the possible positive and negative terms and in order to find the overall sentiment of the text we need to first tokenize the text into individual words and then check every word inside the lexicon to get the number of positive and negative terms, whose sum gives us the overall sentiment to be positive(if +ve), negative(if -ve) and neutral(if 0). For finding the subjectivity of a text we load the "subjectivity" file which is a pre-trained Naive Bayes Classifier and requires input in the form of individually tokenized lowercase words to classify text as subjective or objective. Lastly, we have the "vader_lexicon" that consists of all the dictionaries each of which intricately defines all the words and their corresponding score of emotional valence which it uses to tell us how positive or negative a sentiment is.
5. Next we calculate the readability scores with the help of a library called Readability where we input the entire text and get the readability indices of Flesch-Kincaid, Gunning Fog and Automated readability index. In addition to these scores it also gives us the word counts and the number of unique words to be used as a feature.

6. To calculate the number of capitals we need to separate all the words and check each and every one of them for an all caps word. Similarly, for finding the punctuations we compare all the white space separated characters with a list consisting of punctuation characters.
7. After all the features are extracted we encode the two features of "errors" and "suggestions" with Label Encoder and perform the train test split with a 90-10 partition
8. Then we be gin the training process of fake news classification using both our Decision Tree Classifier and Ridge Classifier to check which one performs better.
9. Finally, we draw out all the metrics like accuracy, precision, recall and plot the confusion matrix and display the explainable training graph

The results would be discussed in the next section.

5 Research Results

We know that the Ridge classifier has been able to outperform our Decision Tree classifier on the previous accounts of multi class classification but as you can see in the figure 5.1 in the case of fake news classification our Decision Tree Classifier outperforms the Ridge Classifier by a margin of 4% in accuracy metric. Because the margin is not huge we can say that both the classifiers give a comparable performance with the accuracy of Ridge Classifier being 66% and that of Decision Tree being 70%.

```
Ridge Classifier
Accuracy 0.6666666666666666
Confusion matrix, without normalization
[[19  3]
 [13 13]]
Normalized confusion matrix
[[0.86363636 0.13636364]
 [0.5        0.5       ]]
Precision, Recall, Fscore (array([0.59375, 0.8125 ]), array([0.86363636, 0.5       ]), array
================================================================================
DecisionTreeClassifier
Accuracy 0.7083333333333334
Confusion matrix, without normalization
[[15  7]
 [ 7 19]]
Normalized confusion matrix
[[0.68181818 0.31818182]
 [0.26923077 0.73076923]]
Precision, Recall, Fscore (array([0.68181818, 0.73076923]), array([0.68181818, 0.73076923]),
```

Figure 5.1: Performance metrics for Ridge Classifier and Decision Tree

Next we look at the confusion matrix for both the classifiers shown in the figures5.2 and 5.3 below. From the looks of it we infer that even though the accuracies are comparable, the Ridge Classifier does a poor job at detecting fake news since most of its predictions are made in the fake news category which in turn also affects its precision score for detecting fake news and hence we would have not much confidence in it when it classifies a piece of news as fake. On the other hand, Our decision tree does a reliably better job at classifying both the classes with the precision and recall scores for both the classes being at the brighter side due to which we can have a good amount of confidence in its classifications.

We will now look at the explainable graph (figure 5.4) generated by the decision tree that gives us a complete insight into the decisions made.

We start by looking at the root node (node 1) where the tree will choose the feature that gives us the maximum separability (by aiming for maximising the Information Gain in its resultant two child nodes) between the two classes and so it chooses the "number of Errors" as the root feature. The gini score at node 1 is 0.5 because both the classes are essentially equal in numbers in the training

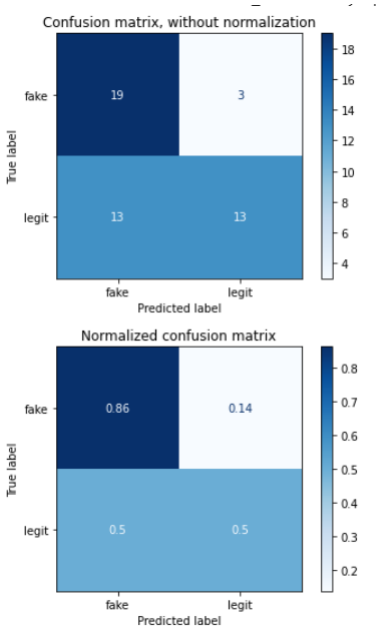

Figure 5.2: Confusion Matrix (Ridge Classifier)

set at 218 for fake and 214 for legit, the decision made however reduces the gini score in both its leaf nodes, node 2 and 3 for a score of 0.48 and 0.35 respectively only because the information gain is maximised but this is not the case at node 9 where one of the gini impurity increases but information gain is still maximised. Contrary to our popular belief of fake news having more grammatical mistakes, we get more number of legit classes(79) than fake classes(24) at node 3 saying that all of these classes have more than 11.5 number of errors (as per the decision made by node 1) but in line with our belief that legit classes will have a greater number of punctuations is our sample values at node 7 with a 24,48 split saying all of these classes have more number of punctuations than 13.5 (as per decision at node 3). Also, we can infer that "punctuations" is a good feature giving us a pure split(node 6) with 31 samples at its first ever decision made at node 3 and it produces a leaf node again at node 9 with 33 samples. Similarly we can draw a multitude of explanations from different features but it must be noted that a decision made at node 7 is not a standalone decision, it is made keeping in mind the previous separations made at node 3 and 1.

