# Hyperpartisanship in Web Searched Articles

Anamika Ashit Sen

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

A. Lynn Abbott, Chair
Jiepu Jiang
Ryan Williams

June 17, 2019
Blacksburg, Virginia

Keywords: Hyperpartisanship, news, fake news, natural language processing,
propaganda, misinformation

# Hyperpartisanship in Web Searched Articles

Anamika Ashit Sen

(ABSTRACT)

News consumption is primarily done through online news media outlets and social media. There has been a recent rise in both fake news generation, and consumption. Fake news refers to articles that deliberately contain false information to influence readers. Substantial dissemination of misinformation has been recognized to influence election results. This work focuses on hyperpartisanship in web-searched articles which refers to web searched articles which have polarized views and which represent a sensationalized view of the content. There are many such news websites which cater to propagating biased news for political and/or financial gain. This work uses Natural Language Processing (NLP) techniques on news articles to find out if a web-searched article can be termed as hyperpartisan or not. The methods were developed using a labeled dataset which was released as a part of the SemEval Task 4 - Hyperpartisan News Detection. The model was applied to queries related to U. S. midterm elections in 2018. We found that more than half the articles in web search queries showed hyperpartisanship attributes.

# Hyperpartisanship in Web Searched Articles

Anamika Ashit Sen

(GENERAL AUDIENCE ABSTRACT)

Over the recent years, the World Wide Web (WWW) has become a very important part of society. It has overgrown as a powerful medium not only to communicate with known contacts but also to gather, understand and propagate ideas with the whole world. However, in recent times there has been an increasing generation and consumption of misinformation and disinformation. These type of news, particularly fake and hyperpartisan news are particularly curated so as to hide the actual facts, and to present a biased, made-up view of the issue at hand. This activity can be harmful to the society as greater the spread and/or consumption of such news would be, more would be the negative decisions made by the readers. Thus, it poses a bigger threat to society as it affects the actions of people affected by the news. In this work, we look into a similar genre of misinformation that is hyperpartisan news. Hyperpartisan news follows a hyperpartisan orientation - the news exhibits biased opinions towards a entity (party, people, etc.) In this work, we explore to find how Natural Language Processing (NLP) methods could be used to automate the finding of hyperpartisanship in web searched articles, focusing on extraction of the linguistic features. We extend our work to test our findings in the web-searched articles related to midterm elections 2018.

# Dedication

*This Thesis is dedicated to my Family: Parents - Mitali and Ashit Sen, and Sister - Malabika Sen.*

# Acknowledgments

There are a lot of people I would like to extend my heartfelt thanks to, here at Virginia Tech. Firstly, I would like to express my sincere gratitude to my supervisor and guide - Dr. Jiepu Jiang for his tremendous help in this thesis - right from inception to its completion. His ideas and guidance paved my path to partake in this exciting field of research. I would also like to extend my thanks to both Dr. Lynn Abbott and Dr. Ryan Williams for agreeing to serve on my committee. I am thankful to Dr. Abbott for his continuous and helpful revisions to this thesis.

I would also like to thank all the people at NDSSL, whom I had a chance to work with or merely interact. I did learn a lot working on exciting projects here. I thank them and the ECE department for providing me the financial assistance to pursue my Masters at Tech.

I would also like to thank all my friends here at Virginia Tech - Soham Zodpey, Mihir Kulkarni, Subhashree Radhakrishnan and many others from Fall 2017 and 2018 batch, who have been a constant source of motivation, encouragement and fun. Last but not the least, the beautiful town and people of Blacksburg which made me feel welcomed the whole 2 years of my stay here.

Lastly, I would like to dedicate this work to my family - Mitali Sen, Ashit Sen and Malabika Sen for their immeasurable support, love and belief in me which has made it possible for me to achieve my goals.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The recent surge in consumption of articles through online media has lead to a paradigm shift in the way people keep themselves updated. Memes, tweets, social media posts and online news are some of the ways that readers are entertained, informed and even influenced to make online purchases. Thus, the Internet has become a source to influence people as it has a faster dissemination than the tradition physical medium. According to Allcott and Gentzkow[7], 14% of Americans considered social media as their "most important" source of information. This can be worrisome as there is a recent trend in effortless propagation of fake news through social media[31]. Propagation and increasing readership of fake news is not healthy as it can affect decisions of a huge population/community, as seen in the 2016 U. S. presidential elections[7, 18, 30, 31]. Generation, consumption and propagation of content on over the Internet is illustrated in Figure 1.1.

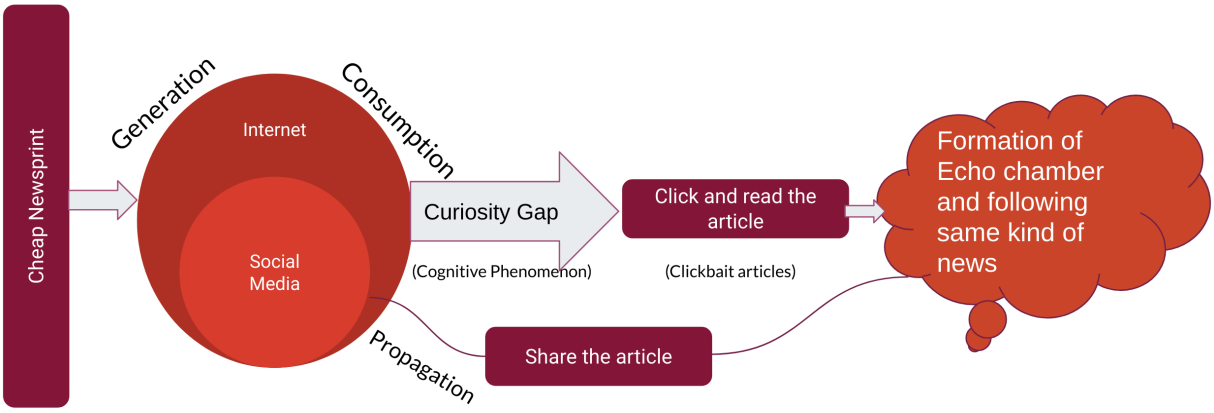



Figure 1.1: Generation, consumption and propagation of content over the Internet

As seen from Figure 1.1 above, creating cheaper content online has led to an increase in generation and easy access of any kind of content all over the world. We consider social media as a subset of the Internet. The process of coming across an article online and actually clicking and reading it is driven by a cognitive phenomenon called as the 'Curiosity Gap'. The article must have come across the reader on the Internet due to his/her recent engaging activities in this genre or because one of their followers on the social media shared it. This formation of a bubble wherein one encounters content engaging and entertaining to their own taste is called an *echo chamber*. These kinds of content have a polarizing and engaging nature to their readers, and consist of anything that one consumes online - including the fake and hyperpartisan news. Consumption of such content could be harmful as information overload of any kind influences human decision making[13].

Detection and control of fake news has become an emerging research topic as the consequences of it can be alarming. Fake news is intentionally written so as to engage and mislead readers into believing false information. Increase in this activity is mainly for financial and political gain[31]. In this work, we look into detecting hyperpartisanship in

web-searched articles.

According to Potthast et al.[26], *hyperpartisan* news follows a hyperpartisan argumentation which means that it exhibits blind, prejudiced or unreasoning allegiance to one party, faction, cause, or person. The content in such articles are not fabricated, although they are suggested in a way which stimulates emotional feelings among the readers. Such polarizing content is generated in order to create a partisan divide among the content consumers. Other than news, hyperpartisan content can be propagated through blog posts, and tweets, etc. Research related to hyperpartisanship is relatively new compared to detection of fake news[31], differentiating between fake news and satire[28] and clickbait detection[11]. Although these are named differently, these different topics more or less fall under the same umbrella: they differ from actually reporting facts.

The premise for this study is to find out cues for hyperpartisanship in web-searched articles related to U. S. midterm election in 2018[3].

## 1.2 Contribution

Our main contribution is to find hyperpartisan cues using both the headlines and the body of any text article. We used a novel approach of using presence of morphological text features (nouns and adjectives) and their sentiments for classification. Additionally, we have also employed finding clickbait traits in hyperpartisan articles using Term Frequency and Inverse Document Frequency (TF-IDF) vectorizer and n-grams. In this work, we start with the modeling of our system to detect hyperpartisanship in news through the dataset provided by PAN @ SemEval 2019[5]. We report our results on web-scraped articles related to U. S. midterm elections in 2018. The web articles were collected using BING web search API[1] and the queries were curated using Google Trends[2] starting with the

search term 'United States midterm election' and iteratively expanding the related queries to a total of 86 queries. We found that more than half of the web-queried results showed hyperpartisanship attributes.

## 1.3 Problem Statement

This thesis covers the following problem:

HYPERPARTISANSHIP DETECTION IN WEB-SEARCHED ARTICLES

**Input:** A web-searched article $A$

**Output:** If the web-searched article $A$ follows the hyperpartisanship argumentation or not.

$$H(A) = \begin{cases} 1 & \text{if } A \textit{ is hyperpartisan} \\ 0 & \text{if } A \textit{ is not hyperpartisan} \end{cases}$$

## 1.4 Outline of Thesis

This thesis is organized as follows.

Chapter 2 gives an overview of the related work done in this field. Chapter 4 discusses the methodology and the algorithms implemented. It further explains the data curation and the modeling of the system. It also lists the results obtained using our primary dataset. In Chapter 5, we discuss our findings on the web-searched articles related to midterm elections 2018. Finally, Chapter 6 concludes this thesis and provides a brief summary related to future work and development.

# Chapter 2

# Review of Literature

Work on detecting hyperpartisanship in news has not been much explored. Potthast et al.[26] explored the technique of 'Unmasking'[22] to find out distinction between writing styles of right-wing and left-wing together (hyperpartisan) and mainstream media and detection of fake news. They used features like n-grams, readability scores and frequency of words using General Inquirer Dictionaries[32] as features for classification in addition to news-domain specific features like ratios of quoted words and external links, the number of paragraphs in the news article and their average length. Their work fared well to distinguish hyperpartisan news from mainstream news ($F1 = 0.78$) as well as from satire ($F1 = 0.81$). Since detection of extremism in text articles is what the aim of this study is, it can be compared to detecting fake news. Thus, we also review the work done in fake news detection as the research done on it can be used as a basis for our work.

As earlier discussed, the motive of generation of fake news is for financial and political benefit. These online articles are written in a way which generate interest in readers due to their disputable nature and are termed as 'clickbaits'. Chakraborty et al.[11] studied how to detect and prevent clickbaits which exploit the cognitive phenomenon called a '*Curiosity Gap*' [23]. They found that traditional news headlines (non-clickbait) contain *content words* referring to people and/or location whereas clickbait headlines are longer and contain both *content* and *function words*. Most text-processing steps involve removal of stop-words, however in non-clickbait articles, inference of stop-words is left to the readers

and thus it was used in the modeling of the classifier. Additionally, clickbaits used 'very positive' sentiment words like *'breathtakingly', 'gut-wrenching'*, internet slangs like 'LOL', 'LMAO' along with punctuation patterns. This work achieved 93% accuracy in detecting clickbaits.

Shu et al.[31] did a comprehensive review of the characterization of fake news using psychology and social studies and then detection of fake news. Detection of fake news is not a trivial problem since they are knowingly written to mislead readers, citing true evidence to support an indisputable claim[16]. The authors say the psychological foundations of fake news which make humans vulnerable to its consumption are the following:

- *Naive Realism*: A reader's belief that one's perception of reality is the only accurate one, and not being open to discussing other's views and beliefs[35].
- *Confirmation Bias*: A reader favoring information which already confirms to his/her existing rationale/views.

Fake news propagates faster than traditional physical medium (newspapers) through social media. The low cost of creating social media accounts encourages the creation of malicious user accounts, many of these accounts being social bots controlled by automation. Social bots and Russian trolls on social media greatly affected the 2016 United States presidential elections[4, 8]. In social media, there are additional auxiliary information in addition to news content which can be used as features in fake news detection[11]. The metainformation used in this work[11] are the source, i.e., the author or the news publisher, headlines, body text and the image/video content in the news body. Visual cues in images and thumbnails of videos help supplement the spread of fake news. Social context features on an individual and group level also help in understanding the reach and propagation of news. Individual level features include user demographics like registration age, number of

followers/followees and number of tweets authored by the user[10]. Group level features include the overall perception of news by a group[38]. Network based features are used to identify dissemination of fake news within a social network since the spread of articles follows an echo chamber cycle.

Satire in news articles can be misunderstood by some readers, as the message is not delivered plainly, rather topped with irony and double-meanings[28]. The paper claims satire and fake news to be a part of deception, although the latter is not intended to be found by the audience. Rubin et al.[28] proposed an SVM based algorithm using 5 predictive features on 360 news articles to detect how satire can be differentiated from fake news. Their best predicting feature combination was *absurdity*, *grammar* and *punctuation* which detected satirical news with an accuracy of 90% and 84% recall.

As discussed earlier, social bots without human interaction have been used to spread misinformation. In addition to being a threat to democracies, misinformation can lead to readers making dangerous health decisions[19] as well as manipulation of stock markets[17]. In their work, Shao et al.[30] studied the large scale systematic analysis of the spread of fake news. They crawled 122 websites which routinely publish fake news according to established media. Their analysis showed that these websites publish approximately 100 articles per week on average. However, the virality of these articles is due to the tweets, on an average of 30 tweets per article per week. The authors showed that the super-spreaders of these misleading articles are the social bots which perform activities like posting links to articles, retweeting other accounts and performing other automated tasks like following and replying to other users. These activities help in amplifying fake news even before a claim goes viral. Social bots use other strategies like mentioning influential people/journalists, politicians in their tweets linking to the debunked claim. This leads to a false appearance that the news is widely shared and the chance that followers of these bot accounts will

share it as well. In addition, social network behavior also can be used as an important characteristic in deception detection as authentic social media profiles equates to trustful posts. Similarly, inclusion of hyperlinks and/or associated metadata can be supplemented to establish proof of veracity[15].

Conroy et al.[14] provide a description of a variety of veracity assessment methods in deception detection. Linguistic cue approaches, network analysis and a hybrid of the two methods are currently used in fake news detection. In the first method, language patterns are extracted to find associative deceptive patterns from word-level to a more discourse-level, whereas in the second method, message metadata or the structured data network are used. Structured data network includes finding a shortest path between generic entities in a statement represented in a graph network. This paper also talks about the inclusion of publicly available gold-standard/benchmark datasets to assist in fact-checking. Thus, in Chapter 3, we review the current datasets in practice, with the details of the dataset we have worked on, and developed.

Clickbaits work on the sole principle that the headlines or the article title should evoke curiosity in the readers' mind. Bourgonje et al.[9] claim that detecting if the headlines bear any relation to the article body should be the first step in clickbait/fake news detection. Thus, they worked on stance detection of headlines with respect to their article bodies as a part of the first Fake News Challenge[1]. Their setup is is based on a lemmatization based n-gram matching using a combination of three binary classifiers which resulted in a weighted score of 89.59% accuracy.

Building on these previous works, we take into account features like length of the articles, writing style of headlines, etc.

In the next section, we go through the current benchmark datasets in practice. We

[1]http://www.fakenewschallenge.org/

also detail the online NLP data challenge that we have worked upon as a part of our research. The data challenge is task 4 of SemEval workshop: Hyperpartisan News Detection[2]. We were provided with a labeled dataset for this challenge and our experiments were conducted using this dataset.

[2]https://pan.webis.de/semeval19/semeval19-web/

# Chapter 3

# Dataset

Due to the recent emergence in fake news and increasing interest in its detection, the resources for its research are still in the infancy stage. Labeling a dataset requires Subject Matter Experts (SMEs) to annotate an article, and the notion of what is *fake* and what is *real* can be fuzzy since such articles are intended to deceive the readers. The lack of reliable benchmark datasets poses a significant challenge in advancing the research. In this section, we review the current labeled datasets available as well as talk about the SemEval Task 4 - Hyperpartisan News Detection[1].

