

# Real or Fake News? Using NLP with Disaster Tweets – Addendum Edition

Lim Zheng Wei

INSTITUTE OF DATA Capstone Project

# Table of Contents

1. Literature Review.....	11
1.1 Fake News.....	11
1.1.1 Definition, Purpose .....	11
1.1.2 Types of Fake News .....	12
1.1.3 Implications of Fake News .....	13
1.1.3.1 Impacts on Health.....	14
1.1.3.2 Financial Impact.....	15
1.1.3.3 Fear .....	16
1.1.3.4 Racist Ideas .....	17
1.1.3.5 Bullying and Violence Against Innocent People .....	18
1.1.3.6 Democratic Impacts.....	18
1.1.4 Identifying Fake News.....	19
1.1.4.1 Consider the Source.....	19
1.1.4.2 Read Beyond the Headline .....	20
1.1.4.3 Check the Author .....	21
1.1.4.4 What is the Support? .....	21
1.1.4.5 Check the Date.....	22
1.1.4.6 Is This a Joke? .....	23
1.1.4.7 Check Your Biases .....	24
1.1.4.8 Consult the Experts.....	24

1.1.5	Fake News Mitigation Methods.....	26
1.1.5.1	Google.....	26
1.1.5.2	Facebook.....	27
1.1.5.3	Singapore.....	29
1.2	Feature Engineering.....	29
1.2.1	Pre-Processing.....	30
1.2.1.1	Regular Expressions.....	31
1.2.1.2	Natural Language Toolkit(NLTK).....	31
1.2.2	Vectorisation.....	32
1.2.2.1	CountVectoriser .....	32
1.2.2.2	Term Frequency Inverse Document Frequency (TF-IDF) Vectoriser.....	32
1.3	Text Classification .....	34
1.3.1	Rule-based Systems .....	35
1.3.2	Machine Learning Based Systems.....	36
1.4	Text Classification Algorithms.....	37
1.4.1	Naïve Bayes Classifier.....	38
1.4.1.1	Bayes Theorem .....	38
1.4.2	Linear Classifier .....	39
1.4.3	Support Vector Classifier .....	40
1.4.4	Ensemble Methods .....	41

1.4.4.1	Bagging .....	42
1.4.4.2	Boosting .....	43
1.4.5	Deep Learning .....	45
1.4.5.1	How Deep Learning Works .....	46
1.4.5.2	Keras .....	49
2	Methodology.....	50
2.1	The Setup .....	50
2.2	Experimental Design .....	51
2.2.1	Exploratory Data Analysis (EDA) .....	51
2.3	Basic Feature Extraction .....	53
2.3.1	Number of Words .....	54
2.3.2	Number of Characters.....	55
2.3.3	Average Word Length .....	56
2.3.4	Number of Stopwords.....	57
2.3.5	Special Character .....	58
2.3.6	Punctuations .....	59
2.3.7	Number of Numerics.....	60
2.3.8	Number of Uppercase Words .....	61
2.4	Cleaning Text.....	61
2.4.1	Removal of http .....	62

2.4.2	Remove Mentions.....	62
2.4.3	Remove RT .....	62
2.4.4	Remove Extra Whitespace and Special Symbols .....	63
2.4.5	Remove Numbers .....	63
2.4.6	Expand Contractions .....	63
2.4.7	Lowercase Words.....	64
2.4.8	Remove Stopwords.....	64
2.4.9	Remove Punctuations .....	64
2.4.10	Remove Common Words.....	65
2.4.11	Remove Rare Words .....	66
2.4.12	Spelling Correction.....	66
2.4.13	Lemmatisation and Tokenisation.....	67
2.5	WordCloud.....	68
2.6	Feature Engineering.....	70
2.6.1	Machine Learning .....	70
2.6.1.1	CountVectoriser.....	70
2.6.1.2	TF-IDF .....	71
2.6.1.2.1	Word Level .....	71
2.6.1.2.2	N-Gram Level.....	71
2.6.2	Keras.....	71

2.7	Text Classification .....	73
2.7.1	Naïve Bayes .....	73
2.7.2	Linear Classifier .....	74
2.7.3	Support Vector Classifier .....	75
2.7.4	Bagging Models (Random Forest).....	76
2.7.5	Boosting Models (Gradient Descent).....	77
2.7.6	Dense Feedforward Classifier (Keras) .....	77
2.7.6.1	Baseline Model .....	78
2.7.6.2	Reduced Model.....	79
2.7.6.3	Regularised Model.....	80
2.7.6.4	Dropout Model .....	81
	References .....	85

## Tables of Figures

Figure 1 Misinformation Matrix [4] .....	13
Figure 2 Stock Market Rises and Falls From Fake News [7] .....	15
Figure 3 Financial Loss Overall [7] .....	16
Figure 4 How to Spot Fake News [16] .....	25
Figure 5 Count Vectoriser [25] .....	32
Figure 6 Mathematical Concept of TF-IDF [26] .....	33
Figure 7 Training the Text [27] .....	36
Figure 8 Predicting the Category [27] .....	37
Figure 9 Determining Hyperplane for Support Vector Classifier [30] .....	41
Figure 10 How Bagging Works [33] .....	42
Figure 11 How Boosting Works [33] .....	44
Figure 12 Neuron Networks [36] .....	46
Figure 13 Adding Weight to the Networks [37] .....	47
Figure 14 Gradient Descent in Cost Function [38] .....	48
Figure 15 Libraries for Both ML and DL .....	50
Figure 16 Libraries for ML Only .....	51
Figure 17 Libraries for DL only .....	51
Figure 18 The first 5 columns of dataset .....	52
Figure 19 Null Values in the Dataset .....	52
Figure 20 Proportion of Real and Fake Disaster .....	53
Figure 21 Words in a Tweet .....	54
Figure 22 Characters in Tweets .....	55

Figure 23 Average Word Length in Each Tweet.....	56
Figure 24 Number of Stopwords in Real (Left) and Fake (Right) Disaster .....	57
Figure 25 Special Characters in Tweets .....	58
Figure 26 Punctuations in Real (Left) and Fake (Right) News.....	59
Figure 27 Numeric Characters in Tweets.....	60
Figure 28 Uppercase Words in Tweets .....	61
Figure 29 Samples after Cleaning .....	68
Figure 30 Most Commonly Used Words for Real Disasters.....	69
Figure 31 Most Commonly Used Words for Fake Disasters .....	69



## List of Codes

Code 1 Remove HTTP.....	62
Code 2 Remove Mentions.....	62
Code 3 Remove RT .....	62
Code 4 Removal of Extra Whitespace and Special Symbols .....	63
Code 5 Remove Numbers .....	63
Code 6 Expanding Contractions .....	64
Code 7 Lowercase Words.....	64
Code 8 Remove Stopwords.....	64
Code 9 Removing Punctuations .....	65
Code 10 Top 10 Words in the Text .....	65
Code 11 Remove Common Words.....	66
Code 12 Remove Rare Words .....	66
Code 13 Spelling Correction.....	67
Code 14 Lemmatisation and Tokenisation .....	67
Code 15 Converting to String.....	67
Code 16 Feature Labels and Train Test Split.....	70
Code 17 CountVectoriser as Feature .....	70
Code 18 TF-IDF Word Level as Features .....	71
Code 19 TF-IDF N-Gram as Features.....	71
Code 20 Setting Parameters .....	72
Code 21 Tokenise the Words in Dictionary .....	72
Code 22 Convert to Sequences.....	72

Code 23 One Hot Encoding .....	72
Code 24 Convert to One Hot Encoding .....	73
Code 25 Label Encoding .....	73
Code 26 Train Test Split after OHE and LE .....	73
Code 27 Naive Bayes CountVectoriser .....	74
Code 28 Naive Bayes TF-IDF Word Level .....	74
Code 29 Naive Bayes TF-IDF N-Gram .....	74
Code 30 Linear CountVectoriser .....	74
Code 31 Linear TF-IDF Word Level .....	75
Code 32 Linear TF-IDF N-Gram .....	75
Code 33 SVC CountVectoriser .....	75
Code 34 SVC TF-IDF Word Level .....	75
Code 35 SVC TF-IDF N-Gram Level .....	76
Code 36 Random Forest CountVectoriser .....	76
Code 37 Random Forest TF-IDF Word Level .....	76
Code 38 Random Forest TF-IDF N-Gram .....	76
Code 39 Gradient Descent CountVectoriser .....	77
Code 40 Gradient Descent TF-IDF Word Level .....	77
Code 41 Gradient Descent TF-IDF N-Gram .....	77
Code 42 Baseline Model .....	78
Code 43 Reduced Model .....	79
Code 44 Regularised Model .....	80
Code 45 Dropout Model .....	81

## Appendix

Appendix 1 Contractions Dictionary .....	82
Appendix 2 List of Stopwords .....	82
Appendix 3 Modelling for Machine Learning .....	83
Appendix 4 Modelling for Keras .....	84

# 1. Literature Review

## 1.1 Fake News

### 1.1.1 Definition, Purpose

“Fake news” essentially means fabricated news. Fake news is an invention, a hoax created out of nowhere – false information that appears to be real news with the purpose of tricking people.

Fake news used to be common in print but has increased with the rise of social media [1]. Post-truth politics, political polarization, confirmation bias, and social media algorithms have been reasons why fake news are being spread. It is sometimes generated and propagated by hostile foreign actors, especially the elections period [2]. The use of anonymously hosted fake news websites has made it challenging to prosecute origins of fake news. In some definitions, fake news includes satirical articles misinterpreted as genuine, and articles that employ sensationalist or clickbait headlines that are not reflected in the text.

Fake news can negate the impact of real news by competing with it; a BuzzFeed analysis found that the top fake news stories about the 2016 U.S. presidential election received more engagement on Facebook than top stories from major media outlets [3]. It has the potential to undermine trust in serious media coverage too.

To understand the current information ecosystem, it is broken down into three elements:

1. The various types of content that are being generated and spread
2. The reasons of those who create this content

### 3. The ways this content is being disseminated

#### 1.1.2 Types of Fake News

Claire Wardle of First Draft News narrows down fake news to seven types [4]:

1. Satire or Parody - No intention to cause harm but can deceive others
2. False Connection – The headlines, visuals or captions do not correspond with the content
3. Misleading Content - Misleading use of information to set up an issue or an individual
4. False Context - Genuine content is shared with wrong contextual information
5. Impostor Content - Genuine sources are impersonated with false, made-up sources
6. Manipulated Content - Genuine information or imagery being manipulated to deceive
7. Fabricated Content - Creating new content which is 100% false, with intention to deceive and do harm








FIRSTDRAFT		MISINFORMATION MATRIX					
	 SATIRE OR PARODY	 FALSE CONNECTION	 MISLEADING CONTENT	 FALSE CONTEXT	 IMPOSTER CONTENT	 MANIPULATED CONTENT	 FABRICATED CONTENT
POOR JOURNALISM		✓	✓	✓			
TO PARODY	✓				✓		✓
TO PROVOKE OR TO 'PUNK'					✓	✓	✓
PASSION				✓			
PARTISANSHIP			✓	✓			
PROFIT		✓			✓		✓
POLITICAL INFLUENCE			✓	✓		✓	✓
PROPAGANDA			✓	✓	✓	✓	✓

Figure 1 Misinformation Matrix [4]

Figure 1 illustrated the motivations behind each type of fake news. The types were loosely arranged based on the intent to deceive. Generally, these were the 8 key factors or the 8 “P” for the existence of fake news – Poor Journalism, Parody, Provoke, Passion, Partisanship, Profit, Political and Propaganda.

### 1.1.3 Implications of Fake News

[5] Fake news and other types of false information could take on different faces. They can also have major impacts, because information shape people’s opinion, people make important decisions based on information. People form an impression about people or a situation by obtaining information. So, if the information received from the web is invented, false, exaggerated, or distorted, people would not make good decisions. Below are some examples and impacts of fake news.