From a decision tree graph we can also make out whether a decision tree is overfitting or not if we look at the node 13 we see that the tree is branching(making a decision) specially to classify that 1 row belonging to a particular class instead of classifying more than one class. The branching that leads to node 14 is acceptable as the tree is also classifying 22 other class labels with that decision at node 15. Too much of leaf nodes like node 13 and your classifier is overfitting the data.

The decision tree graph gives us a very clear understanding of all the decisions that took place in the prediction process along with the values at which the prediction is taking place so that you can trace your way in the decision tree but let's suppose you don't want to drag your finger around a big decision tree then we can also give you the personalized decisions made for that particular entry in the dataset like as shown in figure 5.5

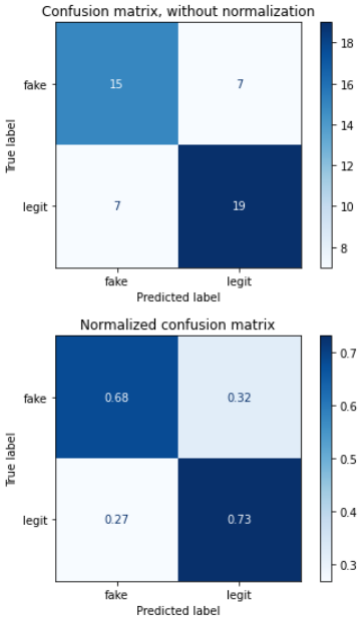

Figure 5.3: Confusion Matrix (Decision Tree Classifier)

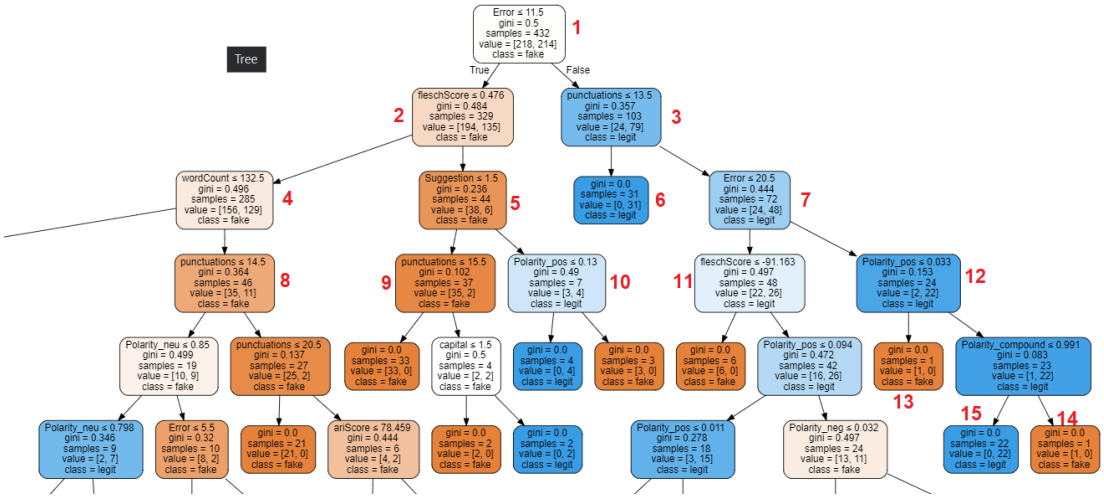

Figure 5.4: Explainable Decision Tree Graph

```
Rules used to predict sample 0:
decision id node 0 : (X_test[0, 0] (Error = 5) <= 11.5)
decision id node 1 : (X_test[0, 14] (fleschScore = -42.4822727272727) <= 0.47618934884667397)
decision id node 2 : (X_test[0, 12] (wordCount = 110.0) <= 132.5)
decision id node 3 : (X_test[0, 18] (punctuations = 16) > 10.5)
decision id node 41 : (X_test[0, 4] (Polarity_pos = 0.054000000000000006) <= 0.14100000262260437)
decision id node 42 : (X_test[0, 0] (Error = 5) <= 5.5)
decision id node 43 : (X_test[0, 8] (Head_Polarity_pos = 0.0) <= 0.11800000071525574)
decision id node 44 : (X_test[0, 8] (Head_Polarity_pos = 0.0) <= 0.05550000071525574)
decision id node 45 : (X_test[0, 14] (fleschScore = -42.4822727272727) <= -13.80526351928711)
decision id node 46 : (X_test[0, 1] (Suggestion = 2) > 0.5)
decision id node 50 : (X_test[0, 7] (Polarity_compound = -0.4019) <= 0.9233500063419342)
decision id node 51 : (X_test[0, 9] (Head_Polarity_neg = 0.0) <= 0.6324999928474426)
decision id node 52 : (X_test[0, 18] (punctuations = 16) <= 18.5)
```