## 3.1 Current Datasets In Practice

In the subsequent sub-sections, we review the current datasets in practice used in deception detection - fake news detection, clickbait detection, stance detection and hyperpartisan news detection.

[1]https://pan.webis.de/semeval19/semeval19-web/

### 3.1.1 Emergent

Vlachos and Riedel[33] proposed using data from fact-checking websites like POLITIFACT.COM[2], a Pulitzer prize-winning website and FULLFACT.ORG[3] (2014). In 2017, Ferrara et al.[17] created a rumor-debunking dataset containing 300 rumored claims and 2595 associated news articles, manually annotated by journalists. These claims were collected by journalists from websites like snopes.com and twitter accounts like @Hoaxalizer. Each claim has sourced news articles, a stance (*for*, *against*, *observing*), article headline and veracity (*true* or *false*). Although a pioneer in creating a dataset for stance detection, the number of samples are quite low in this dataset.

### 3.1.2 LIAR Dataset

The LIAR dataset by Wang[34] is a publicly available dataset for fake news detection. The dataset consists of 12800 manually annotated short statements in various states of affair from POLITIFACT.COM collected over a decade. This dataset has a comprehensive collection of detailed analysis reports and source links to all the documents. The truthfulness ratings are based on 6 fine-grained labels: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true* and *true*. The distribution of articles can be seen in Table 3.1.

The dataset contains snippets from speakers affiliated to both Democrats and Republicans as well as social media posts, such as facebook posts. In addition to party affiliation for speakers, the dataset also contains other metadata like their current job, credit history, home and state.

[2] www.politifact.com
[3] fullfact.org

Table 3.1: Distribution of articles in the LIAR dataset

| pants-fire | false | barely-true | half-true | mostly-true | true |
|---|---|---|---|---|---|
| 1047 | 2507 | 2103 | 2627 | 2454 | 2053 |

### 3.1.3 BuzzFeed News Dataset

BuzzFeed published an article '*Inside The Partisan Fight For Your News Feed*'[4] which identified 667 websites as partisan news outlets along with the associated Facebook pages (452). The dataset is available in the GitHub repository[5].

### 3.1.4 Fake News Challenge Dataset (FNC-1)

FNC-1 dataset aims to use Artificial Intelligence(AI) to find if a headline and the body of the text are related to each other, i.e., *stance detection*. The data consists of (*headline, body, stance*) where stance could be any of the following: (*agrees, disagrees, discusses, unrelated*). The dataset extends the work of Ferrara et al.[17] and the dataset can be found in this GitHub repository[6].

### 3.1.5 Kaggle Fake News Dataset

Kaggle's '*Getting Real about Fake News*' dataset consists of articles scraped from 244 websites amounting to a total of 12,999 posts. These websites were scraped using a chrome extension - *BS Detector*[7]. The samples in the dataset seem rather too obvious and/or extremely fake, which is different from the problem at hand - detecting intentionally mis-

[4]https://www.buzzfeednews.com/article/craigsilverman/inside-the-partisan-fight-for-your-news-feed
[5]https://github.com/BuzzFeedNews/2017-08-partisan-sites-and-facebook-pages
[6]https://github.com/FakeNewsChallenge/fnc-1
[7]https://github.com/selfagency/bs-detector

leading news. The dataset can be found on Kaggle[8].

## 3.2 SemEval 2019 Task 4 - Hyperpartisan News Detection

We start our work with SemEval 2019 Task 4 - Hyperpartisan News Detection[9]. The task is as follows:

> Given a news article text, decide whether it follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person.

### 3.2.1 SemEval 2019 Task 4 Dataset

Our initial modeling is based on SemEval 2019 Task 4 dataset since the size of the dataset is quite large and the articles are labeled by the credibility of the publishers as well as based on the articles.

The dataset is split into two parts - labeled by publishers, and labeled by articles. The distribution of the data is as given in Table 3.2.

Table 3.2: Distribution of articles in the SemEval 2019 Task 4

| Type | By Publisher | By Article |
|---|---|---|
| Training set size | 600,000 | 645 |
| Validation set size | 150,000 | N/A |

The 'by publishers' dataset consists of a total of 750,000 articles divided in a 80/20

[8]https://www.kaggle.com/mrisdal/fake-news/version/1
[9]https://pan.webis.de/semeval19/semeval19-web/

split between training and validation. These articles are labeled by the overall bias of the publisher assigned by journalists at BuzzFeed or MEDIABIASFACTCHECK.COM. 50% of the dataset is *hyperpartisan* while the other half is not. Out of the 375,000 articles which are *hyperpartisan*, half of them(187,500) fall in the left-spectrum of hyperpartisanship while the other half fall in the right-spectrum. Thus, this dataset is quite balanced as opposed to the second part of the dataset labeled by articles.

In the 'by articles' dataset, there are a total of 645 articles labeled through crowd-sourcing on the basis of content in the articles. The labeling of the articles in this group are agreed upon by a consensus. The distribution of hyperpartisanship in this dataset is 238 articles (37%) and the remaining 407 articles (63%) are not hyperpartisan.

### 3.2.2 Midterm Elections 2018 Dataset

The goal of this study is to use the model developed using the preliminary work done on SemEval Task 4 dataset, to comprehend the occurrence of hyerpartisanship in web-searched articles related to U. S. midterm elections in 2018.

We decided to aggregate trending queries related to the query: 'United States midterm election' using Google Trends[10]. A list of 86 queries were curated, less than the initial goal of 100 queries as the queries became repetitive and started bottoming out. The queries can be seen in Table 3.3

Once the queries were curated, BING web-search API[11] was used to scrape web-pages for the queries. A maximum of 100 web-pages were collected for each of the queries, based on the results returned. This resulted in a total of 6616 web articles.

[10]https://trends.google.com/trends/
[11]https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/

Table 3.3: List of queries related to midterm election in 2018

| | |
|---|---|
| united states midterm election | 2018 election results |
| florida midterm election | who won midterm elections |
| midterms results 2018 | midterm election date |
| midterm election map | election day 2018 |
| kavanaugh | us midterm results |
| midterm election ballot 2018 | ted cruz |
| florida midterm election results | pa midterm elections 2018 |
| results of 2018 midterm elections | florida midterm election 2018 |
| who won the midterm elections | texas midterm election results |
| missouri midterm elections | us midterm election results |
| california midterm election results | live midterm election results |
| stacey abrams | texas senate race |
| kansas midterm elections | georgia governor race |
| florida senate race | mid election results 2018 |
| mid term polls | where to vote |
| washington state midterm elections | arizona senate race |
| republicans will win midterms | andrew gillum |
| florida election results | election day |
| election day results 2018 | michigan 2018 election results |
| michigan election results | illinois election results 2018 |
| ca election results | wisconsin election results 2018 |
| wisconsin election results | ca election results 2018 |
| ca election 2018 | colorado election 2018 |

| | |
|---|---|
| colorado election results | ky election results 2018 |
| colorado election results 2018 | nevada election results 2018 |
| georgia governor election results 2018 | election results today |
| maryland election 2018 | minnesota election results 2018 |
| nc election results 2018 | maine election results 2018 |
| maryland election results 2018 | maine election 2018 |
| who won the election | midterm elections 2018 |
| who won the georgia midterm elections | kavanaugh vote |
| kavanaugh hearing | kavanaugh confirmation |
| kavanaugh ford | supreme court |
| christine blasey ford | kavanaugh christine blasey ford |
| beto cruz debate | ted cruz vs beto debate |
| ted cruz vs beto polls | jimmy kimmel vs ted cruz |
| ted cruz mark zuckerberg | us senate election results |
| florida senate race | kemp |
| stacey abrams results | georgia governor race stacey abrams |
| georgia election | did stacey abrams win georgia |
| andrew gillum polls | ron desantis |
| desantis | orange county ca election results 2018 |
| ca secretary of state | sinema |
| us senate makeup | martha mcsally |
| fox news election results | |

In the next chapter, we discuss the features, NLP techniques and the different machine learning algorithms used in the modeling of our data.

# Chapter 4

# Methods and Results

In this chapter, we discuss the various text-features and machine learning algorithms used in the classification of our data to detect hyperpartisanship.

Section 4.1 describes the standard preprocessing techniques done over the text before any feature engineering and/or modeling. The standard process of text mining is shown in the Figure 4.1.

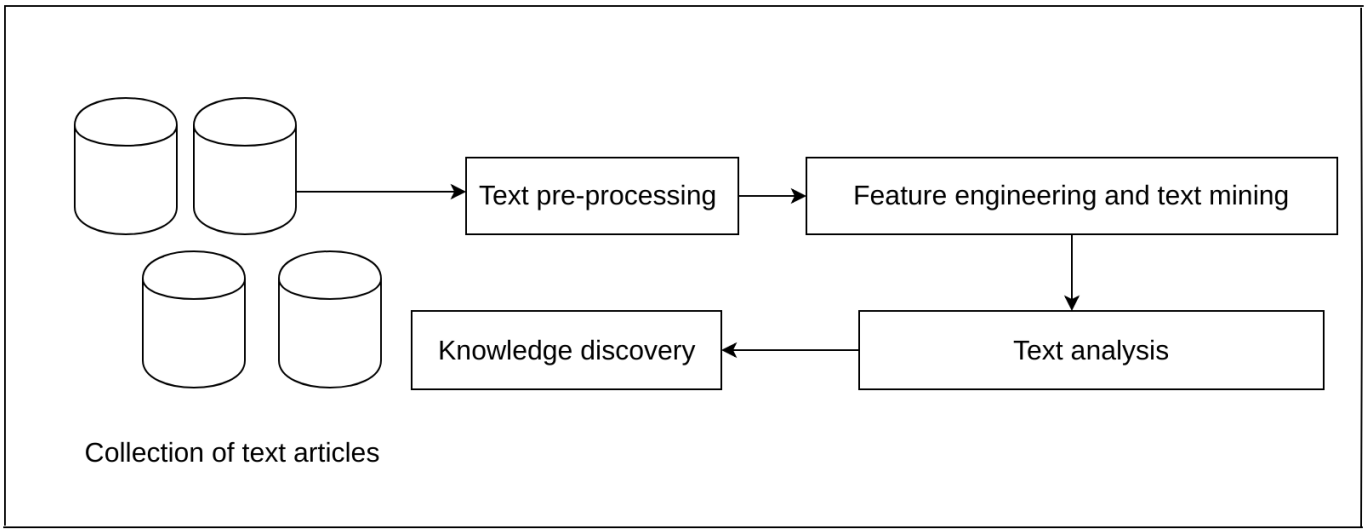



Figure 4.1: Process of text mining

## 4.1 Preprocessing

Preprocessing of text is the preliminary step to convert text into a format feasible for input to an algorithm. The steps involved are explained in the subsequent sub-sections as well as in Figure 4.2.

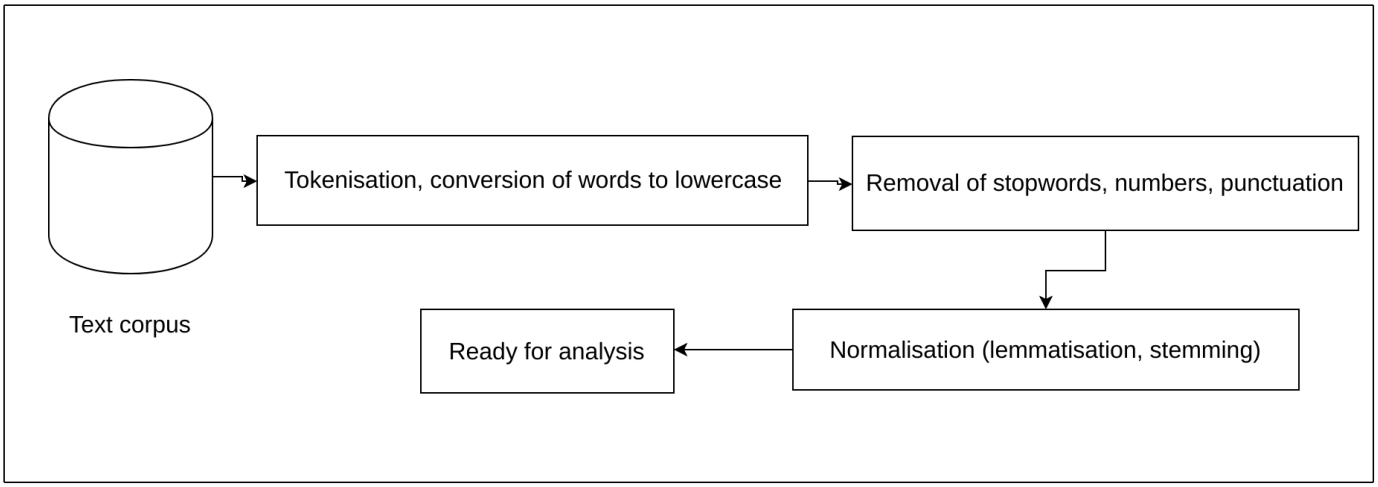



Figure 4.2: Pre-processing of text

### 4.1.1 Tokenization

This step involves splitting paragraphs into sentences and sentences into words. Sentence boundary detection is used to get a list of sentences. We have used *PunktSentenceTokeniser*[1] from NLTK to perform sentence tokenization. It is an implementation of unsupervised multilingual sentence boundary detection by Kiss and Strunk[21]. For word tokenization, white spaces are used as delimiters. Additionally, all the words are either converted to lowercase or uppercase so that capitalized words are not considered different from non-capitalized one. Abbreviations are kept capitalized, and there should be rules

[1]https://www.nltk.org/_modules/nltk/tokenize/punkt.html

implemented to keep them as is, e.g., *US* (United States) and *us* (pronoun).

### 4.1.2 Stop-words Removal

Stopwords are the often-occurring words in a language which connect sentences, and do not hold any importance. We used NLTK's stopwords list [2] to filter out stopwords like '*the*', '*a*', '*so*', etc. from the corpus. Furthermore, we also removed numbers and punctuation since they are not relevant to our analysis.

### 4.1.3 Normalization

Normalization of text refers to reduction of inflectional forms of words into their base forms. While stemming chops off the rear ends of inflections, lemmatization uses lexical knowledge bases to convert them to their root forms. For example, chopping off '*es*' in '*studies*' becomes '*studi*' instead of '*study*' after stemming while it retains the base form '*study*' in lemmatization.
We employed lemmatization in our corpus using WordNet Lemmatizer[3].

## 4.2 Algorithms

In this work, we explore different features which could contribute in automated detection of hyperpartisanship in our dataset, and use the following machine learning algorithms to test our hypotheses. Here, we explain the algorithms used in this work. We have used the standard implementation of the algorithms from scikit-learn[4].

[2]https://www.nltk.org/nltk_data/
[3]https://www.nltk.org/_modules/nltk/stem/wordnet.html
[4]https://scikit-learn.org/stable/

### 4.2.1 Logistic Regression

We use binary logistic regression (LR) to classify the engineered features as hyperpartisan or not. An LR model uses the logistic function to fit the output of a linear equation between 0 and 1. Since the features that we have used in our classification problem are categorical in nature, our LR model transforms the output using the logistic sigmoid function to return a probability value which is used to map to either of the two classes. So, instead of using a line as a decision boundary, we use a sigmoid function that flattens out at 0 and 1 which are our two distinct classes for classification. The logistic function is defined in (4.1) and can be seen in Figure 4.3.