### **1.1.3.1 Impacts on Health**

[6]Even though vaccines are largely proven by the scientific and medical community they are harmless, and necessary for preventing the spread of disease, there are resourceful anti-vaccination activists working tirelessly to convey the impression that vaccines are dangerous and are accountable for countless deaths annually.

Many anti-vaccination advertisements on Facebook were funded by two groups, as The Washington Post reports. They have been successful in using Facebook advertisements to target the individuals they want to reach.

As per NBC News' analysis, the anti-vaccination creators that received the highest interaction are Adams' Natural News; Children's Health Defense, an organization spearheaded by the anti-vaccine activist Robert Kennedy Jr., and Stop Mandatory Vaccination, a website led by Larry Cook. More than a million engagements happened courtesy of their contents that are on the NBC News list.

The Children's Health Defense articles, which went viral, misinterpret research in order come to conclude that vaccines are dangerous for children, teenagers and for pregnant women.

Stop Mandatory Vaccination's articles describe accounts from parents claiming their child's death due to vaccination. As NBC News writes, with medically proven scientific explanations, the accusations from these articles have been dismissed. The causes of deaths

for those cases are due to sudden infant death syndrome, pneumonia, and accidental asphyxiation.

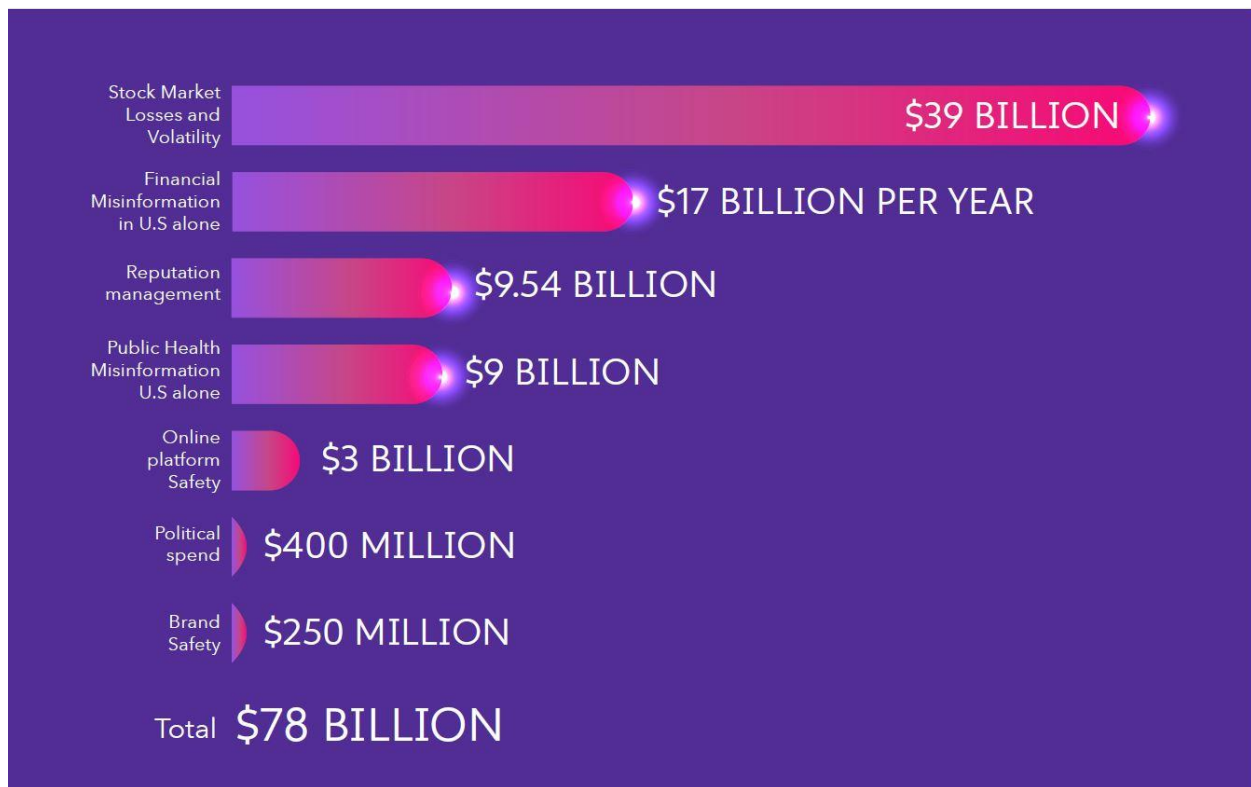
According to Sky News, fewer people are taking vaccines for measles because of fake news even though science has shown vaccines are generally safe.

### 1.1.3.2 Financial Impact



Figure 2 Stock Market Rises and Falls From Fake News [7]





*Figure 3 Financial Loss Overall [7]*

Figure 2 showed how fake news caused the stock market to rise and fall. For example, in 2019, rumors were spread about Metro Bank facing financial difficulties, causing their stock prices to fall. Figure 3 is a summary of the breakdown of different sectors financial cost being incurred, with a staggering amount of 78 US billion dollars annually due to fake news [7].

### **1.1.3.3 Fear**

Spreading fake news could cause panic to the public. False online rumours said that there was a shortage of toilet rolls due to Covid-19. This prompted people to panic buy toilet paper, even though manufacturers assured that was not the case, and the government urging the public not to stockpile [8]. On 17 February, an armed gang stole 600 rolls worth \$130 in the district of Mong Kok, Hong Kong.

In Tokyo, shoppers were limited to one packet of toilet roll per person after fake news misled people to believe that masks and toilet rolls use the same raw material for production, causing a shortage of paper [9].

In Australia, a fight broke out between 3 women at a supermarket in Chullora, a suburb outside Sydney, over toilet paper. The situation deteriorated so bad that police had to intervene. The ladies resorted to screaming and hair-pulling after one of them packed her trolley full of toilet paper at a local Woolworths store [8].

#### **1.1.3.4 Racist Ideas**

The study conducted out by Imran Awan, Professor of Criminology at Birmingham City University and Roxana Khan-Williams concluded that a rise in online Islamophobic hate speech was due to Covid-19 [10].

Fake news theories shared online included:

- Mosques are responsible for spreading COVID-19
- Muslims are super-spreaders of the coronavirus
- Police show favouritism to Muslims because they were scared that they were being accused of racism
- Muslims are not practicing social distancing

#### **1.1.3.5 Bullying and Violence Against Innocent People**

In India, misinformation on Whatsapp led to a fatal mob attack. A group of 5 friends were accused of being child kidnappers. 1 of them was killed by violent attacks while 2 others suffered injuries [11].

[12]In Hong Kong, the allegations began spreading online after Alex Chow fell off the edge of a parking garage. The police who dispersing protesters with tear gas nearby were alleged to pursue and perhaps even push the 22-year-old student. The posts also claimed that the law enforcers blocked an ambulance from reaching Chow, delaying help that could have saved his life. Although that the claims were unverified, that police denied any wrongdoings with regards to Chow and that mainstream news outlets, including the South China Morning Post, described the circumstances of his fall as unclear. Hundreds of protesters seized on his Nov. 8 death to engage in clashes with police that resulted in one person being shot on 3 days later.

#### **1.1.3.6 Democratic Impacts**

The emergence of fake news increased the outbreak of misinformation in the internet era. Concern over this problem is global. The public is made anxious about false information that continues to increase. Difficult to distinguish which information is right and wrong in social media. This phenomenon causes democracy to be corrupted. Some analysts call it a period of "post-truth" [13]. Post-truth refers to the a situation in which people are more likely to accept an argument based on their emotions and beliefs, rather than one based on facts [14].

[13]During the U.S. presidential election of 2016, Wikileaks revealed that many reporters from mainstream news agencies were working with the Clinton campaign to ensure her victory over Democrat primary opponent Bernie Sanders. Media coverage of the contest usually contained solely the slightest pretence of neutrality. In one example of biased coverage, a review of approximately 200 editorials and op-eds in the Washington Post from January to June 2016 found five negative stories for every positive one on Sanders. The ratio between positive and negative stories on Clinton were about the same. Wikileaks also revealed that the Clinton campaign was collaborating with their allies in the media to bolster what they perceived as the most vulnerable Republican candidates: Ben Carson, Ted Cruz, and Donald Trump. When Trump won the Republican nomination to represent their party to run for president, the press swiftly turned on him and showed blatant favoritism for Clinton.

In short, the impacts are very real, and we must avoid sharing fake news.

#### **1.1.4 Identifying Fake News**

Eugene and Lori wrote an article to educate others on identifying fake news before the people should trust the news using the 8 tips below [15].

##### **1.1.4.1 Consider the Source**

Some of the websites are bogus, they appear to be the genuine company, but the URL shows it is not the case. For instance, abcnews.com.co is not the actual ABC News, WTOE 5 News which says that they are a parody news website, and the Boston Tribune

which provided only one email address to contact them. The experts were able to debunk the claim that the Obamas were buying a vacation home in Dubai, a made-up story from WhatDoesItMean.com. The site claimed that they are one of the best websites in the world for “New World Order, Conspiracy Theories and Alternative News”. They did add that most contents on its site is fictional.

There are sites which provide disclaimer that those are fictional news, like WTOE 5, which published the bogus headline, “Pope Francis Shocks World, Endorses Donald Trump for President, Releases Statement.” There are also sites being dishonest, like the Boston Tribune, which provided no information on what it does, working staffs or location of headquarters. This further suggests the site is not a legitimate news organization. The site used to be named Associated Media Coverage, after its work had been debunked by fact-checking organizations.

Websites such as Snopes.com, which has been writing about viral claims and online rumours since the mid-1990s, maintains a list of known fake news websites, several of which have surfaced in the past two years.

#### **1.1.4.2 Read Beyond the Headline**

If the headline seemed to be too good to be true and gathered attention, read a little further before deciding whether to pass along the shocking information. Even if the news are real, the headline does not always tell the whole story. Fake news however, especially efforts to be satirical, can include several revealing signs in the text. That abcnews.com.co story that was being investigated, titled “Obama Signs Executive Order Banning The Pledge Of Allegiance In Schools Nationwide,” went on to quote “Fappy the Anti-Masturbation

Dolphin.” Many readers asked the experts whether this viral rumor was legitimate, suggesting they did not read the full story.

#### **1.1.4.3 Check the Author**

Another method to spot a fake story is often the byline. One of the stories on abcnews.com.co was supposedly written by “Jimmy Rustling.” His author page claimed he is a “doctor” who won multiple prestigious awards, including “fourteen Peabody awards and numerous Pulitzer Prizes.” While the resume is impressive, it is fake. No one named “Rustling” has won a Pulitzer or Peabody award. The photo attached to Rustling’s bio is also displayed on another bogus story on a different website, but this time the article is written by “Darius Rubics.”

#### **1.1.4.4 What is the Support?**

On many occasions these bogus stories will cite official sources, but once they are being investigated, the source does not support the claim. For instance, the Boston Tribune site falsely claimed that President Obama’s mother-in-law was getting a lifetime pension from the government for taking care of her granddaughters in the White House, citing “the Civil Service Retirement Act” and providing a link, but the link is directed to a government benefits website which does not support the claim at all.

Another viral claim being checked was a graphic displaying crime statistic on the percentage of whites killed by blacks and other murder statistics by race. Then-presidential candidate Donald Trump shared on his Twitter account, informing Fox News commentator

Bill O'Reilly that it came from credible sources. But almost every figure in the image was incorrect since the FBI crime data is publicly available and the supposed source given for the data, "Crime Statistics Bureau – San Francisco," does not exist.

#### **1.1.4.5 Check the Date**

Some false stories are not completely fake, but rather distortions of real events. These dishonest claims can take a true story and tweak what it says or even claim that incidents that occurred ages ago is related to current events.