Figure 5.5: Decision made for the prediction of a Sample

6 Conclusion

In conclusion, we were successfully able to classify fake news though not with the method that we initially intended. A decision tree is able to give us a good enough explanation about the decision it took to make any of the predictions and it is up to us whether to agree or disagree with any of the decisions made. The explainability of decision trees comes off at a tradeoff for a lesser accuracy value but still our classifier manages to pull off a decent enough accuracy score. Our work provides a valuable contribution in the domain of fake news detection by providing a comprehensive explainability feature that was otherwise absent from the domain. This feature would also provide a helping hand for all the fact checking experts and help them in explaining the results obtained.

One of the ways we could have improved the accuracy would have been by trialling with some more number of features before fixating on the 19 selected features. We also could have included a better set of n-grams (like including emotion words) to expand our set of vocabulary in our initial set of experiments.

One of the ways how we can preserve the knowledge between different datasets is by building a knowledge graph. A knowledge graph essentially stores all the entities and the relationships between them, acting like a fact storage database. What more is that this data storage can grow incrementally with more data being fed to it and with enough amount of stored data it can act as a fact-checker for verifying claims made in the news.

Bibliography

[1] Joseph Issac. Why people fall for misinformation. https://ed.ted.com/lessons/why-people-fall-for-misinformation-joseph-isaac#watch.

[2] Peter Dizikes. Study: On Twitter, false news travels faster than true stories. https://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308.

[3] Camilla Domonsoke. Students Have 'Dismaying' Inability To Tell Fake News From Real, Study Finds. https://www.npr.org/sections/thetwo-way/2016/11/23/503129818/study-finds-students-have-dismaying-inability-to-tell-fake-news-from-real?t=1599473898608.

[4] Washington Corpus. TREC Washington Post Corpus. https://trec.nist.gov/data/wapost/.

[5] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news, 2017.

[6] Jason Brownlee. A Gentle Introduction to the Bag-of-Words Model. https://machinelearningmastery.com/gentle-introduction-bag-words-model/.

[7] WikiMedia Foundation. n-gram, . https://en.wikipedia.org/wiki/N-gram.

[8] Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853 – 860, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.08.015. URL http://www.sciencedirect.com/science/article/pii/S0957417413006271. Methods and Applications of Artificial and Computational Intelligence.

[9] WikiMedia Foundation. Part of Speech , . https://en.wikipedia.org/wiki/Part_of_speech.

[10] Sachin Malhotra, Divya Godayal. An introduction to part-of-speech tagging and the Hidden Markov Model. https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/.

[11] Beattrice Santorini. Part-of-Speech Tagging Guidelines for the Penn Treebank Pro ject . https://sharedtasksinthedh.github.io/assets/howto-annotation/Penn-Treebank-Tagset.pdf.

[12] Susan li. Named Entity Recognition with NLTK and SpaCy. https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da.

[13] niklas donges. INTRODUCTION TO NLP. https://builtin.com/data-science/introduction-nlp.

[14] Lexanalytics Inc. Sentiment Analysis Explained. https://www.lexalytics.com/technology/sentiment-analysis.

[15] Monther Aldwairi and Ali Alwahedi. Detecting fake news in social media networks. *Procedia Computer Science*, 141:215 – 222, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.10.171. URL http://www.sciencedirect.com/science/article/pii/S1877050918318210. The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops.

[16] Akshay Kulkarni and Adarsha Shivananda. *Converting Text to Features*, pages 67–96. Apress, Berkeley, CA, 2019. ISBN 978-1-4842-4267-4. doi: 10.1007/978-1-4842-4267-4_3. URL https://doi.org/10.1007/978-1-4842-4267-4_3.

[17] Bruno Stecanella. What is TF IDF. https://monkeylearn.com/blog/what-is-tf-idf/.

[18] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[19] Julio Reis, André Correia, Fabricio Murai, Adriano Veloso, and Fabrício Benevenuto. Explainable machine learning for fake news detection. pages 17–26, 06 2019. doi: 10.1145/3292522.3326027.

[20] Elvis. Deep Learning for NLP: An Overview of Recent Trends. https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d.

[21] Thota, Aswini; Tilak, Priyanka; Ahluwalia, Simrat; and Lohia, Nibrat. Fake News Detection: A Deep Learning Approach. https://scholar.smu.edu/datasciencereview/vol1/iss3/10.

[22] Maite taiboada. Misinformation Detection. https://github.com/sfu-discourse-lab/Misinformation_detection.