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)} \tag{4.1}$$

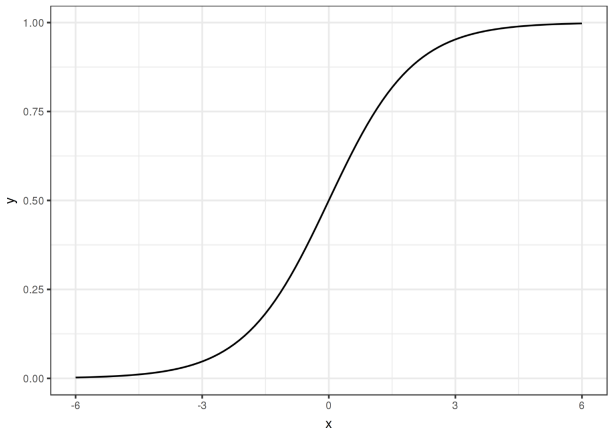



Figure 4.3: Logistic function outputs a value between 0 and 1

We used scikit-learn's logistic regression classifier[5]. In this classification algorithm,

[5]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

the hyperparameter is $C$ which refers to inverse of regularization strength. Regularization is done in order to prevent overfitting of data, and higher value of $C$ corresponds to less regularization. We tested our classifier on a range of values for $C$, and compared the accuracy and time for each different setting.

### 4.2.2 Naive Bayes

The Naive Bayes algorithm assumes that each feature in the dataset makes an equal and independent contribution to the outcome and uses Bayes theorem given in Equation 4.2.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{4.2}$$

Metsis et al. [25] compared the performance of different Naive Bayes versions on text classification (spam filtering). Work by [20, 24, 29] show that Multinomial Naive Bayes works the best with text classification and hence we also carry our implementation using scikit-learn's Multinomial Naive Bayes classifier[6] in addition to Gaussian Naive Bayes[7].

### 4.2.3 Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm which is used in our classification problem to find a hyperplane or a margin which maximizes the nearest points separating the classes from the plane of separation. We have used non-linear kernel: radial basis function as well, to find a non-linear hyperplane separating the classes.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

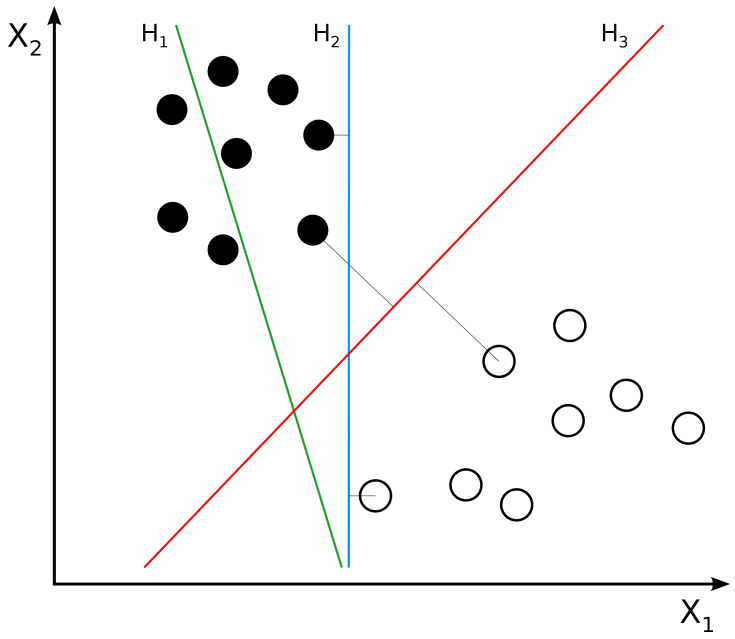



Figure 4.4: Hyperplane in SVM separating the classes [6]

As seen from Figure 4.4, there are three hyperplanes - $H_1$, $H_2$ and $H_3$. While $H_1$ does not separate the classes, $H_2$ does separate with minimal margin and $H_3$ too, but with maximal margin. We used scikit-learn's SVM implementation[8].

### 4.2.4 Decision Trees

We use scikit-learn's default decision tree implementation[9], which uses gini index as the criterion for the split. Decision trees are easier to understand and visualize. The process begins with a root node representing the entire population or the dataset. The classification of instances is done by splitting an instance on its attributes. The criterion of split could be entropy as defined in Equation 4.3 or gini index in Equation 4.4.

[8]https://scikit-learn.org/stable/modules/svm.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

$$E = \sum_{i=1}^{c} -p_i \log p_i \tag{4.3}$$

$$G = 1 - \sum_{j} {p_j}^2 \tag{4.4}$$

where $p$ represents the probability of the presence of an attribute in an instance.

These measures are used to quantify the homogeneity of a split. Higher the information gain (decrease in entropy) or gini index, higher is the homogeneity and thus the split takes place on that attribute.

### 4.2.5 Random Forest

Building upon the previous algorithm, we make use of random forests in our classification problem. In a random forest, each tree has access to a random subset of features or the training data. While making predictions, it takes the majority vote of the decisions made by the individual trees. We have used the hyperparameters *n_ estimators* and *max_ depth* for tuning to find optimum accuracy. This work made use of scikit-learn's implementation[10].

### 4.2.6 Gradient Boosting

This section refers to the implementation of gradient boosting called XGBoost by Chen and Guestrin[12] or eXtreme gradient boosting. This algorithm has been known to fare well in many machine learning challenges due to its scalability and speed [12]. They use a sparsity-aware algorithm to find the split in tree-learning and block structure to support

[10]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

parallelization. We use the default implementation of scikit-learn's XGBClassifier[11].

### 4.2.7 Stochastic Gradient Classifier

Since we are dealing with huge number of instances, using stochastic gradient descent is useful. The updates in coefficients is performed for each training instance and not at the end of the batch of instances.

## 4.3 Feature Engineering

Our study is based on exploring how the content of an article, mainly the title and the body, could be used as predictive features for the detection of partisanship in an article. We explore if there are linguistic and style differences between mainstream articles and articles with a partisan inclination. We divide the following sections into exploring the features of headlines and the body of articles separately. Our metric of evaluation in all cases is accuracy.

## 4.4 Headlines of articles

As discussed in Chapter 2, headlines of news articles can be an important feature in our case since these articles are created on the foundation of being clickbaits. We applied Term Frequency-Inverse Document Frequency (TF-IDF) on the titles of the news articles and used the classification algorithms discussed above.

[11]https://github.com/dmlc/xgboost/blob/master/python-package/xgboost/sklearn.py

### 4.4.1 Term Frequency-Inverse Document Frequency (TF-IDF)

We use TD-IDF as a weighted statistical measure to find the importance of a word in a document in a corpus. The importance of a word increases proportionally by the number of times it appears in the document, but is offset by its frequency in the corpus. The TF-IDF score is found out by the multiplication of the following two terms:

- **Term Frequency.** It measures the frequency of the occurrence of terms in a document. Since every document differs in length, a term might appear more in longer documents than the shorter ones. To normalize this metric, the term frequency is divided by the total number of terms in the document or the document length.
  **Term Frequency** $= \frac{No.\ of\ terms\ 't'\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$
  (All the terms in a document are considered important here.)

- **Inverse Document Frequency.** This is a measure of how important a term is, in the whole corpus (collection of documents). Since words like "the", "is", and "of" are used very often, and do not hold much importance, it becomes necessary to scale up the rare terms and weigh down the frequent terms.
  Thus, **Inverse Document Frequency** $= \ln \frac{Total\ number\ of\ documents}{No.\ of\ documents\ with\ term\ 't'\ in\ it}$

  Combining these two,
  $\mathbf{TF-IDF(t, d, D)} = TF(t, d) \cdot IDF(t, D)$
  where $t$ is the term, $d$ is the document, and $D$ is the corpus.

We used scikit-learn's TfidfVectorizer[12] instantiated using stop words since the headlines are smaller in length, and in non-clickbait articles, inference of stop-words is left to the

[12]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

readers. The hyperparameter '*min_df*' is used to ignore words strictly below the threshold to be considered while building the vocabulary. We tuned it for maximum accuracy for each algorithm, results of which can be found in the Appendix section described below.

We used each of the algorithms as described in Section 4.2 after building the TF-IDF vector. We now describe the parameter tuning done for the algorithms, if any.

### Logistic Regression

The result can be seen in Figure 4.5.

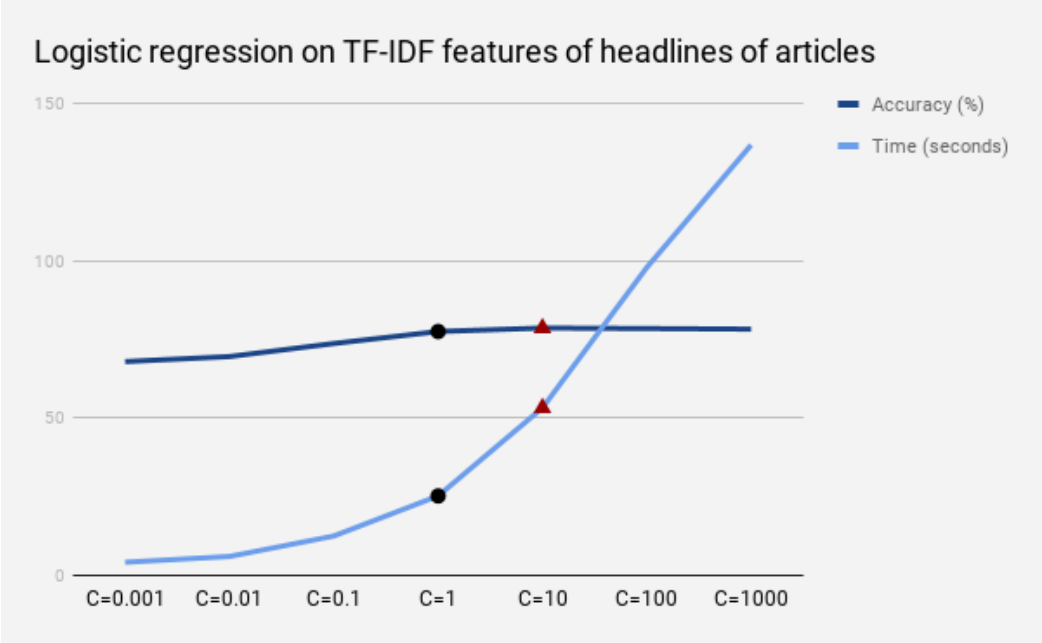



Figure 4.5: Accuracy and timing of different values of hyperparameter $C$ in Logistic regression classifier for TF-IDF on headlines of articles

As seen from Figure 4.5, accuracy at $C$=1 increases and thereafter it does increase, but at a lower pace. Whereas the time required to fit model increases considerably for $C$=10, 100, 1000. Thus, considering the time and accuracy trade-off, we choose $C$=1.0

as our inverse regularization factor. The timing and accuracy data can be found in Section A.1.

### Decision Tree, Naive Bayes

Our dataset is quite huge consisting of 700K articles, and thus fitting the whole vocabulary into memory was not feasible. We tried to reduce the vocabulary using '*min_df*' parameter in TF-IDF and worked it on a range of values, and selected 0.0001 as the cut-off. The cut-off values can be found in Section A.2 and Section A.3 for these algorithms.

### Random Forest

We worked on a range of values for *n_estimators* and *max_depth* for random forest and the results can be seen in Section A.4.

### Final Results

The final results of using TF-IDF vectors of headlines of articles can be seen in Table 4.1 and Figure 4.6 below. In the results of all our experiments, the 'Time' column refers to the training time of the algorithms.

Table 4.1: Test accuracy for all algorithms with optimum hyperparameter using TF-IDF

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 77.49 | 25.25 |
| SVM (*Linear* kernel) | 79 | 142323.33 |
| Decision Tree ($min\_df = 0.0001$) | 71.10 | 789.29 |
| Gaussian Naive Bayes ($min\_df = 0.0001$) | 74.30 | 27.00 |
| Multinomial Naive Bayes ($min\_df = 0.0001$) | 73.62 | 33.62 |
| Random Forest | 74.34 | 878.34 |
| XGBoost | 66.55 | 391.89 |
| Stochastic Gradient Descent | **83.23** | 3064.4 |

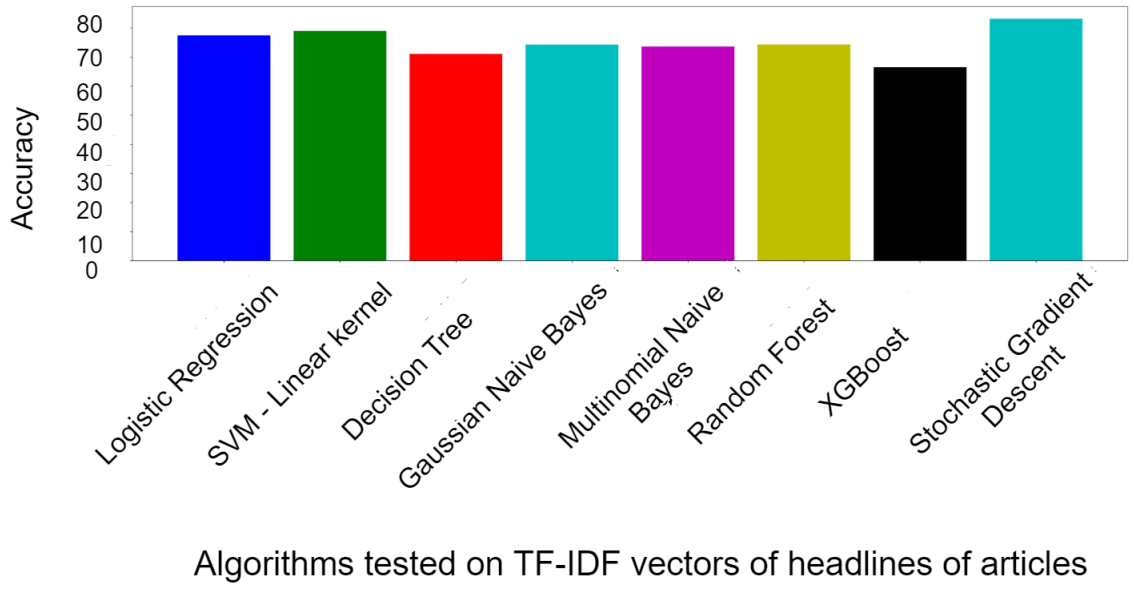



Figure 4.6: Accuracy of the TF-IDF vectors on the headlines

### 4.4.2 Length of the headline

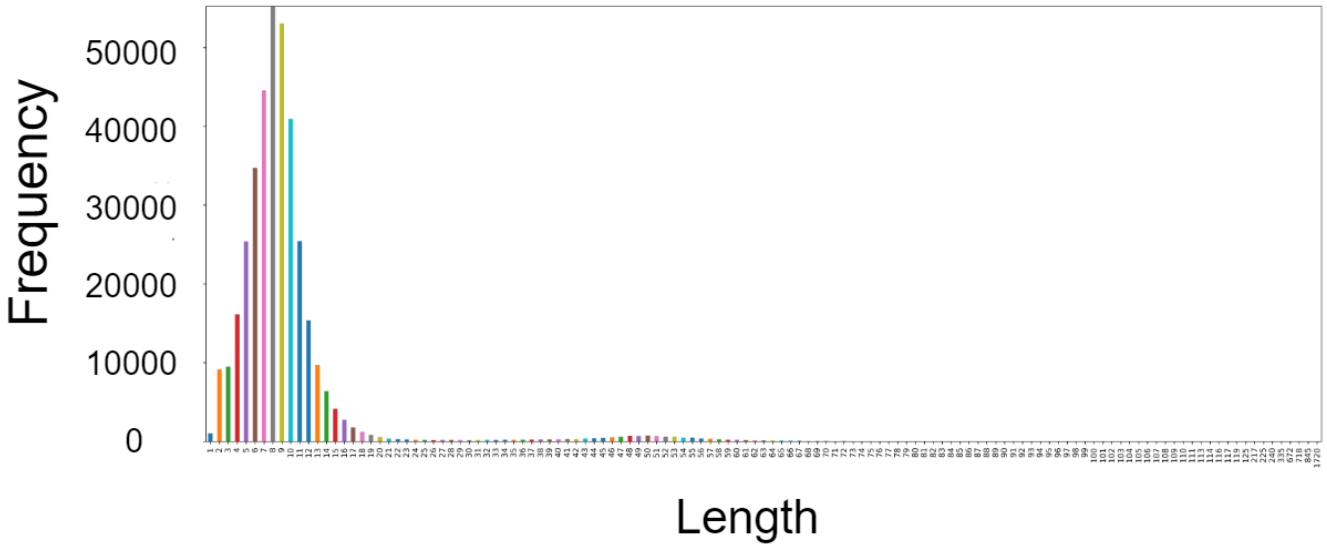
Chakraborty et al. [11] explained in their paper that headlines of clickbait articles tend to be longer than the mainstream articles, since they contain more function words, and are expected to gauge readers' attention. Thus, we use the length of headlines to model the algorithms.