Since Trump became president, many inquiries were received wanting to find out whether Ford had shifted car production from Mexico to Ohio, because of Trump's election. Readers cited various several items that quoted from and linked to a CNN Money article titled "Ford shifts truck production from Mexico to Ohio." However, that news originated in August 2015, which clearly showed that the outcome of the election is not the reason Ford made any move.

One bogus website took CNN's 2015 story without crediting them and created a new headline and publication date on it, claiming that Ford had relocated their truck production from Mexico to Ohio since Donald Trump won the presidential election. Not only is the headline untrue, copyright violation is involved as well.

If this Ford story sounds familiar, that is because the CNN article has been taken out of context before.

In October 2015, Trump wrongly boasted that Ford would build a plant in Ohio instead of building new plants in Mexico. Trump believed he was the reason for Ford's

alleged change of heart and shared on Twitter a link to a story on a blog called Prntly.com, which cited the CNN Money story. However, Ford had never altered their plans, and Trump deserved no credit.

In truth, the CNN article was about the movement of several pickup assembly work from Mexico to Ohio, a move announced by Ford back in March 2014. Ford released a statement that the plans for new plants in Mexico were still on. The statement also mentioned that they had not spoken with Mr. Trump and they did not make any adjustments to their plans.

#### **1.1.4.6 Is This a Joke?**

There is such thing as satire. Usually, it is clearly mentioned as such, and sometimes it has comical effect. Andy Borowitz has been writing a satirical news column named “the Borowitz Report” since 2001, and it has been featured in the New Yorker since 2012. But not everyone understands the jokes.

Among the headlines being flagged: “Putin Appears with Trump in Flurry of Swing-State Rallies” and “Trump Threatens to Skip Remaining Debates If Hillary Is There.” When experts informed readers those were satirical columns, some expressed their scepticism because the details were far-fetched and wanted to be sure.

The posts by Horner and others, be it satire or just “fake news” are designed with the intention to clickbait and earn money for the creator through advertisement revenue. Horner told the Washington Post his income are dependent on his posts. When questioned



why his material garnered so many views, Horner responded by saying people just keep spreading articles around without verifying the source anymore.

#### **1.1.4.7 Check Your Biases**

Confirmation bias guides people to put more weightage in information that confirms their beliefs and discount information that does not.

The experts were encouraged by some of the readers' responses — like the ones uncertain of Borowitz's columns — when they questioned whether they were genuine news, and just wanted to be sure their scepticism is justified. But they were also discouraged when they saw debunked claims gain new life.

There was a resurgence of a fake quote from Donald Trump since the election — a viral image being circulated claims Trump told People magazine in 1998 that if he were to run for president, he would run as a Republican. He added that they were the dumbest group of voters in the country because they believed anything on Fox News. He could lie and they would still eat it up. He bet that his numbers would be terrific. No such quote was found in People's archives from 1998, or any other date. A public relations representative for the magazine confirmed that there were no such quotes or any interview with Trump in 1998.

#### **1.1.4.8 Consult the Experts**

Debunking claim can take time. But there are people being paid to do carry out these investigations. It is likely at least one of FactCheck.org, Snopes.com, the Washington Post

Fact Checker and PolitiFact.com has already fact-checked the latest viral claim to pop up in news feed.

FactCheck.org was among a network of independent fact-checkers who signed an open letter to Facebook's Mark Zuckerberg suggesting that Facebook to begin an open conversation on the basis that could build a more authentic news ecosystem on its News Feed. They hope that dialogue occurs, but readers themselves remain the first line of defence against fake news. A summary of the points mentioned above is shown in Figure 4 below.

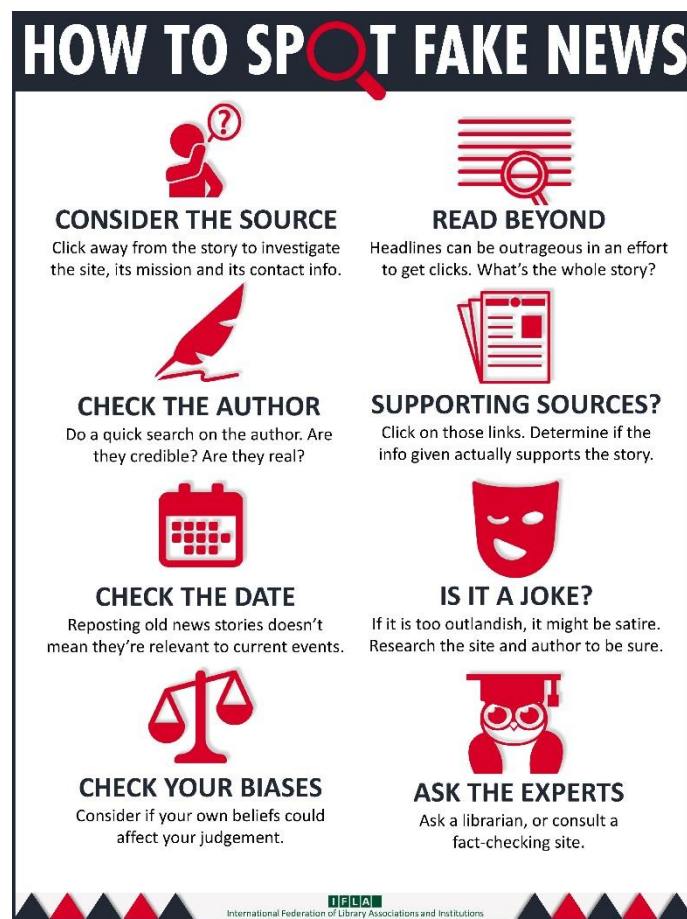


Figure 4 How to Spot Fake News [16]

### 1.1.5 Fake News Mitigation Methods

#### 1.1.5.1 Google

[17]Like every online platform, Google has suffered from all the fake news that has dominated headlines for the past few years. The company has taken several actions to combat the problem, ranging from collaborating with fact-checking companies to launching the Google News Initiative which cost \$300 million. Now it is expanding its transparency efforts more by being detailed at the steps taken to fight disinformation across its services.

At the Munich Security Conference, Google disclosed their plan and elaborated how it deals with "deliberate efforts to deceive and mislead using the speed, scale and technologies of the open web" across Search, News and YouTube. Its work falls generally into three categories: quality, malicious actors, and context.

The plan also outlined the specifics the company's 20 years' experience in fighting spam, and how this can be used against content creators that attempt to trick the systems to get more visibility. And the company points to its Knowledge and Information Panels in Search and YouTube to aid users with more context and background on the information they are seeing, and why they are seeing it.

The plan mentions shortly on the company's future plans, noting that better media literacy will play a big role in combating fake news, while threats are likely to include vicious event around democratic elections, and "deepfakes", or "synthetic media" generated by AI that could sidestep layers of security.

The company also confirmed that it is changing its AI to reduce the promotion of hoax videos and fake news on YouTube. Researchers have already correlated the rise of Flat Earthers to the availability of conspiracy videos hosted on site, with one study finding that

29 of 30 believers never thought the Earth to be flat until viewing clips promoting the theory on YouTube.

#### **1.1.5.2 Facebook**

[18]Facebook is attempting to redefine authoritativeness on the internet as part of its efforts to fight the spread of misinformation and abuse on its platforms.

The company rolled out a bunch of announcements that aim to push more reliable news sources, block Groups that spread misinformation, and offer the public a lot of insight into how Facebook crafts its content policies. The changes, broadly, obtain to nurture what Facebook refers to as "integrity" on the platform at a time once many users, regulators, and politicians have come to see Facebook and its alternative apps—WhatsApp, Instagram, and Messenger—as the chief propagators of propaganda, hate speech, and fake news online.

The feature among these news is the debut of a metric named Click-Gap, which Facebook's News Feed algorithms will utilise to determine where to grade a given post. Click-Gap is the company's attempt to limit the spread of websites that are unusually popular on Facebook compared with the rest of the web. If Facebook detects many shares being linked to a certain website, but few websites on the broader web are linking to that site, Facebook will use that signal, among others, to limit the website's reach.

Click-Gap could be bad news for fringe sites that maximise their content to go viral on Facebook. Some of the most popular stories on Facebook come small domains specifically tailored to appeal to Facebook's algorithms.

Facebook is also focusing on Groups, where so much amplification happens. Private or semi-private Groups are trending in recent years, partly due by Facebook's push to promote communities. At the same time, they have become perfect environment for spreading misinformation, radicalization, and abuse. Groups are a particularly tricky part of Facebook's ecosystem, because they are confidential, isolated, and often opaque. Because many are secret or closed, when abuse happens, members must report, or Facebook's automated tools do not detect it.

Facebook will now take a harsher approach for administrators of toxic Groups and will factor in moderator behaviour when assessing the health of a group. Rosen and Lyons wrote that when reviewing a group to decide whether or not to take it down, they would look at admin and moderator content violations in that group, including member posts they had approved, as a stronger signal that the group violated their standards.

Facebook will also punish these Groups for spreading fake news, which does not always break community rules. In much the same method the platform reduces the reach of sites that get repeatedly dinged by its partner fact-checkers, Facebook will now downgrade the reach of Groups it finds to be steadily sharing links to such sites. That, the company hopes, will make the Groups tougher to locate.

Finally, Facebook has begun publishing changes it makes to its community standards, which dictate what is and is not allowed on the platform. This is a lengthy, living document, which Facebook is constantly reassessing. Facebook made this document public last year, but until now, there has not been an easy way to track the changes the company makes to it over time.

### **1.1.5.3 Singapore**

In 2017, the Ministry of Communications and Information set up Factually, a website intended to debunk false rumours regarding issues of public interest such as the environment, housing and transport [19] while in 2018, the Parliament of Singapore formed a Select Committee to consider new legislation to tackle fake news [20].

Furthermore, the Singapore government has introduced draft legislation with regards to Fake News in April 2019, which is called the Protection from Online Falsehoods and Manipulation Bill [21]. This legislation is intended to regulate websites that spread misinformation and combat fake news.

## **1.2 Feature Engineering**

[22]Feature engineering is a course of transforming the given data into something that is clearer to interpret. The aim is to make the data more transparent for a Machine Learning (ML) model, but some features can be generated so that the data visualization can be understood for people without a data-related knowledge. However, the theory of transparency for the machine learning models is not straightforward because different models often call for a particular concept for the different kinds of data.

To use textual data for predictive modelling, the text must be translated to extract some specific words. This process is called tokenization. These words need to then be encoded as whole numbers, or floating-point values, before they can be used as inputs in machine learning algorithms. This process is called feature extraction (or vectorisation).

### 1.2.1 Pre-Processing

[23]An easy method is to assume that the smallest unit of information in a text is the word (as opposed to the character). Therefore, texts are being represented as word sequences. For instance:

Text: This is a cat. Word Sequence: [this, is, a, cat]

In this example, the punctuation is removed, and each word is being lowercased because punctuation and letter case are being assumed to have no impact on the meaning of words. In truth, distinctions are avoided between similar words such as This and this or cat. and cat.

Moreover, real life text is often not cleaned. Because this text is usually extracted from web scraping, some HTML code can get bundled with the actual text. Therefore, these texts need some tidying to prevent having HTML code words inside the word sequences. For example:

<div>This is not a sentence.<\div> --> [this, is, not, a, sentence]

Editing the texts before converting them into word sequences is called pre-processing. Despite being relatively straightforward, the pre-processing techniques seen so far work very well in practice. Varying on the types of texts, it may be necessary to add more complicated pre-processing steps. The more steps to clean text, the longer the pre-processing will take.

Using Python 3, pre-processing function can be written that takes a block of text and then outputs the cleaned version of that text. These chunks of blocks are called regular expressions.

#### **1.2.1.1 Regular Expressions**

[23]A regular expression (or regex) is a string of characters that serve as a explore order. Each character has a meaning; for example, '.' means any character except newline character: '\n'. These characters are often linked with quantifiers, such as \*, which means zero or more occurrence. Merging these two characters, the regex looks for an expression in the form '<' + 'zero or more' of 'anything but \n' + '>'. This regex is <.\*?>. Here the character '?' specifies a non-greedy search.

Regular expressions are very useful for processing strings. For example, the <.\*?> regex can be used to detect and remove HTML tags. Other regex are being used such as \' to remove the character ' so that words like that's become “that’s” instead of two separate words that and s.

#### **1.2.1.2 Natural Language Toolkit(NLTK)**

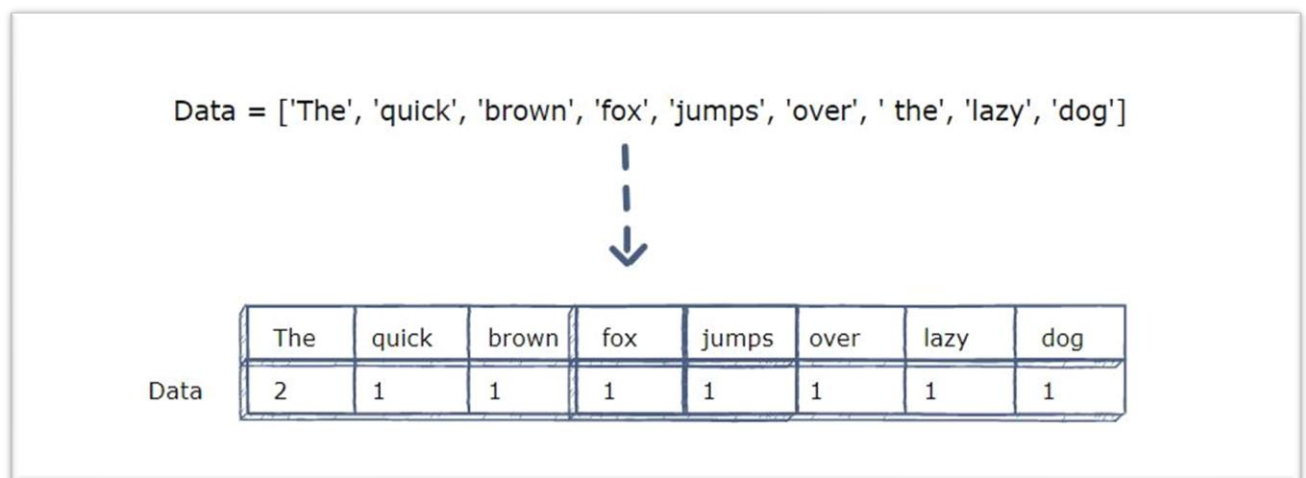
[24]NLTK is a go-to platform for constructing Python programs to analyse human language data. It offers user-friendly interfaces to over 50 corpora and lexical resources such as WordNet, as well as a collection of libraries for text processing such as for classification, tokenization, stemming and many others, and a forum for discussion should people have queries.



## 1.2.2 Vectorisation

### 1.2.2.1 CountVectoriser

[25]CountVectorizer is used to transform a compilation of text documents to a vector of term/token counts. It also facilitates the pre-processing of text data before developing the vector representation. This process makes it a versatile feature representation module for text. Figure 5 below shows the process of CountVectorizer.



*Figure 5 Count Vectoriser [25]*

### 1.2.2.2 Term Frequency Inverse Document Frequency (TF-IDF) Vectoriser

[26] TF-IDF is a very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. While Count Vectorizer give number of frequencies with respect to index of vocabulary, TF-IDF consider overall documents of weight of words. Figure 6 depicted a summary of the mathematical concept of TF-IDF.

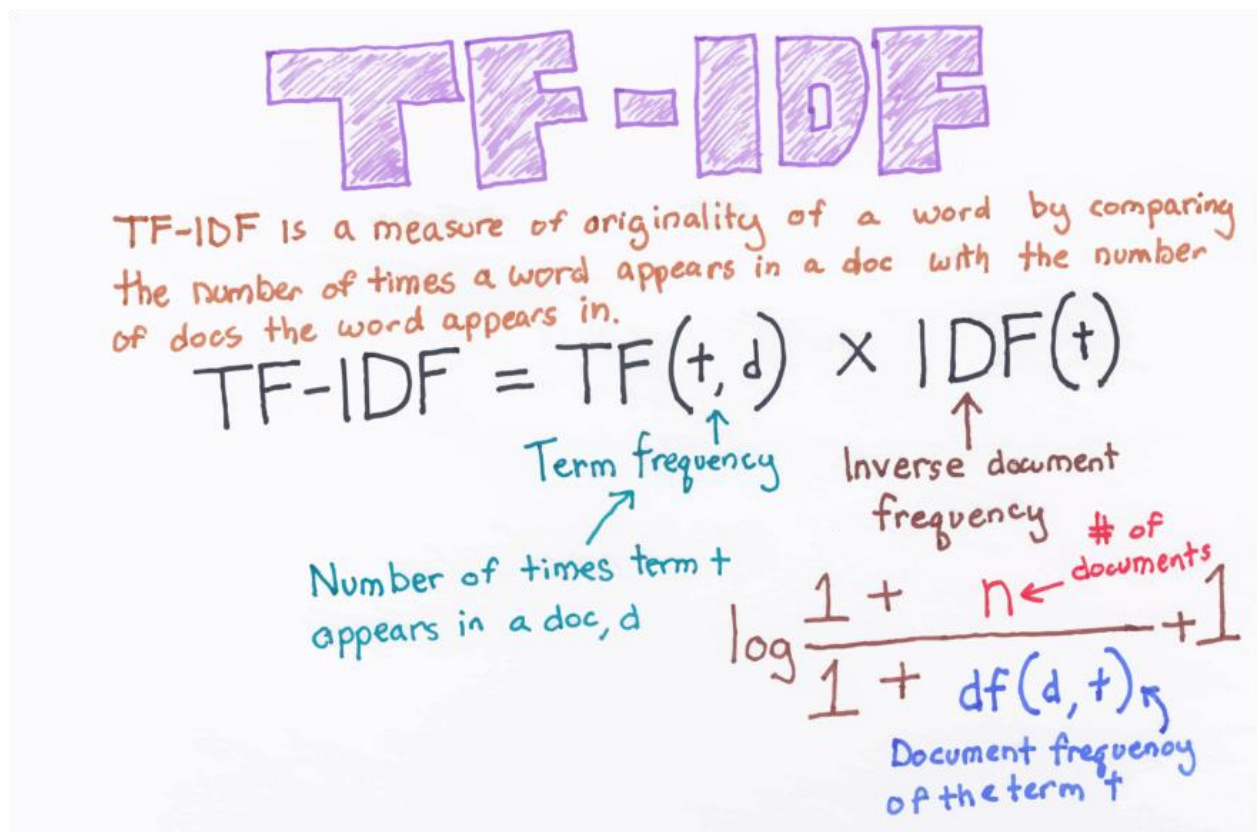


Figure 6 Mathematical Concept of TF-IDF [26]

TF-IDF is the product of term frequency and inverse document frequency. The objective is to determine the emphasise of a keyword or phrase within a document or page.

The term frequency  $TF(t, d)$  refers to the frequency the term  $t$  appears in a document  $d$ .

The inverse document frequency is a measure of how much information the word provides, i.e., if it is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$IDF(t) = \log \frac{|N|}{1 + |\{d : t \in d\}|}$$

Where N is the corpus,  $\{d : t \in d\}$  refers to the number of documents where the term  $t$  appears, when the term-frequency function satisfies  $TF(t, d) \neq 0$ . 1 is added to prevent zero-division.

The formula for TF-IDF is therefore:

$$TF - IDF(t) = TF(t, d) \times IDF(t)$$

This formula has an importance effect that a high weight of the tf-idf calculation is reached when we have a high term frequency in the given document(local parameter) and a low document frequency of the term in the whole collection ( global parameter). TF-IDF can be performed using N-gram model as well in which a sequences of words are taken into account.

### 1.3 Text Classification

[27] Text classification is the assignment of selecting a set of pre-determined conditions to free-text. Text classifiers can be used to organise, structure, and categorise. For example, news articles can be split by topics, chat conversations can be sorted by language, brand mentions can be organized by sentiment.

Text classification can be done manually and automatically. In the former, humans decipher the content of text and categorise it accordingly. This method usually provides quality results, but it is time-consuming and expensive. The latter applies techniques such as

machine learning, natural language processing to automatically classify text in a quicker and more cost-effective way.

There are many approaches to automatic text classification, which have broadly grouped into three different categories:

1. Rule-based
2. Machine Learning based
3. Hybrid

### 1.3.1 Rule-based Systems

Rule-based sorts text into structured groups by using a set of customised linguistic rules. These rules dictate the system to use semantically relevant elements of a text to pick out relevant categories based on its content. Each rule is made up of a trend and an expected category.

For example, the job is to classify news articles into Sports and Politics. First, two lists of words that characterise each group are generated (words related to sports and politics respectively). Next, when classifying a new text, calculate how many sport-related and politics-related words that appear in the text. If the number of sport-related word appearances is fewer than the number of politics-related word count, then the article would be considered political and vice versa.

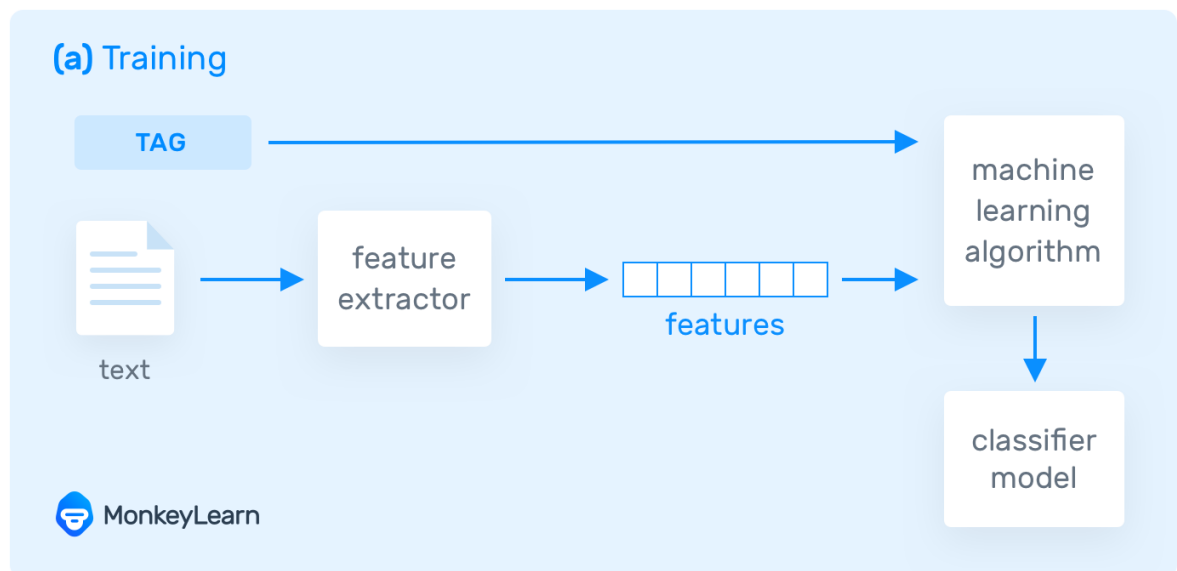
Rule-based systems are apprehensive and can be improved over time. However, these systems require deep knowledge of the domain. They are also time-consuming, since coming up with rules for a sophisticated system can be a challenge and usually requires a lot of

analysis and testing. Rule-based systems are also hard to maintain and do not scale well when new rules are added, which can affect the results of the pre-existing rules.

### 1.3.2 Machine Learning Based Systems

Machine learning learns to make classifications based on past observations instead of depending on customised rules. By using pre-determined examples as training data, a machine learning algorithm can learn the different associations between pieces of text and that a particular tag is expected based on a particular text.