From Figure 4.7, it can be seen that in both the classes, the length of a headline is primarily concentrated in the range of [6-9] words. The Figure 4.8 shows the class distribution in the whole training dataset. The results of the modeling the algorithms using length of headlines can be seen in Table 4.2 and in Figure 4.9. The parameter tuning results can be found in Section B.1.

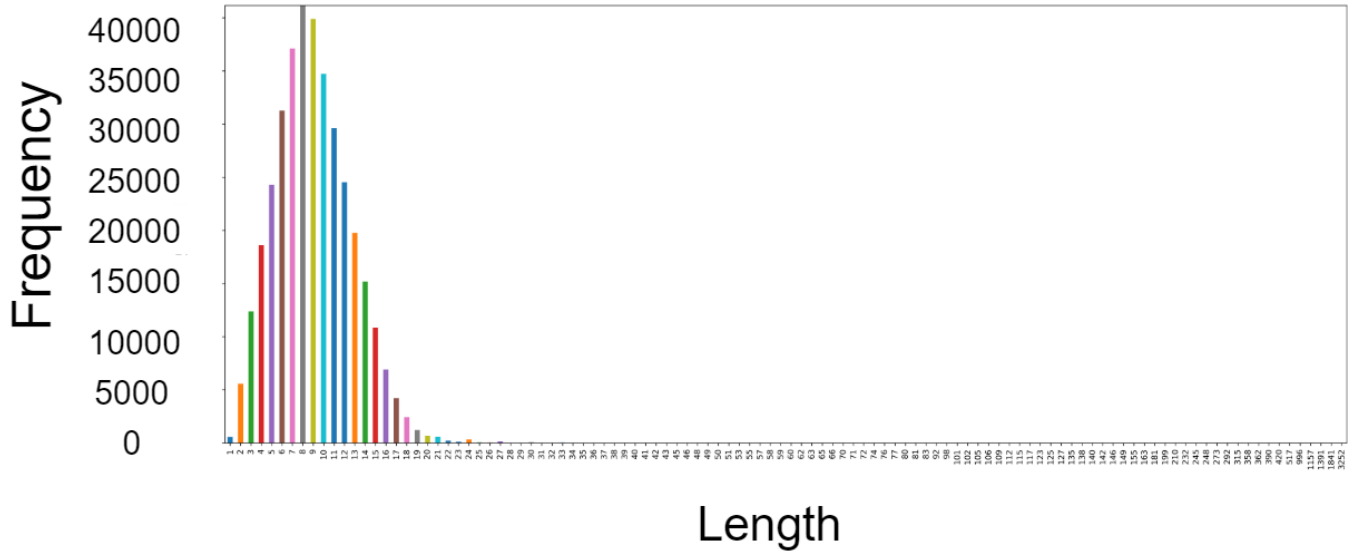
As we see from the results in Table 4.2, length of headlines are not much of a predictive feature, due to majority of the articles from both the classes falling in the same range of length.

Table 4.2: Test accuracy for all algorithms with optimum hyperparameter using length of headlines

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 45.62 | 1.00 |
| SVM (*Linear* kernel) | **74** | 11752.31 |
| SVM (*Radial basis* kernel) | 59 | 16395.32 |
| Decision Tree | 57.58 | 0.20 |
| Gaussian Naive Bayes | 48.25 | 0.09 |
| Multinomial Naive Bayes | 49.99 | 574.65 |
| Random Forest | 57.82 | 157.04 |
| XGBoost | 57.82 | 7.99 |
| Stochastic Gradient Descent | 49.99 | 0.35 |




(a) Frequency distribution of length of headlines in the non-hyperpartisan articles




(b) Frequency distribution of length of headlines in the hyperpartisan articles

Figure 4.7: Class frequency distribution of length of headlines of articles

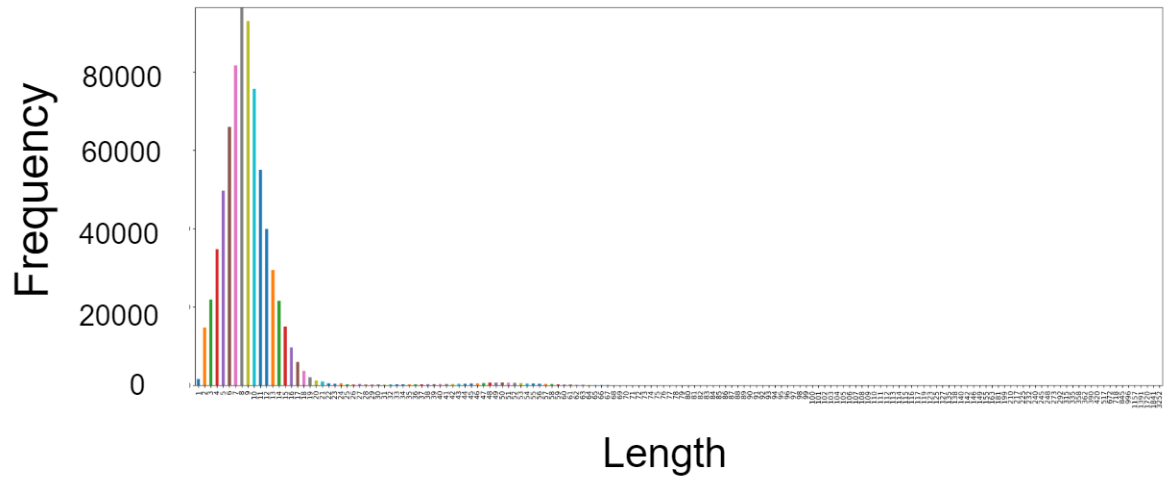



Figure 4.8: Class frequency distribution of length of headlines of all the news articles in the training dataset

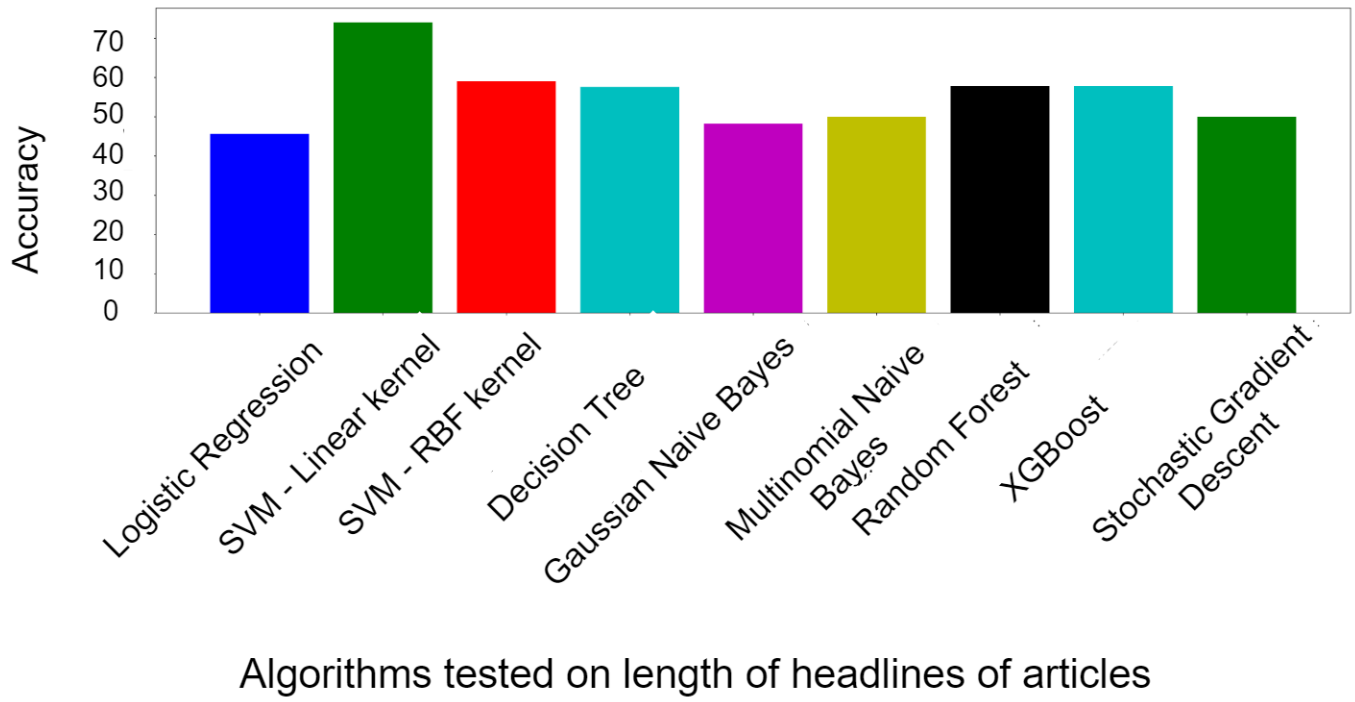



Figure 4.9: Accuracy of using the length of the headlines as features

### 4.4.3 Ratio of capitalized and stop words to total words

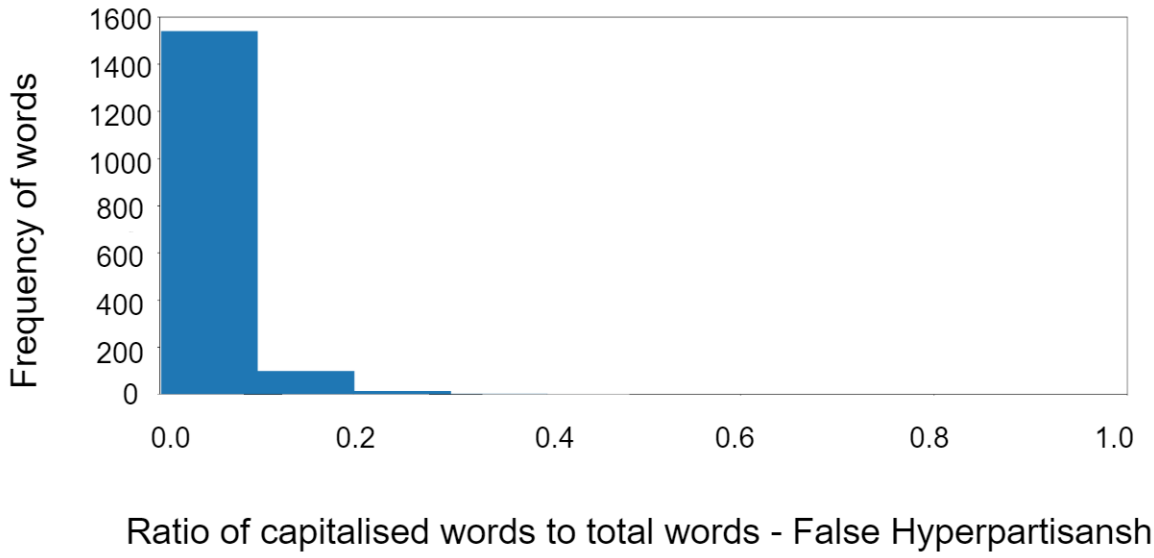
In this subsection, we explore the writing style of headlines, i.e., usage of capitalized words and stop words. Captivating titles tend to have more punctuation and uppercase words. Such titles have fully formed sentences rather than only using content words and leaving their interpretation to the readers [11]. We calculate the ratio of both these factors and model our algorithms using these features.

Figure 4.10 shows the frequency distribution of ratio of capitalized words to total words in the headline. The bulk of distribution is in the range of [0.0-0.1] for both classes. The ratio was encoded and the results of the classification can be seen in Table 4.3. Thee results of both these factors can be compared in Figure 4.12.

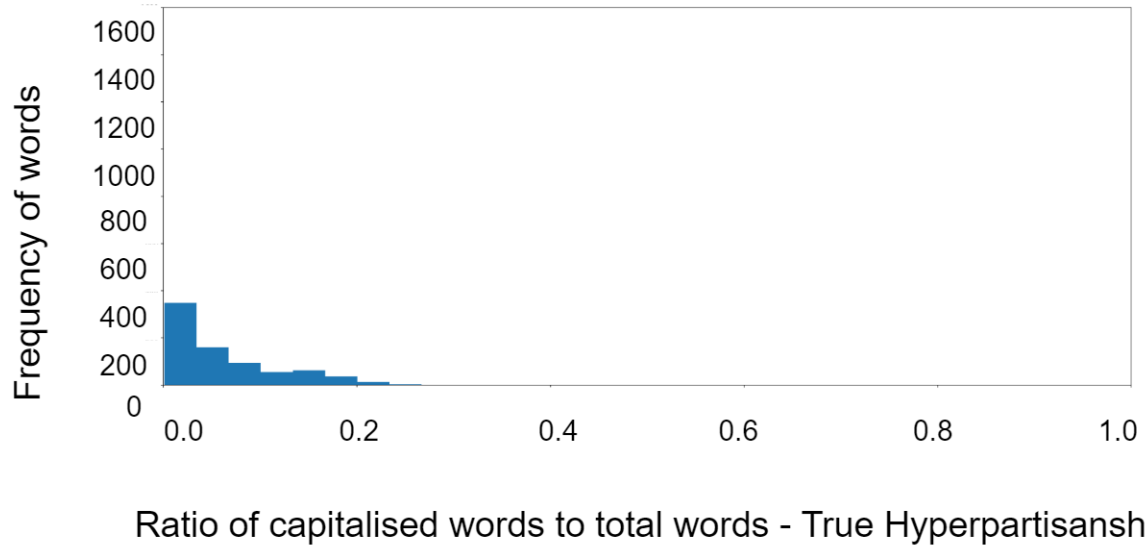
Table 4.3: Test accuracy for all algorithms with optimum hyperparameter for ratio of capitalized words to total words in the headline of the article

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 49.06 | 0.65 |
| SVM (*Linear* kernel) | **58** | 5439.24 |
| SVM (*Radial basis* kernel) | 52 | 14548.12 |
| Decision Tree | 51.35 | 0.13 |
| Gaussian Naive Bayes | 49.82 | 0.16 |
| Multinomial Naive Bayes | 49.88 | 30.89 |
| Random Forest | 51.42 | 157.04 |
| XGBoost | 51.35 | 7.31 |
| Stochastic Gradient Descent | 49.88 | 0.34 |

We perform a similar analysis using stop words and the frequency distribution can be seen in Figure 4.11. The results of the classification can be seen in Table 4.4.