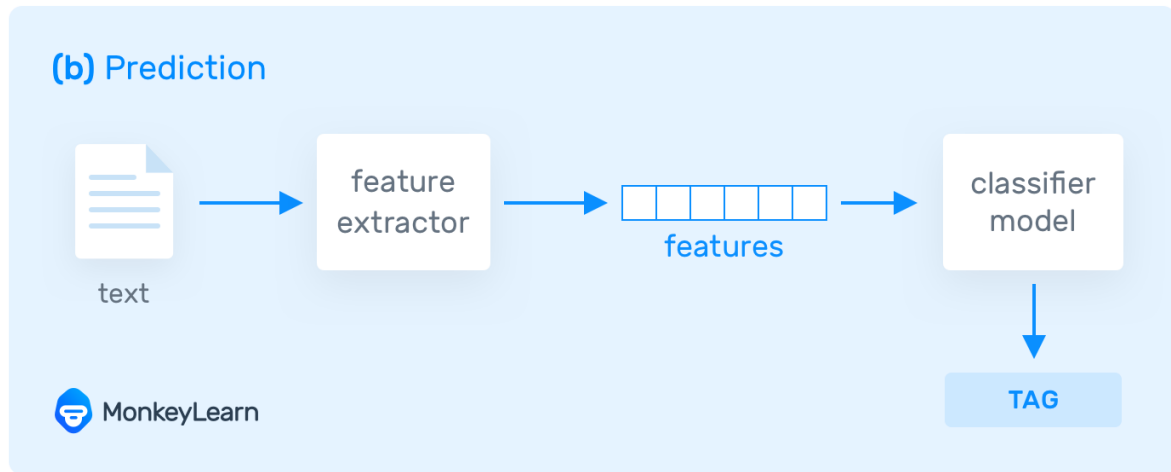
After feature extraction is done, the algorithm is fed with training data that is made up of pairs of feature sets (vectors for each text example) and tags (e.g. sports, politics) to come out with a classification model as shown in Figure 7:



*Figure 7 Training the Text [27]*

Once it is trained with sufficient training samples, the model can start to predict more accurately. The same feature extractor is used to transform new data to feature sets which

can be fed into the classification model to get predictions on tags as shown in Figure 8 (e.g. sports, politics):



*Figure 8 Predicting the Category [27]*

Text classification with machine learning is usually much more accurate than human-crafted rule systems, especially on complex classification tasks. Also, classifiers with machine learning are easier to maintain and new examples can always be tagged to learn new tasks.

## 1.4 Text Classification Algorithms

Some of the most popular machine learning algorithms for creating text classification models include the naive bayes family of algorithms, support vector machines, and Deep Learning(DL).

### 1.4.1 Naïve Bayes Classifier

Naïve Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem, used in a wide variety of classification tasks.

#### 1.4.1.1 Bayes Theorem

Bayes' Theorem is a simple mathematical formula used for calculating conditional probabilities [28].

Conditional probability is a method of the probability of an event occurring given that another event has occurred.

The formula is:

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

Where

$P(A|B)$  refers to the probability that event A occurs given that event B has occurred

$P(B|A)$  refers to the probability event B occurs given that event A has occurred

$P(A)$  refers to the probability event A happens

$P(B)$  refers to the probability even B happens

Naïve Bayes assumes that each feature makes an independent and equal contribution to the outcome.

Combine all the pre-processing techniques and create a dictionary of words and each word's count in training data.

Probability is calculated for each word in a text and words with a probability less than threshold probability are filtered out because they are irrelevant.

Then for each word in the dictionary, create a probability of that word being in insincere questions and its probability insincere questions. Then the conditional probability is found to use in Naïve Bayes classifier.

#### 1.4.2 Linear Classifier

[29]Linear models are popular for solving classification tasks. They seek to divide the feature space into a collection of regions labelled according to the values the target can take, where the decision boundaries between those regions are linear: they are lines in 2D, planes in 3D, and hyperplanes with more features.

Another approach to linear classification is the logistic regression model. Logistic regression models the probabilities of an observation belonging to each of the K classes via linear functions, ensuring these probabilities add up to one and stay in the (0, 1) range. The model is specified in terms of K-1 log-odds ratios, with an arbitrary class chosen as reference class (in this example it is the last class, K). Consequently, the gap between log-probabilities of belonging to a given class and to the reference class is modelled linearly as

$$\log \frac{\Pr(G = 1 \mid X = x)}{\Pr(G = K \mid X = x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{\Pr(G = 2 \mid X = x)}{\Pr(G = K \mid X = x)} = \beta_{20} + \beta_2^T x$$

...



$$\log \frac{\Pr(G = K - 1 | X = x)}{\Pr(G = K | X = x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

where  $G$  stands for the true, observed class. The probabilities of an observation belonging to each of the classes can be calculated as

$$\Pr(G = k | X = x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}, k = 1, \dots, K - 1$$

$$\Pr(G = K | X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}$$

which shows that all class probabilities add up to one.

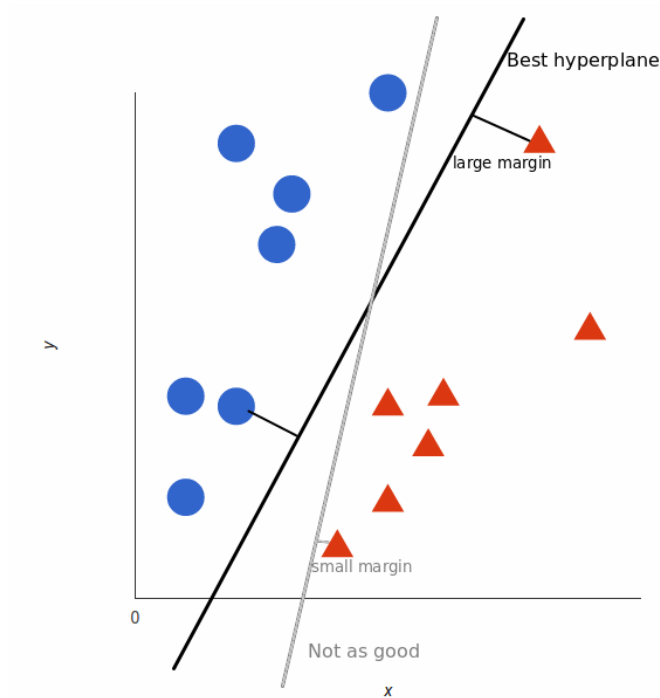
Logistic regression models are usually predicted by maximum likelihood, which is taken care of by scikit-learn. Linear models for regression can be regularized to improve accuracy, likewise for logistic regression.

### 1.4.3 Support Vector Classifier

[30]A support vector machine (SVM) is a supervised machine learning model that utilises classification algorithms for binary classification. SVM model can categorise new text after training data are labelled for each category.

With reference to Figure 9, the SVM takes these data points and outputs the hyperplane (or a line for two dimensions) that best segregates the tags. This line or hyperplane is the decision boundary: anything that falls to one side of it will be classified as blue, the rest will be grouped as red.

For SVM, the best hyperplane is the one that maximizes the margins from both tags. In other words: the hyperplane whose distance to the nearest element of each tag is the largest.



*Figure 9 Determining Hyperplane for Support Vector Classifier [30]*

Using word frequencies, just like how it done in Naive Bayes, every text in the dataset is represented as a vector with thousands (or tens of thousands) of dimensions, every one representing the frequency of one of the words of the text. All of these will be given to SVM for training. Using pre-processing techniques, like stemming, removing stopwords, and using n-grams can improve results.

#### 1.4.4 Ensemble Methods

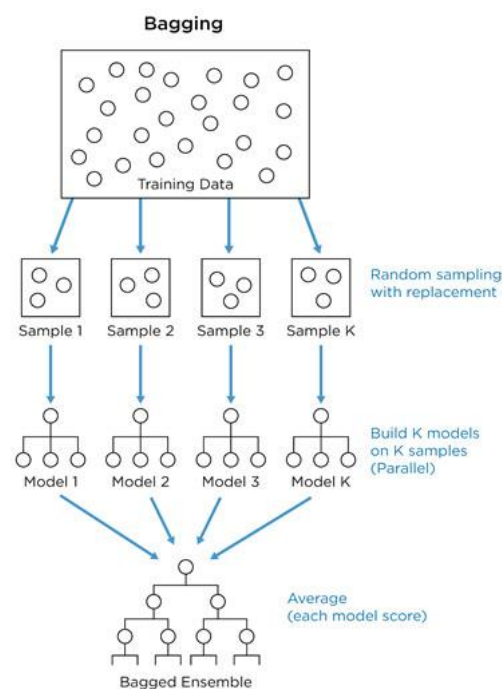
[31]Ensemble is a machine learning concept in which some models are trained using the same learning methods. Ensemble methods merge a few decision trees classifiers to come out with a better predictive performance than one decision tree classifier. The key

idea behind the ensemble model is that a bunch of weak learners come together to create a powerful learner, therefore improving the accuracy of the model.

#### 1.4.4.1 Bagging

[32] Bootstrapping is a sampling method in which subsets of observations are created based of the original dataset, with replacement. Both the size of the subsets and the original set are equal.

Bagging, also known as Bootstrap Aggregating, utilises these subsets or bags to get an overview of the distribution or complete set. The size of the bags may be less than the original set.



*Figure 10 How Bagging Works [33]*

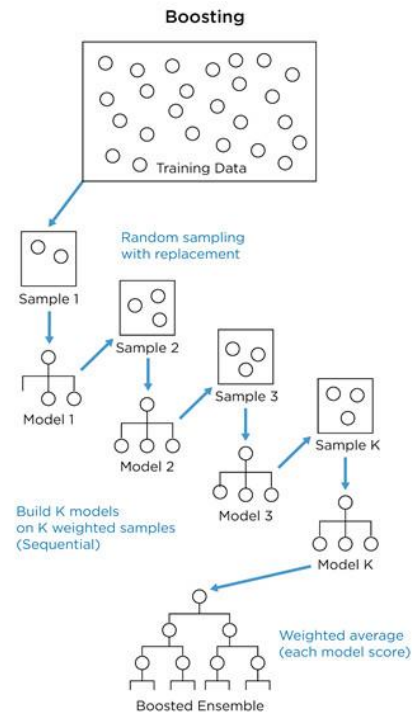
With reference to the Figure 10, these are the following steps for Bagging:

1. Multiple subsets are created from the original dataset via random sampling, samples are selected with replacement.
2. A base model (weak model) is formed on each subset.
3. The models run in parallel and are independent of each other.
4. The final predictions are based on the combinations of the predictions from all the models.

[31]Bagging is a preferred method because it reduces over-fitting of the model, handles higher dimensionality data very well and maintains accuracy for missing data. However, because the final prediction is based on the mean predictions from subset trees, it will not give precise values for the classification and regression model.

#### **1.4.4.2 Boosting**

Boosting is a sequential process, where each consecutive model attempts to amend the mistakes of the previous model. The succeeding models are dependent on the previous model [32]. Learners are learned subsequently with early learners fitting simple models to the data and then interpreting data for mistakes. Consecutive trees or random sample are fitted and at every step, the objective is to be more accurate than the previous tree. When an input is wrongly classified by a hypothesis, it has more weightage so that next hypothesis is more likely to classify it correctly. This process transforms poor learners into better performing model [31].



*Figure 11 How Boosting Works [33]*

With reference to Figure 11, the steps are as follow:

1. A subset is created from the original dataset, with every data points given equal weightage.
2. A base model is created on this subset.
3. This model is used to make predictions on the whole dataset.
4. Errors are calculated using the actual values and predicted values.
5. The observations which are incorrectly predicted, are given higher weights.
6. Another model is created, and predictions are made on the dataset. This model tries to correct the errors from the previous model
7. Similarly, multiple models are created, each correcting the errors of the previous model.