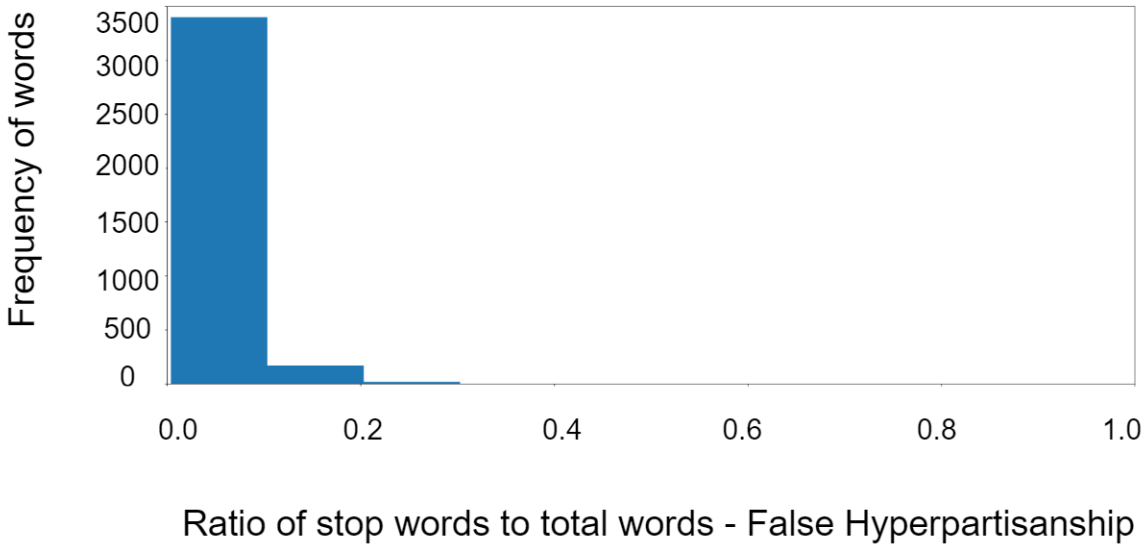



(a) Frequency distribution of ratio of capitalized words to total words in non-hyperpartisan articles

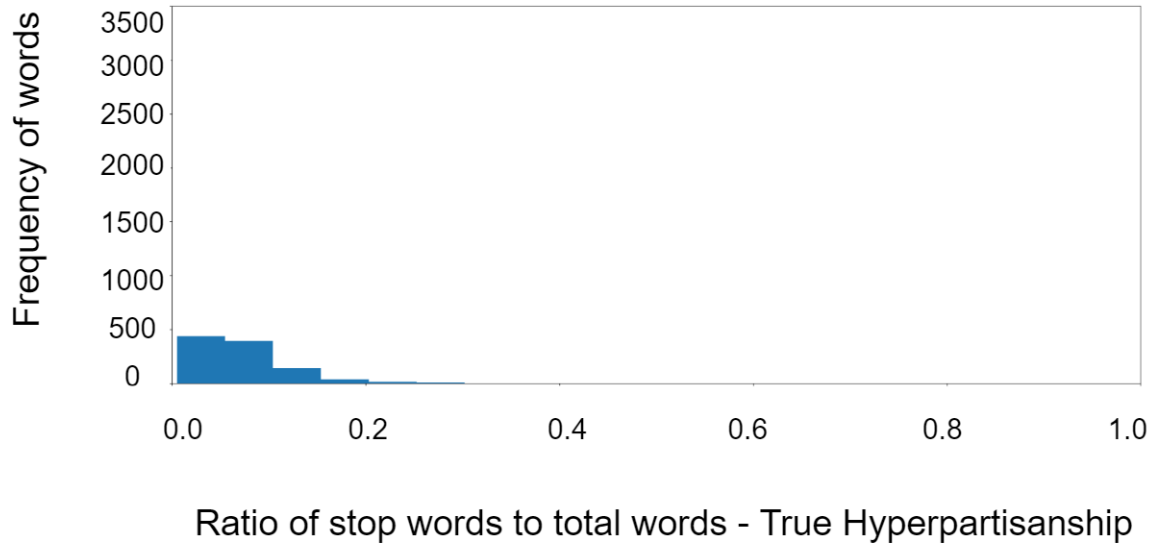



(b) Frequency distribution of ratio of capitalized words to total words in hyperpartisan articles

Figure 4.10: Frequency distribution of capitalized words to total words in the headlines of the articles for both classes




(a) Frequency distribution of ratio of stop words to total words in non-hyperpartisan articles




(b) Frequency distribution of ratio of stop words to total words in hyperpartisan articles

Figure 4.11: Frequency distribution of stop words to total words in the headlines of the articles for both classes.

Table 4.4: Test accuracy for all algorithms with optimum hyperparameter for ratio of stop words to total words in the headline of the article

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 56.17 | 0.53 |
| SVM (*Linear* kernel) | **58** | 5439.24 |
| SVM (*Radial basis* kernel) | 52 | 14548.12 |
| Decision Tree | 56.23 | 0.16 |
| Gaussian Naive Bayes | 54.92 | 0.06 |
| Multinomial Naive Bayes | 49.86 | 0.17 |
| Random Forest | 55.85 | 146.91 |
| XGBoost | 56.03 | 7.27 |
| Stochastic Gradient Descent | 54.92 | 0.3 |

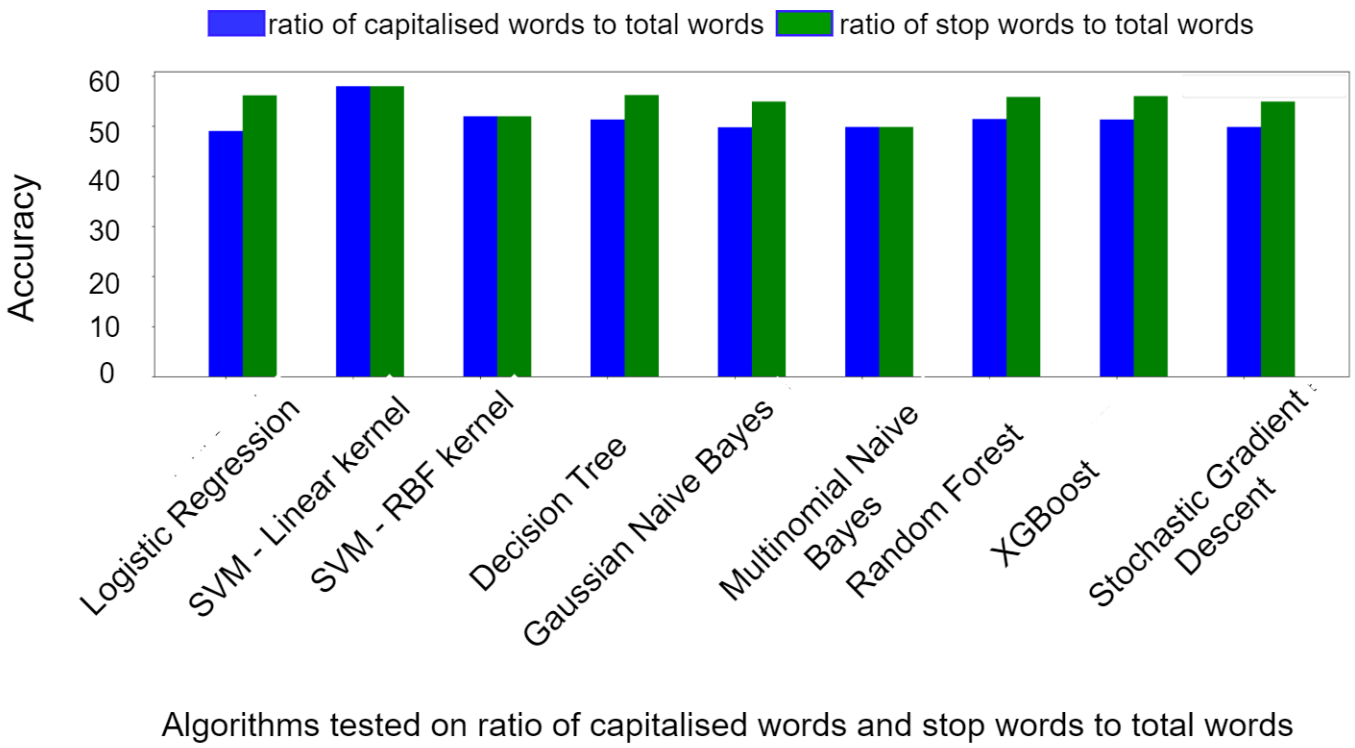



Figure 4.12: Accuracy of using the ratio of capitalized words and stop words of total words of length of the headlines as features

### 4.4.4 Using Content Words - Parts of Speech analysis

Morphology of text could be divided into two parts - the *function* words and the *content* words. Content words are words that have meaning and belong to the parts of speech, particularly nouns, adjectives, adverbs and verbs. Function words are used to provide structural and grammatical relationship into which the content words might fit. These words belong to the parts of speech: determiner, conjunction and preposition.

#### Adjectives

In this section, we explore how using parts of speech in a text, specifically the content words, can be used to find out partisanship in an article. We consider the idea of *subjectivity tagging* [36]. Subjectivity tagging distinguishes opinions from factual information, mainly the idea of distinguishing hyperpartisan articles from the mainstream articles. There have been previous studies which show that there is a statistically significant and a positive correlation between presence of adjectives and subjectivity [37]. We extend this idea to find out if subjectivity can be used to distinguish partisanship from mainstream in text articles.

We used NLTK's pos_tagger[13] to find parts of speech in the title of the articles. Adjectives are represented by the tag 'JJ' according to the Penn treebank[14]. The article titles in the training set were traversed to create a dictionary with all the adjectives as well as to keep their count. The top 80% of the adjectives (745) were used as features to classify our dataset.

[13]https://www.nltk.org/_modules/nltk/tag.html
[14]https://www.clips.uantwerpen.be/pages/mbsp-tags


Figure 4.13: Wordcloud of adjectives in the training set

As seen in Figure 4.13, higher the occurrence of a word, bigger is the word represented in the wordcloud. This technique is independent of the dataset, since the adjectives covered are mostly used in article representations, and are not limited to a particular niche. And thus this method can be extended to more unseen articles for analysis. The result of this analysis can be seen in Table 4.5.

Table 4.5: Test accuracy using the top 80% adjectives as features

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | **71.1** | 109.87 |
| Decision Tree | 67.5 | 14518.27 |
| Gaussian Naive Bayes | 58.31 | 194.57 |
| Multinomial Naive Bayes | 69.1 | 40.26 |
| Random Forest | 67.11 | 14592.91 |
| XGBoost | 62.69 | 11103.74 |
| Stochastic Gradient Descent | 70.44 | 202.64 |

**Nouns**

We continue our experiment using another part of speech, i.e., nouns. Nouns represent people, places and/or things. Using NLTK's pos_tagger, we segregate the nouns (NN*) consisting of NN (noun, singular or mass), NNS (noun, plural), NNP (noun, proper singular) and NNPS (noun, proper plural).

The wordcloud of the nouns can be seen in Figure 4.14.

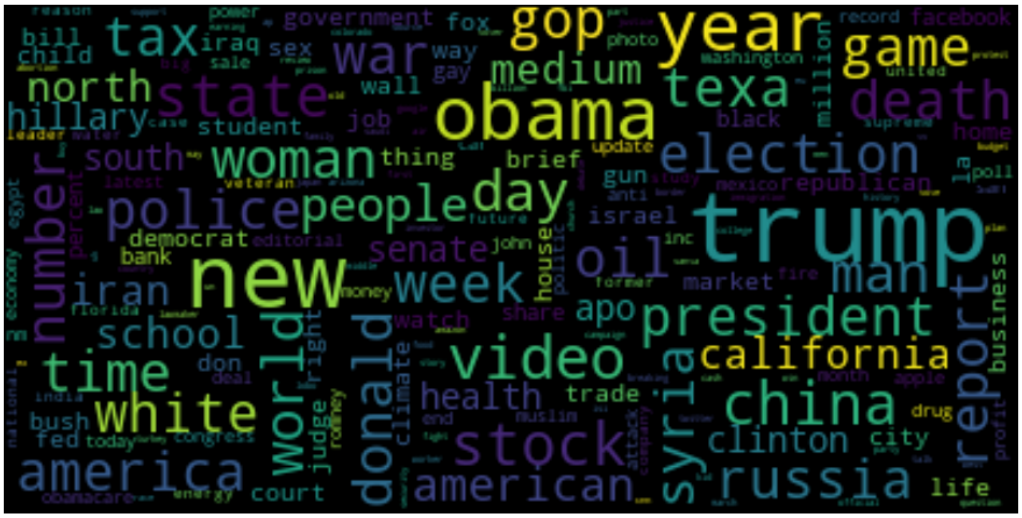



Figure 4.14: Wordcloud of nouns in the training set

The results of classification can be seen in Table 4.6.

Table 4.6: Test accuracy using the top 80% nouns as features

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | **72.55** | 40.9 |
| Decision Tree | 68.87 | 5778.43 |
| Gaussian Naive Bayes | 67.23 | 76.56 |
| Multinomial Naive Bayes | 71.33 | 9.69 |
| Random Forest | 68.23 | 9812.23 |
| XGBoost | 63.71 | 3044.02 |
| Stochastic Gradient Descent | 72.27 | 47.14 |

The final results using parts of speech tags, namely adjectives and nouns, can be visualized in Figure 4.15.

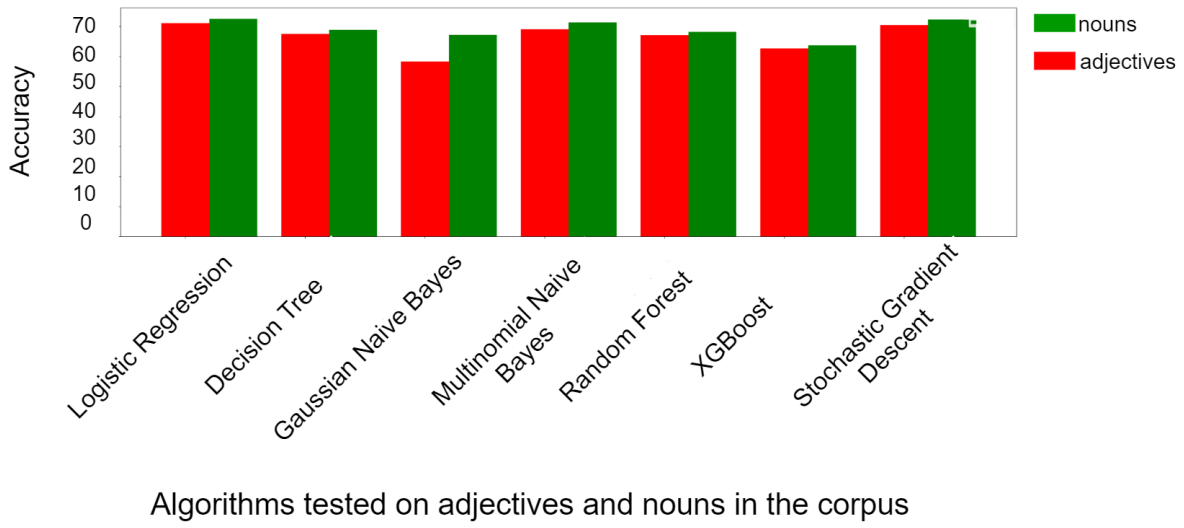



Figure 4.15: Accuracy of using 80% most occurring adjectives and nouns for training

As seen, this method yields better results compared to others (73%).

## 4.5 Publishers

The mainstream media is distorted with biased and hyperpartisan content for financial and political benefits. Thus, publishing houses or the publishers are more of a source for constant production of such articles, and in this section we analyze the different URLs of the news articles to check for their contribution.

Our analysis showed that a total of 256 and 318 unique websites contributed to the two classes. Their intersection resulted in only 27 websites. Top 20 websites of non-hyperpartisan class contributed to 94% of the total 350K articles, while top 20% of the hyperpartisan class contributed to 72% of the remaining 350K articles. There was no intersection between the top 20 websites in either classes. The common websites in the

two classes and their frequency distribution can be seen below in Figure 4.16. The class frequency distribution can be seen in Figure 4.17. These common websites are mentioned in Table 4.8.

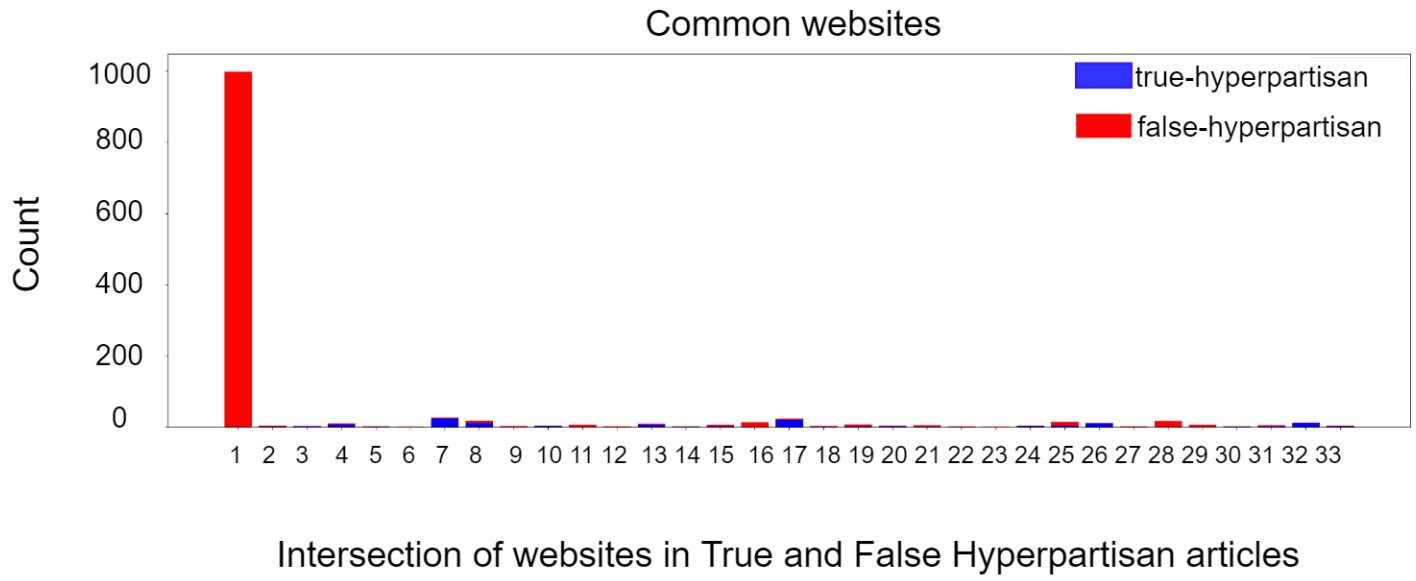
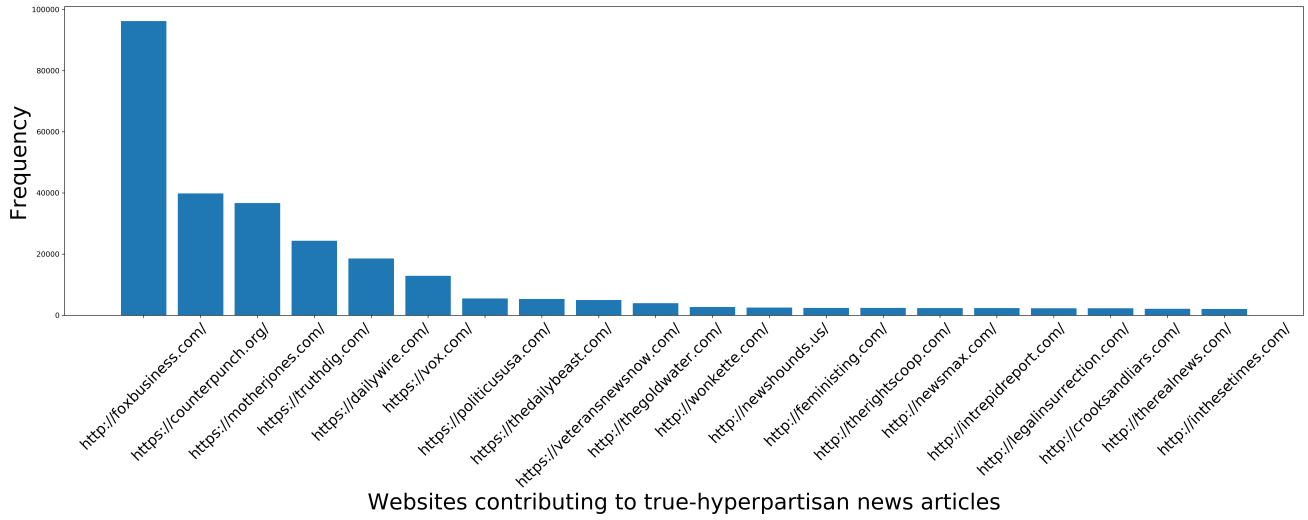



Figure 4.16: Websites common to both classes and their contribution to each class

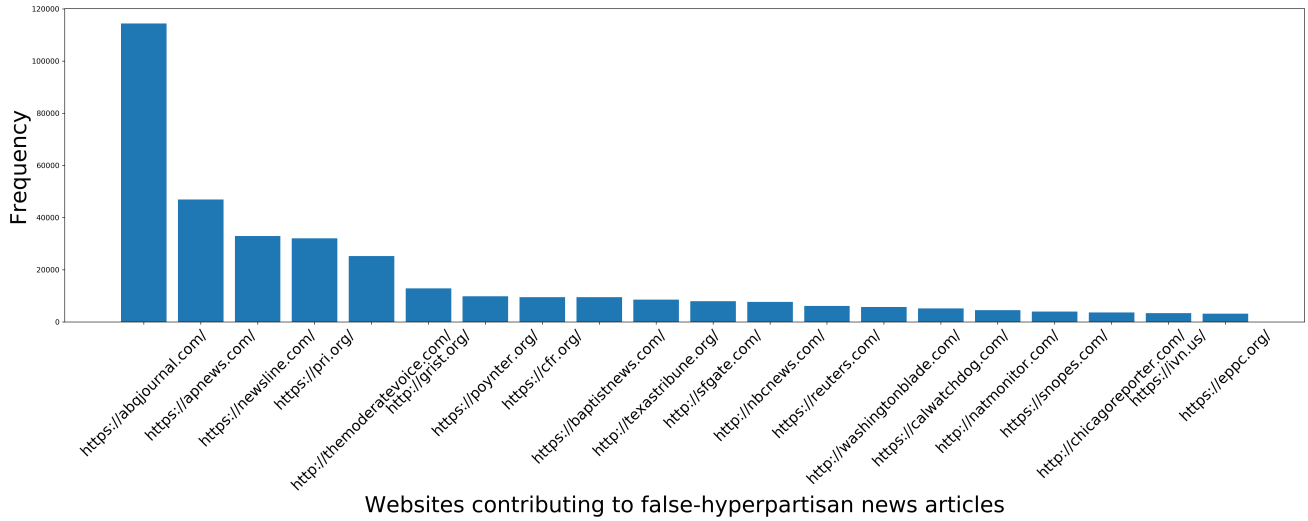
The results of classification using the websites can be seen in Table 4.7. Although this method is not practical since we need to have a look-up of all the websites and their preference to being in either one of the classes. However, their writing styles can be studied as sources of hyperpartisan articles.

## 4.6 Body of articles

In this section we explore how the writing style of the body of an article can be of help in automated classification using vocabulary features. We make use of n-grams, type token ratio, readability scores and presence of dominant parts of speech words.




(a) Frequency distribution of the top 20 of hyperpartisan websites




(b) Frequency distribution of the top 20 of non-hyperpartisan websites

Figure 4.17: Class frequency distribution of top 20 of the websites in each class

Table 4.8: List of websites common to both True and False Hyperpartisan articles

| | |
|---|---|
| 1. http://elitedaily.com/ | 2. http://heavy.com/ |
| 3. http://hollywoodlife.com/ | 4. http://ijr.com/ |

| | |
|---|---|
| 5. http://insider.foxnews.com/ | 6. http://nypost.com/ |
| 7. http://people.com/ | 8. http://theweek.com/ |
| 9. http://truepundit.com/ | 10. http://turtleboysports.com/ |
| 11. http://www.bizpacreview.com/ | 12. http://www.brandenton.com/ |
| 13. http://www.cbsnews.com/ | 14. http://www.complex.com/ |
| 15. http://www.dcclotheslines.com/ | 16. http://www.express.co.uk/ |
| 17. http://www.foxbusiness.com/ | 18. http://www.foxnews.com/ |
| 19. http://www.glamour.com/ | 20. http://www.gq.com/ |
| 21. http://www.idahostatesman.com/ | 22. http://www.nytimes.com/ |
| 23. http://www.rollingstone.com/ | 24. http://www.snopes.com/ |
| 25. http://www.thegatewayundit.com/ | 26. http://www.trueactivist.com/ |
| 27. http://www.washingtonexaminer.com/ | 28. http://bearingarms.com/ |
| 29. http://bossip.com/ | 30. http://goodmenproject.com/ |
| 31. http://www.circa.com/ | 32. http://www.rawstory.com/ |
| 33. http://www.realclearpolitics.com/ | |

### 4.6.1 Type Token Ratio (TTR)

This vocabulary feature signifies the lexical diversity of a block of text. Tokens in a text refers to the total number of words, whereas type represents the number of unique words. Thus, TTR can formulated as shown in Equation 4.5.

$$\text{TTR} = \frac{Type}{Tokens} = \frac{Total\ unique\ words}{Total\ words\ in\ the\ text} \tag{4.5}$$

Table 4.7: Test accuracy for all algorithms with optimum hyperparameter for URL

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 67.25 | 1.4948725700378418 |
| Decision Tree | 99.98 | 0.54 |
| Gaussian Naive Bayes | 66.91 | 0.08 |
| Multinomial Naive Bayes | 49.83 | 48.57 |
| Random Forest | 99.98 | 351.04 |
| XGBoost | 98.75 | 8.76 |
| Stochastic Gradient Descent | 67.25 | 0.39 |

Richards [27] explains that TTR is mainly used in research for child language. Since this ratio signifies *lexical variety*, it can be used in our dataset to see if text has been written more fluidly in partisan articles compared to the ones without any bias, which is our hypotheses. Since the length of the text can vary among different articles, we employed TTR on a window of 100 words in each article to maintain uniformity.
Random forest works the best among other classifiers (53%), the results of which can be found in Section C.2.

### 4.6.2 Readability scores

Readability scores are designed to perceive how difficult a passage in the English language is to comprehend. There are many tests which use factors like word length, sentence length, and number of syllables to result a score. We used the following different tests, out of which 'Automated Readability Index' (ARI) yielded the best results, which can be found in Table 4.9.

- The Flesch Reading Ease Score formula[15]
- The Flesch-Kincaid Grade Score Level[16]

[15]https://en.wikipedia.org/wiki/Flesch–Kincaid_readability_testsFlesch_reading_ease
[16]https://en.wikipedia.org/wiki/Flesch–Kincaid_readability_testsFlesch–Kincaid_grade_level

- The Fog Scale (Gunning FOG Formula)[17]
- The SMOG Index[18]
- Automated Readability Index[19]
- The Coleman-Liau Index[20]
- Linsear Write Formula[21]
- Dale-Chall Readability Score[22]

Table 4.9: Test accuracy using Automated Readability Index (ARI) on the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression ($C = 1.0$) | 57.16 | 1.41 |
| Decision Tree | 57.16 | 0.12 |
| Gaussian Naive Bayes | 56.77 | 0.05 |
| XGBoost | **57.16** | 15.19 |
| Random Forest | 57.16 | 112.82 |
| Stochastic Gradient Descent | 55.83 | 0.29 |

We used textstat[23] library for these tests.

[17]https://en.wikipedia.org/wiki/Gunning_fog_index
[18]https://en.wikipedia.org/wiki/SMOG
[19]https://en.wikipedia.org/wiki/Automated_readability_index
[20]https://en.wikipedia.org/wiki/Coleman–Liau_index
[21]https://en.wikipedia.org/wiki/Linsear_Write
[22]https://en.wikipedia.org/wiki/Dale–Chall_readability_formula
[23]https://pypi.org/project/textstat/

### 4.6.3 n-grams

An n-gram is an contiguous block of words of length $n$ in a sentence. The value for $n$ can start from 1 implying unigrams and the word itself, to any number. Large values of $n$ are not considered since larger n-grams might rarely occur together in other text articles and lead to the problem of data-sparsity. We varied $n$ from 1 to 3, i.e., unigrams, bigrams and trigrams. We used a TF-IDF implementation of n-grams ranging from 1 to 3, excluding stopwords, and using the lemmatized form of the word. This method yielded a good accuracy of 90.99% using logistic regression. This method cannot be practically deployed to unseen texts since n-grams would vary depending on the topic.

### 4.6.4 Parts Of Speech

In this implementation, we employ parts of speech tags over the body of the articles, namely extracting the adjectives (JJ*) and nouns (NN*). In this section, we further use a combination of both nouns and adjectives to classify our data and see the result.

#### Nouns

There were a total of 316082 nouns in the training corpus out of which we retained the top 80% most occurring 2485 nouns for our analysis. The wordcloud can be visualized in Figure 4.18.

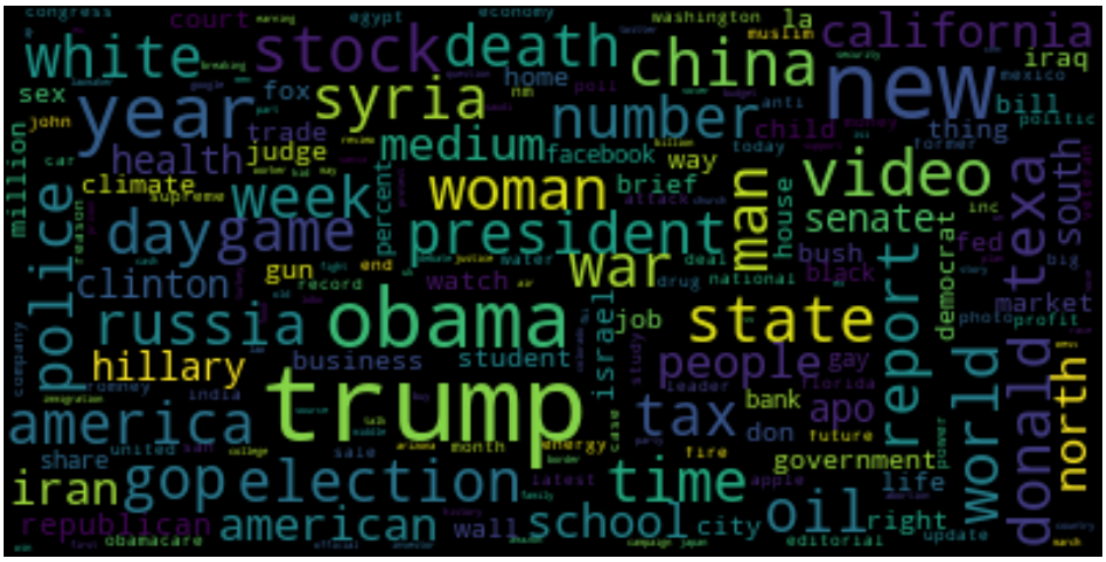



Figure 4.18: Wordcloud of the nouns according to their density in the body of training corpus

**Adjectives**

The total number adjectives found in the training corpus was about 187430. We used the adjectives which contributed to top 80% of their occurrences. This brought down the number of features to a total of 2273 adjectives. The adjectives can be visualized in Figure 4.19

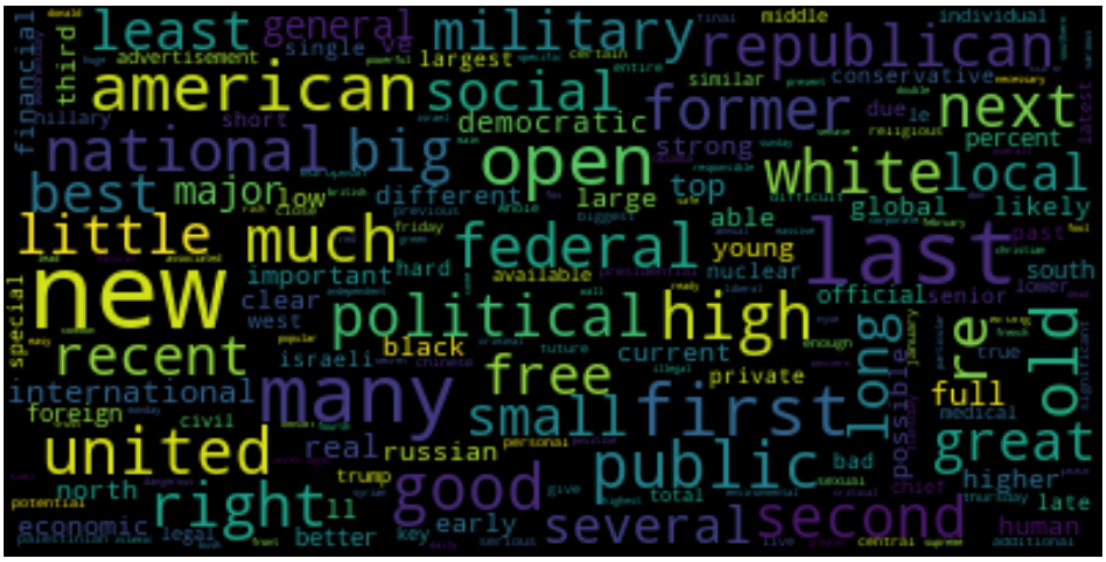



Figure 4.19: Wordcloud of the adjectives according to their density in the body of training corpus

We used a combination of nouns and adjectives as well. The results of this analysis can be found in Appendix section D and can be visualized below in Figure 4.20.