8. The final model (strong learner) is the weighted mean of all the models (weak learners)

[31]Boosting is a preferred method because it supports different loss function ('binary:logistic' for this example), works well with interactions. However, it can be prone to over-fitting and requires careful tuning of different hyper-parameters.

#### 1.4.5 Deep Learning

[34]Artificial intelligence refers to machines performing tasks that require human intelligence. It includes machine learning. Deep Learning (DL) is a subdivision of machine learning where artificial neural networks learn from large amounts of data. Like how humans learn from experience, DL algorithm would execute out a task non-stop, making minor adjustment along the way to boost the outcome. This process is called DL because the neural networks have various layers that facilitate learning. Any issue that requires "thought" to solve is a problem deep learning can learn to solve.

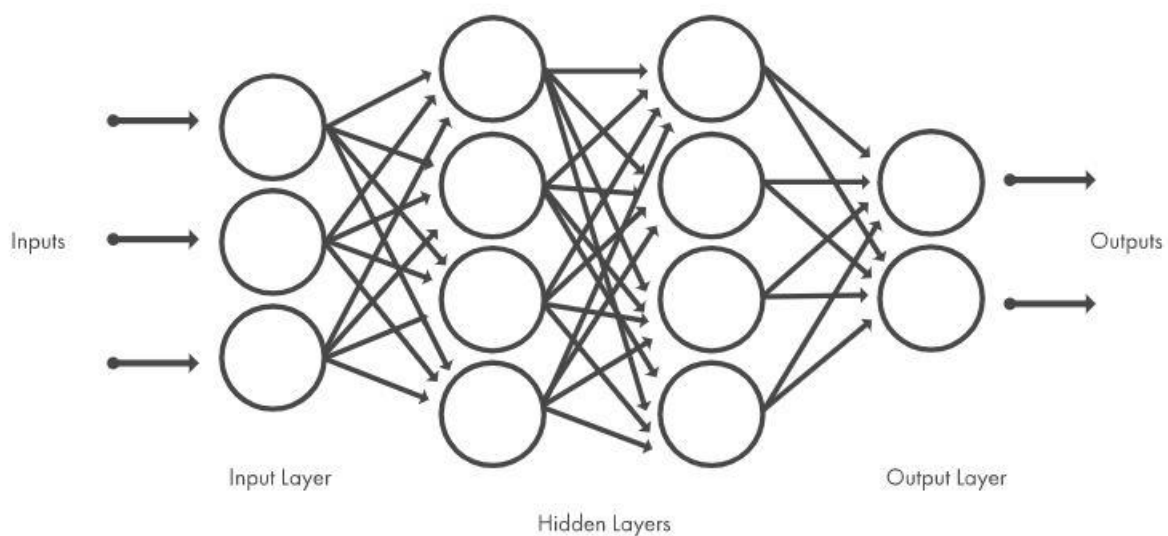
The amount of data generated daily is staggering [35]—and they are the resource that makes DL possible. Since DL algorithms need a ginormous amount of data to learn from, this rise in data creation is one reason that deep learning capabilities have increased in recent years. Besides more data creation, deep learning algorithms gain from more powerful computing power that is available today as well as the proliferation of Artificial Intelligence (AI) as a Service. AI as a Service has given smaller companies access to AI

technology and especially the AI algorithms required for deep learning without spending big in investment.

DL permits machines to solve complicated problems even when using a data set that is very diverse, unstructured, and inter-connected. The more deep learning algorithms learn, the better they perform.

#### 1.4.5.1 How Deep Learning Works

Similar to animals, our estimator AI's brain has neurons. They are represented by circles. These neurons are inter-connected.

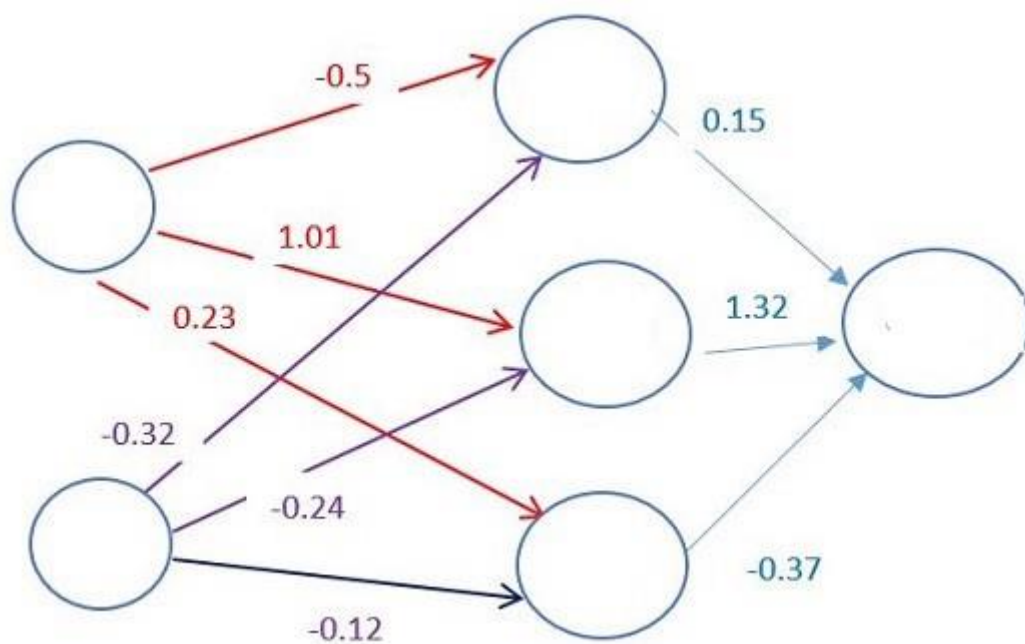


*Figure 12 Neuron Networks [36]*

The neurons are split into three layers - Input Layer, Hidden Layer(s) and Output Layer. With reference to Figure 12, the input layer receives input data. There are three

neurons in the input layer. The input layer gives the inputs to the first hidden layer. The hidden layers perform some mathematics on the inputs. One problem of creating neural networks is choosing the quantity of hidden layers, and how many neurons for each layer. The “Deep” in DL refers to having more than one hidden layer. The output layer comes back the output data.

With reference to Figure 13, each connection between neurons is linked with a weight. This weight dictates the importance of the input value. The initial weights are set randomly. When predicting the outcome, the more important factor would be given a bigger weight.



*Figure 13 Adding Weight to the Networks [37]*

Each neuron has an Activation Function. Without mathematical reasoning, these functions are hard to comprehend. In layman terms, one of its purposes is to “standardize” the output from the neuron.



The moment a set of input data has passed through all the layers of the neural network, it returns the output data through the output layer. Once the whole data set was processed, a function named Cost Function can be created to illustrate how wrong the AI's outputs were from the real outputs.

Ideally the optimal result is when our AI's outputs are the same as the data set outputs, meaning the cost function is zero. Weights could be adjusted between neurons and amend the value until the cost function is low, but that is not very efficient. Gradient Descent can solve this problem.

Gradient Descent can find the minimum of the cost function. It functions by changing the weights in small increments after each data set iteration. By computing the gradient of the cost function at a certain set of weight, the direction of the minimum can be seen. The illustration can be found at Figure 14.

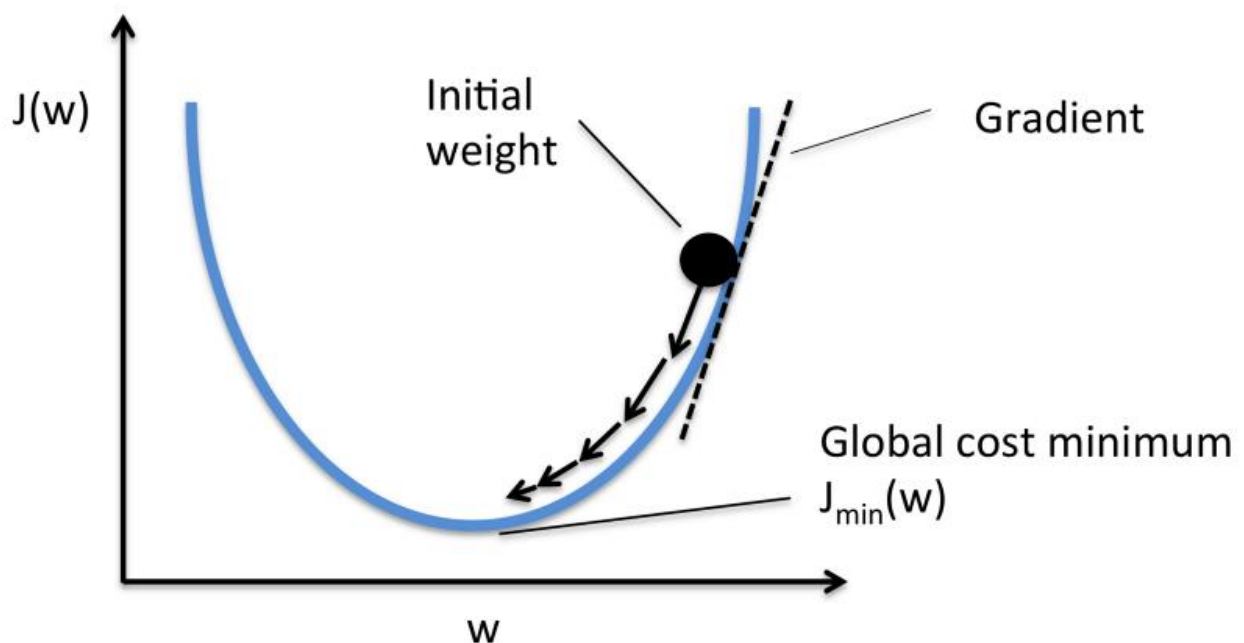


Figure 14 Gradient Descent in Cost Function [38]

To minimize the cost function, the data set must be iterated many times. Therefore, a large amount of computational power is needed. Updating the weights using gradient descent is done automatically. Once the model has been trained, it can be used to predict future.

#### **1.4.5.2 Keras**

[39]Keras is a high-level library that is built on top of Theano or TensorFlow. It provides a scikit-learn type API (written in Python) for building Neural Networks. Developers can use Keras to build neural networks swift without having to worry about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods.

The key idea behind the development of Keras is to guide experiments by fast prototyping. The ability to go from an idea to result with the least lag is instrumental to good research.

## 2 Methodology

### 2.1 The Setup

This project is inspired by one of the competitions on Kaggle. The dataset was obtained from their website [40]. The experiments were carried out using Python version 3.7.8, and the list of libraries or tools used were shown in the next few figures.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import math
from pandas_profiling import ProfileReport

import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

from textblob import TextBlob

from collections import defaultdict

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import average_precision_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

from sklearn.model_selection import train_test_split
```

*Figure 15 Libraries for Both ML and DL*

With reference to Figure 15, pandas were required to convert the data into a dataframe. Numpy and math were utilised for any calculation purposes. Matplotlib and seaborn were deployed for visualisation purposes. ProfileReport, collections were being used for finding out more information about the data. The purpose of re was to handle all the regular expressions issues. TextBlob, string and all NLTK-related libraries were applied for text cleaning purposes. All the sklearn metrics libraries were used for assessing the

models, while the train-test split was handling the segregation of the datasets into training and testing sets.

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
```

*Figure 16 Libraries for ML Only*

```
from keras.preprocessing.text import Tokenizer
from keras.utils.np_utils import to_categorical
from sklearn.preprocessing import LabelEncoder

import keras
from keras import models
from keras import layers
from keras import regularizers
import tensorflow as tf
```

*Figure 17 Libraries for DL only*

With reference to Figure 16, CountVectoriser and TF-IDF Vectoriser were involved as feature extraction, while the other 5 were the classifiers in ML to attempt to split the fake and real news as accurate as possible. On the other hand, the first 3 libraries in Figure 17 were the packages for data preparation, while the rest participated in modelling the dataset.