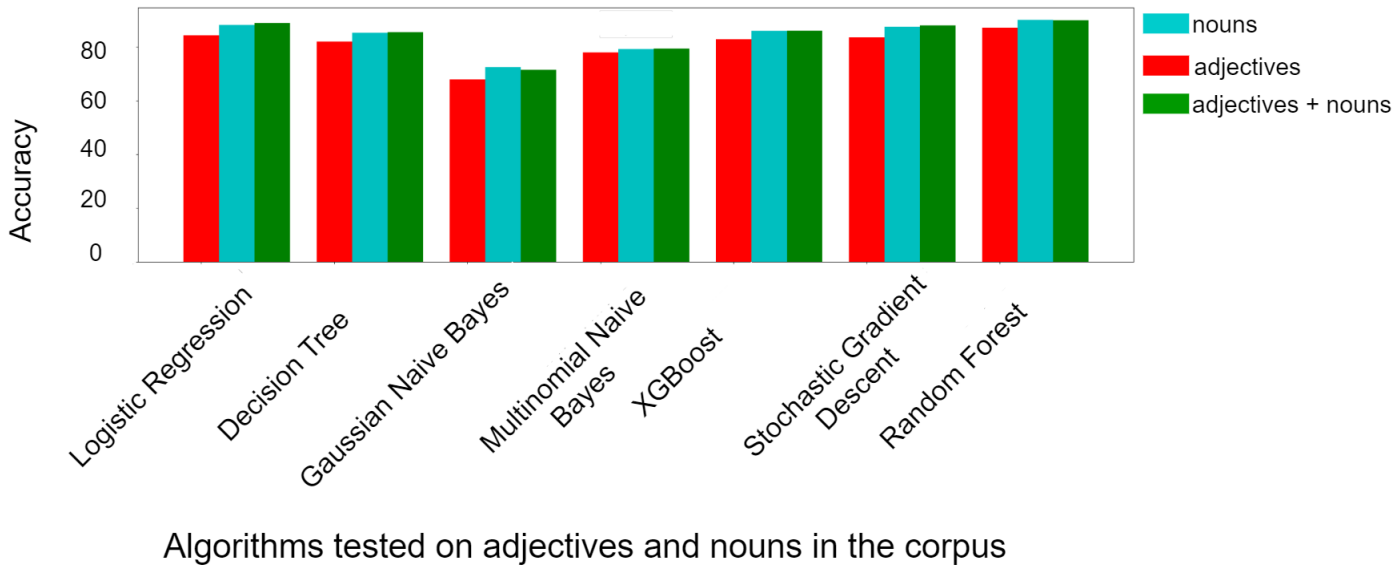



Figure 4.20: Accuracy using adjectives, nouns and both nouns and adjectives

As seen, the combination of both nouns and adjectives performed well in almost all algorithms. This method has yielded better results than the rest of the methods.

### Sentiment Analysis

The context in which a word is used leads us to a better understanding of the stance rather than just the presence. For example, as in the previous section, we studied how presence of certain nouns and adjectives could tell us about the subjectivity or the opinion in a text. However, if an adjective is preceded by a '*not*', the presence of that particular adjective does not hold any importance.

Thus, in this subsection, we evaluate how exploring the sentiments surrounding the context of the usage of a word could be used for classification.

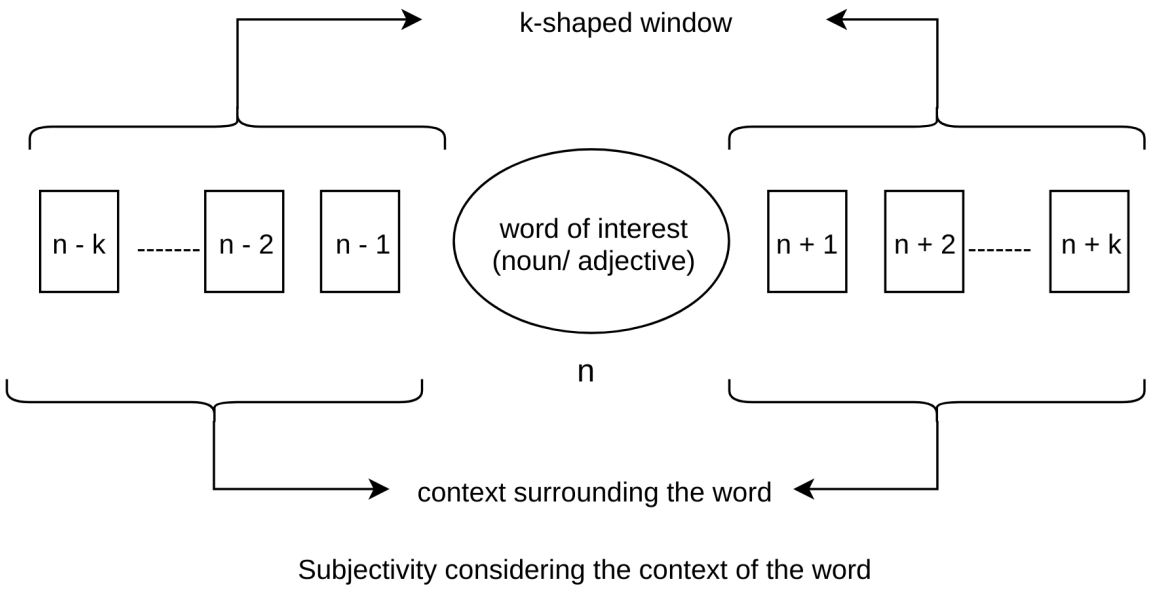



Figure 4.21: Sentiment analysis over the context of the word

As seen in Figure 4.21, we form a $k$-shaped window around our word of interest which are nouns and adjectives and see if the sentiments of the context can be used as good predictive features. We used TextBlob library[24] to find the sentiment polarity of a block of text. The polarity could be negative, neutral or positive and is a float score represented in a range of [-1.0, 1.0].

The results for this can be seen in Appendix section D and can be visualized in Figure 4.22. Using sentiments of both nouns and adjectives on random forest yielded the best accuracy of 78.13%.

[24]https://textblob.readthedocs.io/en/dev/

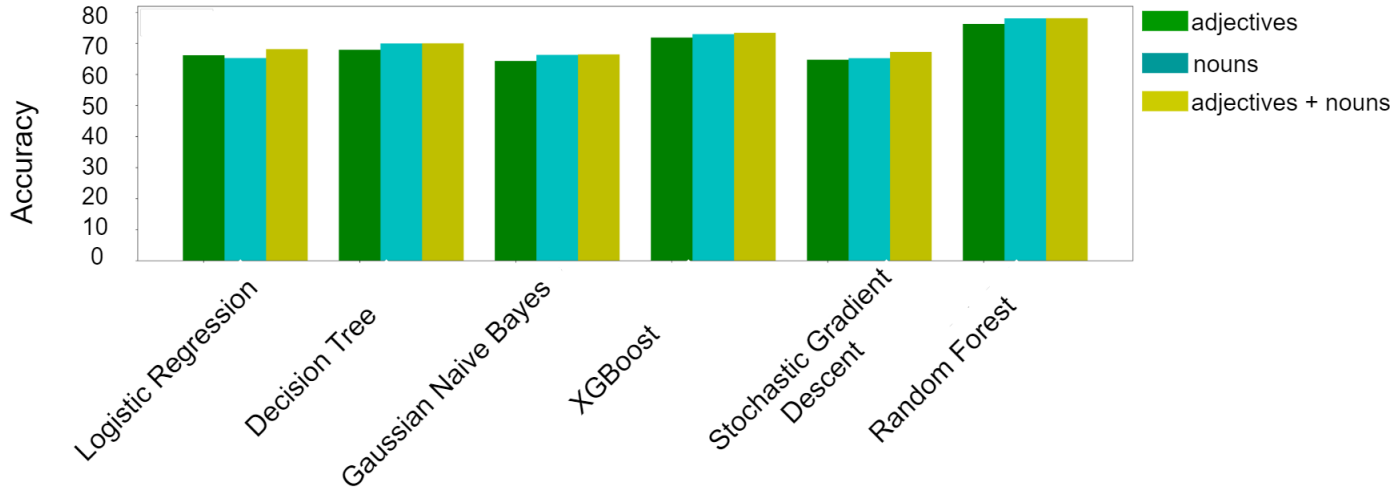



Figure 4.22: Accuracy of sentiment analysis over the context of the word

# Chapter 5

# Discussion

The results of all the above experiments were solely evaluated on the initial dataset that was released as a part of Sem Eval Task 4. The data challenge did not publicly release the testing set which was used for their evaluation, and thus the models could not be evaluated on the actual test dataset. The feature engineering for all the models was strictly done on the training set so as to not lead to any information leakage.

The best model barring any topical dependency used presence of nouns and adjectives in the text of article. This served as our final model - using nouns, adjectives, and noun AND adjectives. A random forest was used to model the features. We tested the model on web searched articles related to 'Midterm Election 2018'. The web articles were scraped using the BING web-search API and the topics can be found in subsection 3.2.2.

We scraped queries related to 'United States midterm election' and found a total of 86 other queries and at most of 100 articles for each query, resulting in a total of 6616 web articles. The following are the results of testing the partisanship of the web-searched articles using our best-performing model which uses random forest with nouns and adjectives, as seen in Table 5.1.

Table 5.1: Result of classifying the midterm elections data

| **Algorithm uses** | **No. of articles (hyperpartisan)** | **No. of articles (non-hyperpartisan)** |
|---|---|---|
| Nouns | 4051 | 2565 |
| Adjectives | 4258 | 2358 |
| Nouns & Adjectives | 4352 | 2264 |

As seen in Table 5.1, there seem to be more than half the articles indicating a hyperpartisan nature. A deeper analysis was done for the first 10 results of the web-search queries. Since users tend to click and read the articles on the first page of the search engine, we thought this analysis would serve a better perspective as to how many articles among the top 10 returned show hyperpartisanship.

Table 5.2: Result on the top 10 queries of classifying the midterm elections data

| **Algorithm uses** | **No. of articles (hyperpartisan)** | **No. of articles (non-hyperpartisan)** | **Mean (hyperpartisan)** |
|---|---|---|---|
| Nouns | 473 | 307 | 6.06 |
| Adjectives | 516 | 264 | 6.61 |
| Nouns & Adjectives | 505 | 275 | 6.47 |

Table 5.2 shows that on average a total of 6 articles were found to be hyperpartisan in nature from the first 10 displayed to the user on their web-search query.

# Chapter 6

# Conclusions and Future Work

In this thesis, we explored the different linguistic and morphological features of a text article which can be used to distinguish between hyperpartisan and mainstream news. This is an important problem to address since the implications of consuming such biased content have an impact on society and daily life. We saw how the headlines and the body of an article can separately be used to analyze if there are hyperpartisan attributes in them. Our best model worked with using both nouns and adjectives from the training corpus as features. The analysis on the midterm election dataset have also shown that more than half of the articles in it do show hyperpartisan attributes, similar is the result on the first web pages retrieved. Easy access and retrieval of hyperpartisan articles could result in misinformation overload since an Internet user mostly reads what is displayed on the first page results of a web search query. Thus, we need to have measures in place in order to block propagation of such articles, or even warn the readers of what they are about to consume.

There should be general public awareness about hyperpartisanship, and they should be educated to report instances of such articles. Reporting such content could be used in curation of more labeled dataset for training and automated detection. Additional work could include implementing an unsupervised classification and identifying the cluster traits. Identifying clusters would lead to comprehending the underlying distinguishing as well as intersecting patterns between hyperpartisan and mainstream news better.

In addition to NLP, we could also employ some domain knowledge like reporting style of a news article for better classification.

# Bibliography


[1] Bing search API. https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/. Accessed: 2019-04-09.

[2] Google trends. https://trends.google.com/trends/. Accessed: 2019-04-09.

[3] US midterm election 2018. https://en.wikipedia.org/wiki/2018_United_States_elections. Accessed: 2019-04-09.

[4] Russian troll accounts for presidential elections 2016. https://www.huffpost.com/entry/russian-trolls-fake-news_n_58dde6bae4b08194e3b8d5c4. Accessed: 2019-04-09.

[5] Semeval task 4 - hyperpartisan news detection. https://pan.webis.de/semeval19/semeval19-web/index.html. Accessed: 2019-07-01.

[6] SVM supporting hyperplanes. https://en.wikipedia.org/wiki/Support-vector_machine#/media/File:Svm_separating_hyperplanes_(SVG).svg. Accessed: 2019-05-28.

[7] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36, May 2017. doi: 10.1257/jep.31.2.211. URL http://www.aeaweb.org/articles?id=10.1257/jep.31.2.211.

[8] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 U.S. Presidential election online discussion. *First Monday*, 21(11), 2016. ISSN 13960466. doi: 10.5210/fm.v21i11.7090. URL https://firstmonday.org/ojs/index.php/fm/article/view/7090.

[9] Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the EMNLP Workshop: Natural Language Processing meets Journalism*, pages 84–89, 2017.

[10] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, pages 675–684. ACM, 2011.

[11] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16. IEEE, 2016.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages=785–794, year=2016, organization=ACM.*

[13] Yu-Chen Chen, Rong-An Shang, and Chen-Yu Kao. The effects of information overload on consumers' subjective state towards buying decision in the internet shopping environment. *Electronic Commerce Research and Applications*, 8(1):48–58, 2009.

[14] Niall J. Conroy, Victoria L. Rubin, and Yimin Chen. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4, 2015.

[15] David M. Cook, Benjamin Waugh, Maldini Abdipanah, Omid Hashemi, and Shaquille Abdul Rahman. Twitter deception and influence: Issues of identity, slacktivism, and puppetry. *Journal of Information Warfare*, 13(1):58–71, 2014.

[16] Song Feng, Ritwik Banerjee, and Yejin Choi. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics, 2012.

[17] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[18] Andrew Guess, Brendan Nyhan, and Jason Reifler. Selective exposure to misinformation: Evidence from the consumption of fake news during the 2016 US presidential campaign. *European Research Council*, 9, 2018.

[19] Peter J. Hotez. Texas and its measles epidemics. *PLoS medicine*, 13(10):e1002153, 2016.

[20] Johan Hovold. Naive Bayes spam filtering using word-position-based attributes. In *The Conference on Email and Anti-Spam (CEAS)*, pages 41–48, 2005.