## 2.2 Experimental Design

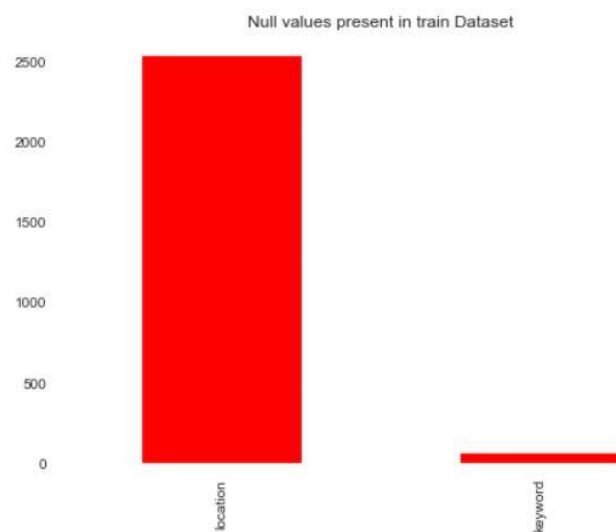
### 2.2.1 Exploratory Data Analysis (EDA)

The first step was to find out more information of the dataset that were being dealt with. The relevant files and libraries were imported before proceeding to load the dataset. The dataset was being named as “train”. The dataset consisted of 7613 rows and 5 columns. The first 5 rows were printed to observe what kind of information were to be expected.

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

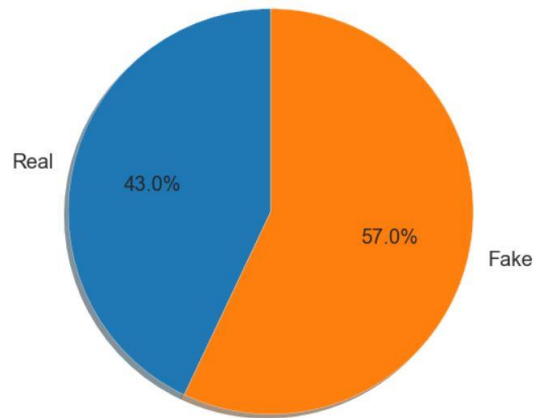
*Figure 18 The first 5 columns of dataset*

From Figure 18, it was seen that there were 5 columns, containing the ID, keyword, location of the tweet, as well as the tweet and the target, where 1 denoted a real disaster and 0 represented a fake disaster. It was worth noticing there were empty values from Figure 19. Next, the dataset was investigated deeper by finding out the total amount missing values in all column.



*Figure 19 Null Values in the Dataset*

From Figure 19, 2 columns contained missing values. These 2 columns and 'ID' would no longer involve in the experiment since they were deemed as not informative in determining real or fake disaster. Next, the volume of target was being examined.



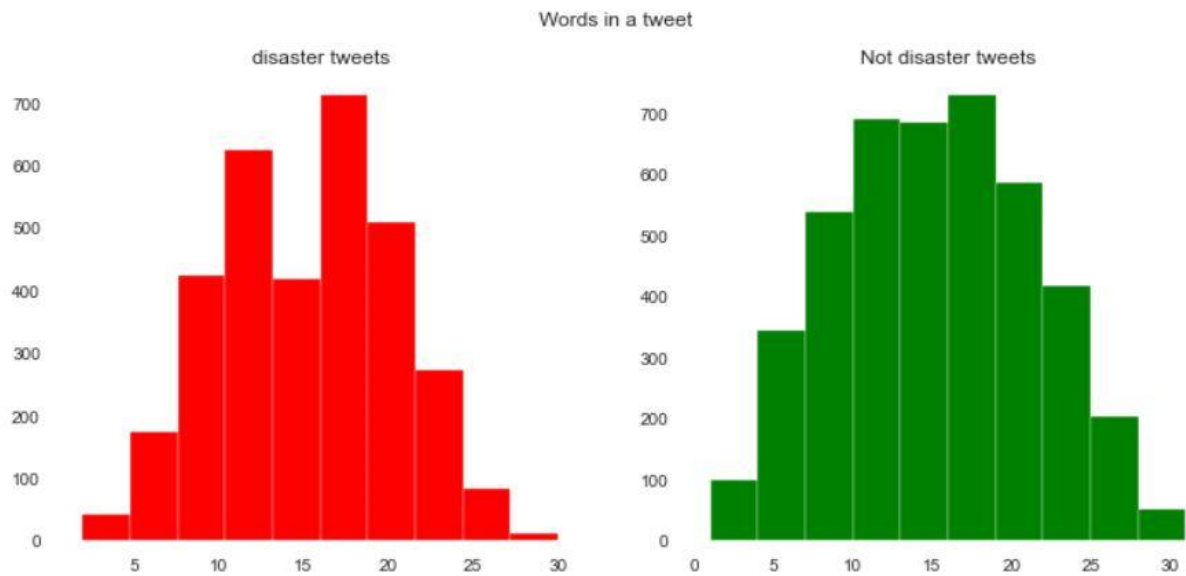
*Figure 20 Proportion of Real and Fake Disaster*

There was a total of 4342 fake disasters and 3271 real disasters. Figure 20 illustrated a pie chart the proportion of fake news against the total population which is 57%. Likewise, for the genuine disasters as compared to the whole dataset which is 43%.

### 2.3 Basic Feature Extraction

The following methods were being used to see if there were any distinct features to differentiate between a fake and a real disaster. The outcome of the results would likely affect how the cleaning of the texts in the subsequent stages.

### 2.3.1 Number of Words

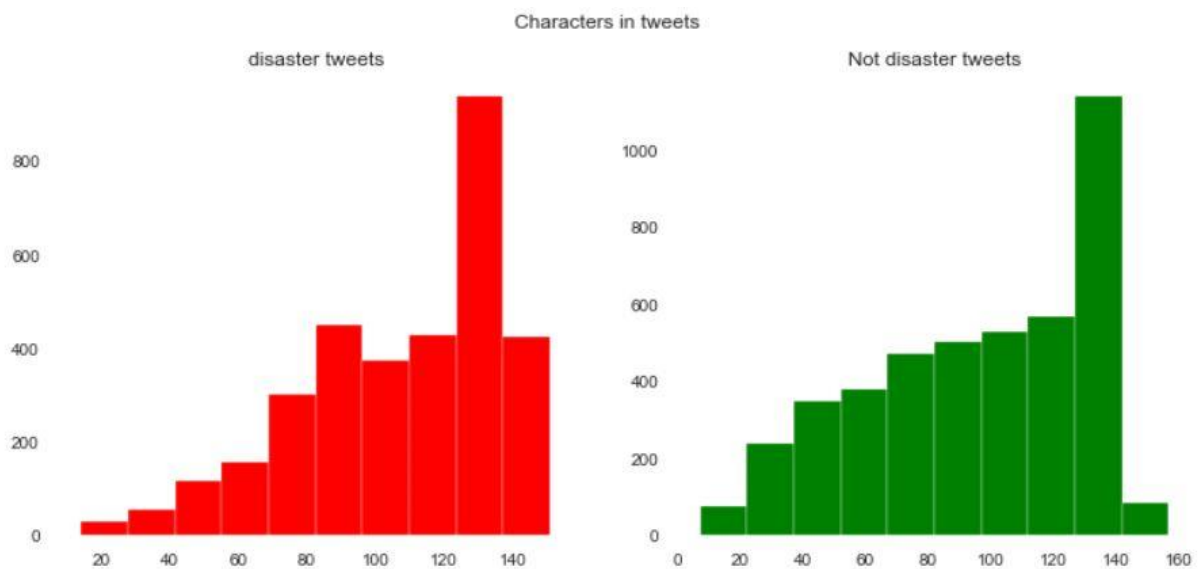


*Figure 21 Words in a Tweet*

The structure of the 2 histograms were similar as shown in Figure 21, therefore this feature could not be used to tell apart between the 2 outcomes.

### 2.3.2 Number of Characters

This component was derived from 2.3.1, the number of characters were calculated in each tweet. This was done by computing the length of the tweet.



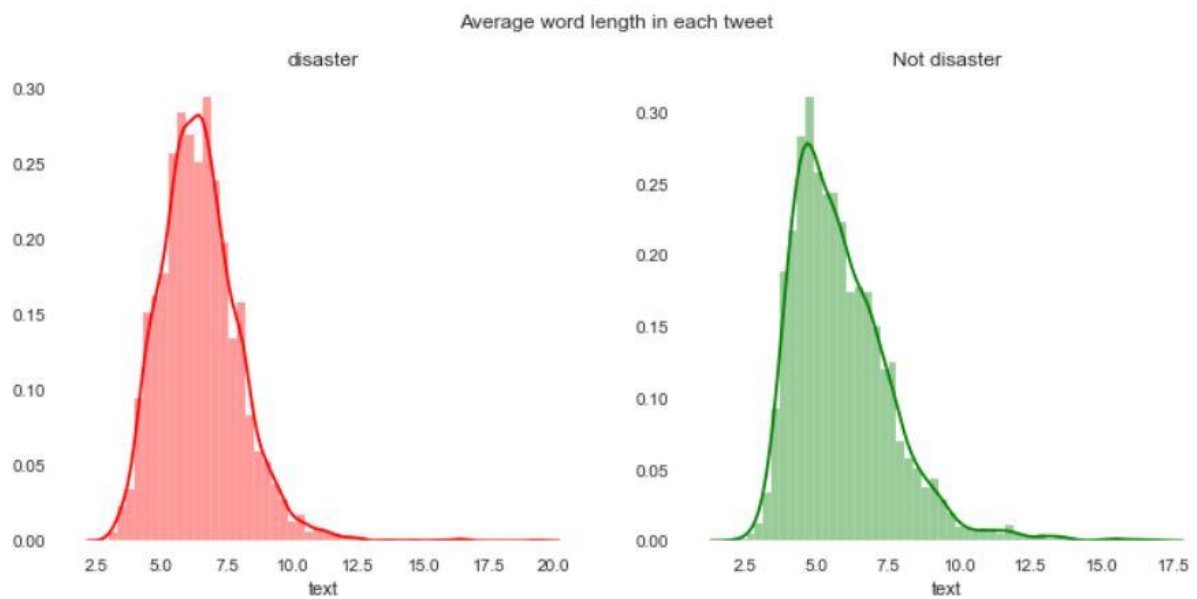
*Figure 22 Characters in Tweets*

The distribution of both histograms seemed to be almost the same according to Figure 22. 120 to 140 characters in a tweet are the most common among both. Hence this feature could not be used for telling the 2 outcomes apart.



### 2.3.3 Average Word Length

Another attribute to be chosen was to calculate the average word length of each tweet. The sum of the length of all the words were taken and divided it by the total length of the tweet.



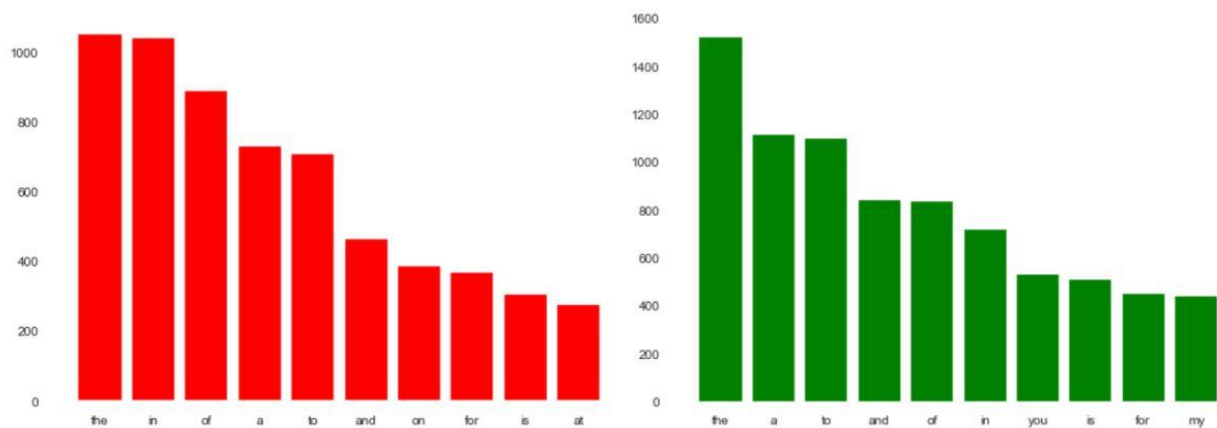
*Figure 23 Average Word Length in Each Tweet*

From Figure 23, the shapes of both histograms could not be separated. For that reason, this attribute was not applicable to identify real or fake disasters.