[21] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[22] Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(Jun):1261–1276, 2007.

[23] George Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychological Bulletin*, 116(1):75, 1994.

[24] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48. Citeseer, 1998.

[25] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with Naive Bayes-which Naive Bayes? In *The Conference on Email and Anti-Spam (CEAS)*, volume 17, pages 28–69. Mountain View, CA, 2006.

[26] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*, 2017.

[27] Brian Richards. Type/token ratios: what do they really tell us? *Journal of Child Language*, 14(2):201–209, 1987. doi: 10.1017/S0305000900012885.

[28] Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, 2016.

[29] Karl-Michael Schneider. A comparison of event models for Naive Bayes anti-spam e-mail filtering. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics-Volume 1*, pages 307–314. Association for Computational Linguistics, 2003.

[30] Chengcheng Shao, Giovanni Luca Ciampaglia, Onur Varol, Alessandro Flammini, and Filippo Menczer. The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, pages 96–104, 2017.

[31] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explorations Newsletter*, 19 (1):22–36, September 2017. ISSN 1931-0145. doi: 10.1145/3137597.3137600. URL http://doi.acm.org/10.1145/3137597.3137600.

[32] Philip J. Stone, Robert F. Bales, J. Zvi Namenwirth, and Daniel M. Ogilvie. The

general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7(4):484–498, 1962.

[33] Andreas Vlachos and Sebastian Riedel. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pages 18–22, 2014.

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

[35] Andrew Ward, L. Ross, E. Reed, E. Turiel, and T. Brown. Naive realism in everyday life: Implications for social conflict and misunderstanding. *Values and Knowledge*, pages 103–135, 1997.

[36] Janyce Wiebe. Learning subjective adjectives from corpora. *Association for the Advancement of Artificial Intelligence*, 20(0):0, 2000.

[37] Janyce M. Wiebe, Rebecca F. Bruce, and Thomas P. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 246–253, 1999.

[38] Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. Automatic detection of rumor on Sina Weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.

# Appendices

# Appendix A

# Timing and accuracy statistics for TF-IDF on headlines of articles

The whole dataset was split into a 80-20 train and test split after random shuffling. This appendix contains the timing and accuracy statistics of hyperparameter tuning on the TF-IDF vectors of the headlines of the articles.

## A.1 Logistic Regression

In this subsection, we varied different values of the inverse regularization factor $C$.

Table A.1: Test accuracy with variation in $C$ for logistic regression

| **C** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| 0.001 | 67.92 | 4.19 |
| 0.01 | 69.53 | 6.02 |
| 0.1 | 73.68 | 12.52 |
| **1** | **77.5** | **25.25** |
| 10 | 78.6 | 53.23 |
| 100 | 78.43 | 97.83 |
| 1000 | 78.25 | 136.75 |

As seen from Table A.1, the test accuracy increases from $C = 1.0$, and as $C$ increases thereafter, there is slight improvement in the accuracy with comparable increase in time. Thus, we select $C = 1.0$ keeping the accuracy and time trade-off into consideration.

## A.2 Naive Bayes

In this subsection, we varied the hyperparameter '*min_ df*' for Gaussian Naive Bayes and Multinomial Naive Bayes. '*min_ df*' is used to limit words below a threshold while building a vocabulary, and is explained in subsection 4.4.1.

### A.2.1 Gaussian Naive Bayes

We see the results of tuning the hyperparameter '*min_ df*' using Gaussian Naive Bayes in Table A.2.

Table A.2: Test accuracy with variation in *min_df* for Gaussian Naive Bayes

| **min_df** | **Test accuracy (%)** | **min_df** | **Test accuracy (%)** |
|---|---|---|---|
| **0.0001** | **74.3** | 0.007 | 65.87 |
| 0.0002 | 73.21 | 0.008 | 59.76 |
| 0.0003 | 72.77 | 0.009 | 62.86 |
| 0.0004 | 72.22 | 0.01 | 62.47 |
| 0.0005 | 71.66 | 0.02 | 60.1 |
| 0.0006 | 71.07 | 0.03 | 59.12 |
| 0.0007 | 70.77 | 0.04 | 59 |
| 0.0008 | 70.62 | 0.05 | 58.89 |
| 0.0009 | 70.4 | 0.06 | 58.19 |
| 0.001 | 70.18 | 0.07 | 57.41 |
| 0.002 | 67.35 | 0.08 | 57.41 |
| 0.003 | 65.69 | 0.09 | 57.41 |
| 0.004 | 64.36 | 0.1 | 57.38 |
| 0.005 | 62.48 | 0.2 | 50.44 |
| 0.006 | 60.43 | 1 | Vocabulary too huge to fit into memory |
| 0.007 | 59.25 | | |

### A.2.2 Multinomial Naive Bayes

We see the results of tuning the hyperparameter '*min_df*' using Multinomial Naive Bayes in Table A.3.

Table A.3: Test accuracy with variation in *min_ df* for Gaussian Naive Bayes

| **min_ df** | **Test accuracy (%)** | **min_ df** | **Test accuracy (%)** |
|---|---|---|---|
| **0.0001** | **73.62** | 0.001 | 69.41 |
| 0.01 | 62.07 | 0.1 | 55.63 |
| 1 | Vocabulary too huge to fit into memory | | |

## A.3 Decision Tree

In decision trees, we varied the hyperparameter '*min_ df*' thus limiting our vocabulary for training and the rest of the parameters were used as default. The results can be seen in Table A.4.

Table A.4: Test accuracy with variation in *min_ df* for decision trees

| **min_df** | **Test accuracy (%)** | **min_df** | **Test accuracy (%)** |
|---|---|---|---|
| **0.0001** | **71.10** | 0.007 | 65.87 |
| 0.0002 | 71 | 0.008 | 65.09 |
| 0.0003 | 70.54 | 0.009 | 64.27 |
| 0.0004 | 70.64 | 0.01 | 63.86 |
| 0.0005 | 70.34 | 0.02 | 61.79 |
| 0.0006 | 70.67 | 0.03 | 60.11 |
| 0.0007 | 65.87 | 0.04 | 59.91 |
| 0.0008 | 65.87 | 0.05 | 59.97 |
| 0.0009 | 64.26 | 0.06 | 58.74 |
| 0.001 | 70.01 | 0.07 | 57.73 |
| 0.002 | 68.95 | 0.08 | 57.73 |
| 0.003 | 67.68 | 0.09 | 57.73 |
| 0.004 | 67.22 | 0.1 | 57.64 |
| 0.005 | 66.83 | 0.2 | 50.6 |
| 0.006 | 66.17 | 1 | 67.87 |

## A.4 Random Forest

In random forests, we tuned the hyperparameters '*max_ depth*' and '*n_ estimators*' for maximum accuracy. The hyperparameter-tuning can be seen in Table A.5. These hyperparameters are explained in subsection 4.2.5.

Table A.5: Test accuracy with variation in *max_ depth* and *n_ estimators* for random forest

| **n_estimators** | **max_depth** | **Test accuracy (%)** | **n_estimators** | **max_depth** | **Test accuracy (%)** |
|---|---|---|---|---|---|
| 30 | 10 | 57.44 | 3000 | 10 | 72.61 |
| 50 | 20 | 65.5 | 3000 | 20 | 73.51 |
| 100 | 10 | 65.3 | 3000 | 30 | 73.84 |
| 200 | 10 | 69.63 | 3000 | 40 | 74.01 |
| 300 | 10 | 70.34 | 3000 | 50 | 74.21 |
| 400 | 10 | 70.94 | 3000 | 60 | 74.34 |
| 500 | 10 | 71.25 | 4000 | 10 | 72.58 |
| 600 | 10 | 71.67 | 5000 | 10 | 72.62 |
| 700 | 10 | 71.85 | 6000 | 10 | 72.70 |
| 800 | 10 | 72.11 | 7000 | 10 | 72.80 |
| 900 | 10 | 72.15 | 8000 | 10 | 72.88 |
| 1000 | 10 | 72.20 | 9000 | 10 | 72.91 |
| 2000 | 10 | 72.53 | 10000 | 10 | 72.94 |

# Appendix B

# Timing and accuracy statistics for headlines

In this section, we tuned the inverse regularization factor $C$ for logistic regression, as explained in subsection 4.4.1. The results are for three different features of the headlines of a news article, mainly its length, the ratio of capitalized words to total words and the ratio of stop words to total words.

## B.1 Logistic Regression - length of headlines

In this subsection, we explored how length of a headline can be used in hyperpartisanship detection, as explained in subsection 4.4.2, and tuned $C$ for maximum accuracy. The result of tuning the hyperparameter $C$ can be seen in Table B.1.

Table B.1: Test accuracy with variation in $C$ for logistic regression

| **C** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| 0.001 | 45.62 | 1.04 |
| 0.01 | 45.62 | 1.01 |
| 0.1 | 45.62 | 1.01 |
| 1 | 45.62 | 1.01 |
| 10 | 45.62 | 1.01 |
| 100 | 45.62 | 1.01 |
| 1000 | 45.62 | 1.01 |

## B.2 Logistic regression - ratio of capitalized words to total words

In this subsection, we explored how presence of capitalized words can be used in hyperpartisanship detection, as explained in subsection 4.4.3, and tuned $C$ for maximum accuracy. The result of tuning the hyperparameter $C$ can be seen in Table B.2.

Table B.2: Test accuracy with variation in $C$ for logistic regression

| C | Test accuracy (%) | Time (sec) |
|---|---|---|
| 0.001 | 49.06 | 1.01 |
| 0.01 | 49.06 | 0.64 |
| 0.1 | 49.06 | 0.65 |
| 1 | 49.06 | 0.64 |
| 10 | 49.06 | 0.64 |
| 100 | 49.06 | 1.26 |
| 1000 | 49.06 | 0.64 |

## B.3 Logistic regression - ratio of stop words to total words

In this subsection, we explored how presence of stop words can be used in hyperpartisanship detection, as explained in subsection 4.4.3, and tuned $C$ for maximum accuracy. The result of tuning the hyperparameter $C$ can be seen in Table B.3.

Table B.3: Test accuracy with variation in $C$ for logistic regression

| C | Test accuracy (%) | Time (sec) |
|---|---|---|
| 0.001 | 56.17 | 0.53 |
| 0.01 | 56.17 | 0.53 |
| 0.1 | 56.17 | 0.53 |
| 1 | 56.17 | 0.53 |
| 10 | 56.17 | 0.53 |
| 100 | 56.17 | 0.53 |
| 1000 | 0.56 | 0.53 |

# Appendix C

# Timing and accuracy statistics for websites and TTR

In this section, we tuned the inverse regularization factor $C$ on the websites of the news articles and the Type Token Ratio (TTR).

## C.1 Logistic Regression - URLs

In this subsection, we varied $C$ for maximum accuracy on URLs of the websites of the news articles. More about the websites can be found in Section 4.5. The result can be seen in Table C.1.

Table C.1: Test accuracy with variation in $C$ for logistic regression

| **C** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| 0.001 | 67.25 | 1.44 |
| 0.01 | 67.25 | 1.49 |
| 0.1 | 67.25 | 1.58 |
| 1 | 67.25 | 1.49 |
| 10 | 67.25 | 1.56 |
| 100 | 67.25 | 1.43 |
| 1000 | 67.25 | 1.43 |

## C.2 Accuracy and Timing statistic using TTR

In this subsection, we report the accuracy results of all the algorithms using the TTR of the body of the news articles. More about TTR can be found in subsection 4.6.1. The result can be found in Table C.2.

Table C.2: Test accuracy using TTR on all the algorithms

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 51.71 | 1.30 |
| Decision Tree | 52.69 | 7.27 |
| Gaussian Naive Bayes | 51.43 | 0.19 |
| XGBoost | 52.09 | 17.37 |
| Stochastic Gradient Classifier | 50.39 | 0.48 |
| Random Forest Classifier | **53.43** | 4469.11 |

# Appendix D

# Timing and accuracy statistics for parts of speech on body of articles

In this section, we list the timing and accuracy statistics of using nouns and adjectives in the analysis of hyperpartisanship in news articles. More about this can be read in subsection 4.4.4.

## D.1 Adjectives

In Table D.1, we see the timing and accuracy statistics of the part of speech analysis of adjectives over the body of the news articles.

Table D.1: Test accuracy using adjectives in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 84.38 | 64.14 |
| Decision Tree | 82.08 | 527.21 |
| Gaussian Naive Bayes | 68.01 | 31.13 |
| Multinomial Naive Bayes | 78.01 | 2.75 |
| XGBoost | 82.95 | 2032.22 |
| Stochastic Gradient Classifier | 83.68 | 22.41 |
| Random Forest Classifier | **87.19** | 7501.19 |

## D.2 Nouns

In Table D.2, we see the timing and accuracy statistics of the part of speech analysis of nouns over the body of the news articles.

Table D.2: Test accuracy using nouns in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 88.25 | 93.97 |
| Decision Tree | 85.31 | 570.75 |
| Gaussian Naive Bayes | 72.56 | 40.06 |
| Multinomial Naive Bayes | 79.27 | 3.74 |
| XGBoost | 86.06 | 1900.49 |
| Stochastic Gradient Classifier | 87.58 | 21.74 |
| Random Forest Classifier | **90.14** | 7028.27 |

## D.3 Nouns and Adjectives

In Table D.3, we see the timing and accuracy statistics of the parts of speech analysis of both nouns and adjectives over the body of the news articles.

Table D.3: Test accuracy using both nouns and adjectives in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 88.98 | 154.86 |
| Decision Tree | 85.56 | 1175.81 |
| Gaussian Naive Bayes | 71.55 | 69.26 |
| Multinomial Naive Bayes | 79.45 | 5.61 |
| XGBoost | 86.07 | 3722.14 |
| Stochastic Gradient Classifier | 88.04 | 48.88 |
| Random Forest Classifier | **89.96** | 8931.88 |

# Appendix E

# Timing and accuracy statistics for sentiment analysis on parts of speech of body of articles

In this section, we list the timing and accuracy statistics of using sentiments of both nouns and adjectives in the analysis of hyperpartisanship in news articles. More about this can be read in Section 4.6.4.

## E.1 Nouns

In Table E.1, we see the timing and accuracy statistics of the sentimental analysis of nouns over the body of the news articles.

Table E.1: Test accuracy using sentiments of nouns in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 65.30 | 10.13 |
| Decision Tree | 70.00 | 251.56 |
| Gaussian Naive Bayes | 66.33 | 6.35 |
| XGBoost | 72.99 | 496.21 |
| Stochastic Gradient Classifier | 65.25 | 5.41 |
| Random Forest Classifier | **78.09** | 1579.85 |

## E.2 Adjectives

In Table E.2, we see the timing and accuracy statistics of the sentimental analysis of adjectives over the body of the news articles.

Table E.2: Test accuracy using sentiments of adjectives in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 66.24 | 12.18 |
| Decision Tree | 67.97 | 237.83 |
| Gaussian Naive Bayes | 64.38 | 7.9 |
| XGBoost | 71.94 | 469.69 |
| Stochastic Gradient Classifier | 64.74 | 4.92 |
| Random Forest Classifier | **76.27** | 1525.71 |

## E.3 Nouns and Adjectives

In Table E.3, we see the timing and accuracy statistics of the sentimental analysis of nouns and adjectives over the body of the news articles.

Table E.3: Test accuracy using sentiments of both nouns and adjectives in the body of text

| **Algorithm** | **Test accuracy (%)** | **Time (sec)** |
|---|---|---|
| Logistic Regression | 68.18 | 14.05 |
| Decision Tree | 70.05 | 419.15 |
| Gaussian Naive Bayes | 66.45 | 12.34 |
| XGBoost | 71.94 | 469.69 |
| Stochastic Gradient Classifier | 73.43 | 956.77 |
| Random Forest Classifier | **78.13** | 2201.12 |