### 2.3.4 Number of Stopwords

Generally, when dealing with NLP problems, stopwords were being removed.

However sometimes when calculating the number of stopwords could provide some additional information.

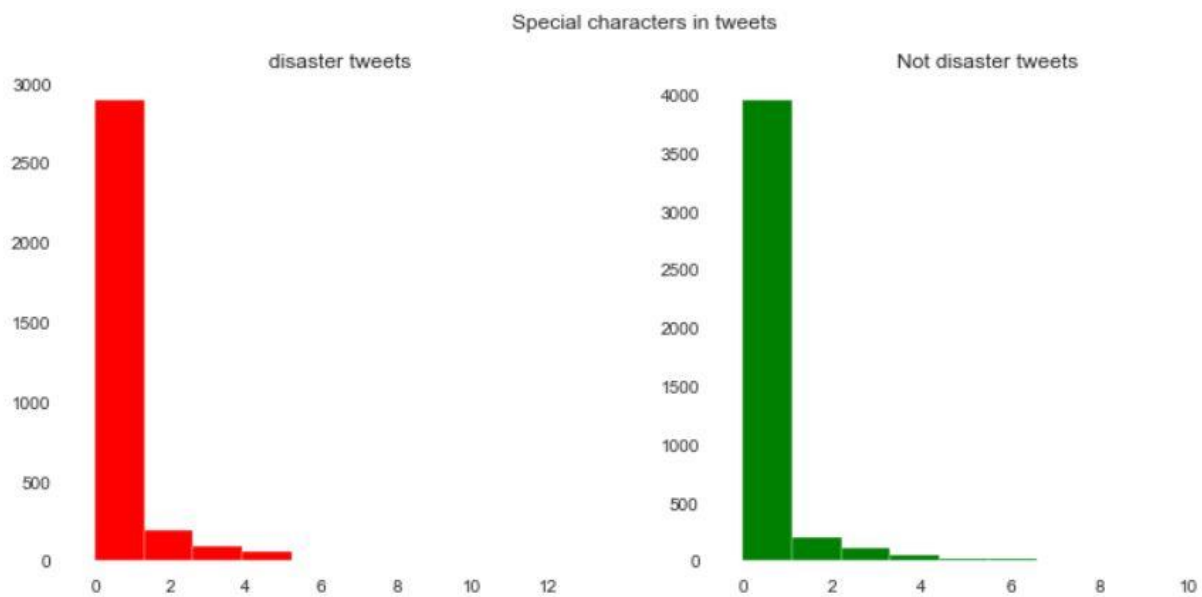


*Figure 24 Number of Stopwords in Real (Left) and Fake (Right) Disaster*

In both bar charts from Figure 24, “the” was the most used words, followed by “in” and “a” for real and fake disasters, respectively. The top 10 stopwords for both categories were mostly the same, with that being the case stopwords would not be a factor in determining real or fake disasters.

### 2.3.5 Special Character

Another characteristic which can be obtained from a tweet is calculating the number of hashtags or mentions present in it. This could help in gathering extra information from the text data. The 'starts with' function was used because hashtags (or mentions) always appear at the beginning of a word.



*Figure 25 Special Characters in Tweets*

Upon looking at Figure 25, the 2 graphs generally look alike. As a result, there is no telling whether the disaster is real or fake according to this criterion.

Punctuations were examined to find out whether it could be an element to differentiate real or fake disasters.

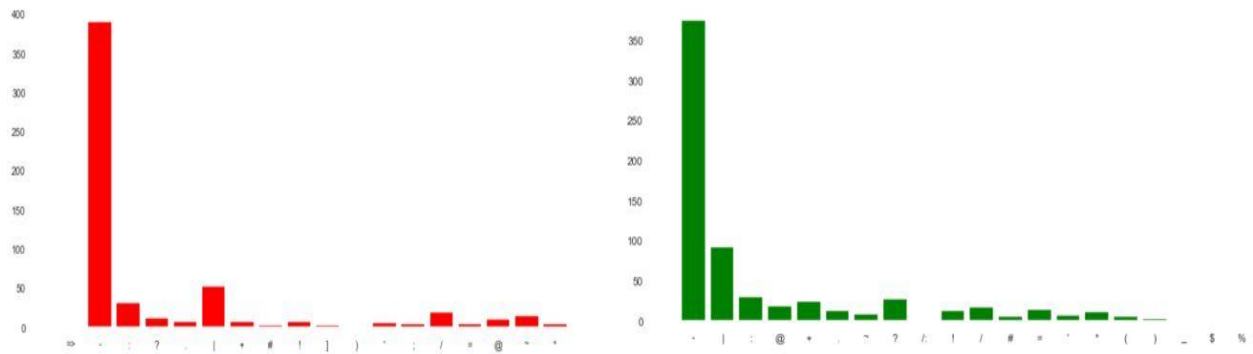
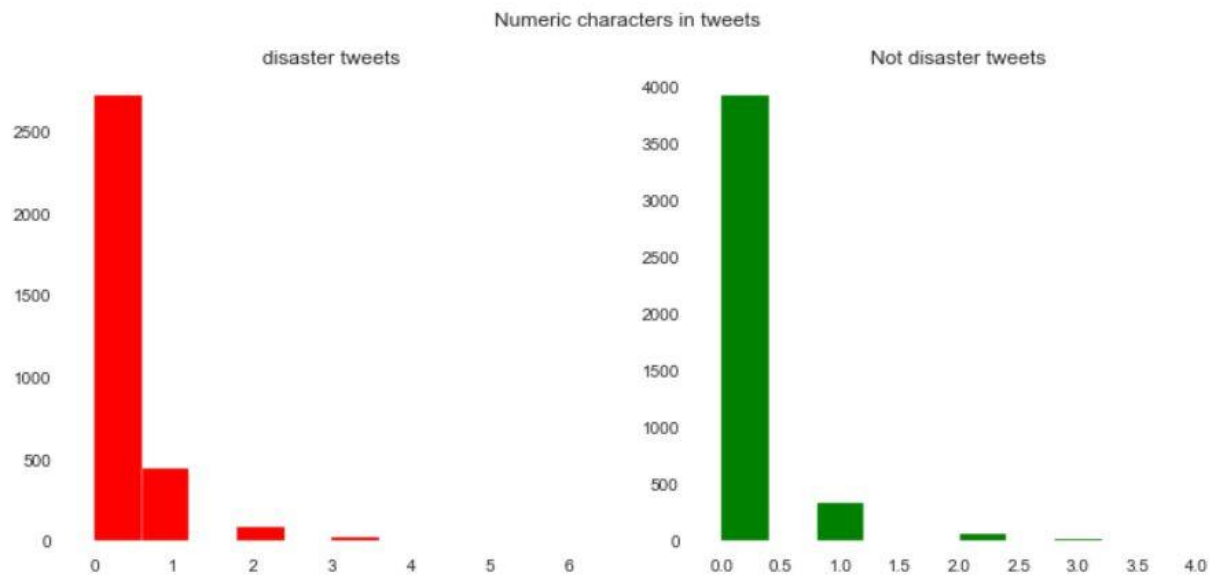


Figure 26 Punctuations in Real (Left) and Fake (Right) News

Based on the 2 charts in Figure 26, the symbol “-” was the most used punctuations for both categories. Both charts had many punctuations in common. While there were some punctuations that seemed distinctive, the count was small and considering the total sum of data, punctuations would not be able to split real and fake disasters.

### 2.3.7 Number of Numerics

Just like words, the number of numerics could be calculated to see how many were present in the text.

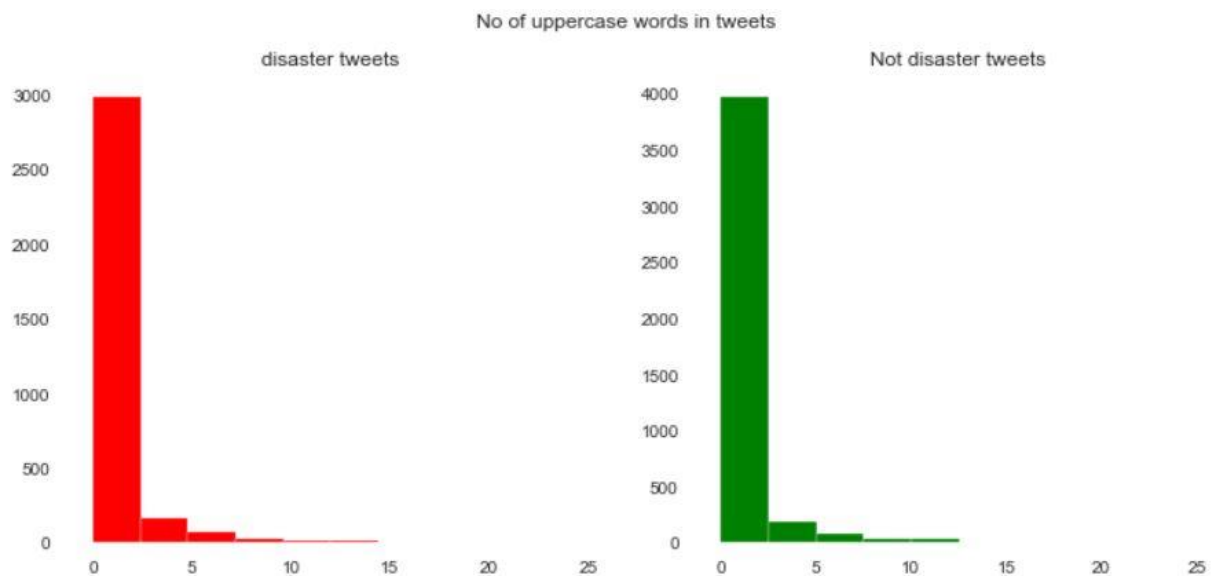


*Figure 27 Numeric Characters in Tweets*

From Figure 27 it was observed that most of the tweets did not have numeric characters. Those with numbers did not carry much weight in this context. Thus, numbers would not constitute to find real or fake disasters.

### 2.3.8 Number of Uppercase Words

Anger or rage is quite often expressed by writing in UPPERCASE words which makes this a necessary operation to identify those words.



*Figure 28 Uppercase Words in Tweets*

According to Figure 28, both histograms shared very similar formation. Therefore, using this feature would not aid in finding out real or fake disasters.

## 2.4 Cleaning Text

Earlier, basic features have been separated from text data. Before diving into text and feature extraction, the first step should be cleaning the data to obtain better features by doing some of the basic pre-processing steps on the training data.

### 2.4.1 Removal of http

To begin with text cleaning, websites and linked were removed from the analysis because they were redundant.

```
def clean_text(text):  
    text = re.sub(r'https://t.co\S+\s*', '', text) # remove URLs http://t.co/lbwejojpxOd  
    text = re.sub(r'http://t.co\S+\s*', '', text)  
    return text
```

*Code 1 Remove HTTP*

### 2.4.2 Remove Mentions

The account name was irrelevant to what was trying to be analysed, so they have been erased as well.

```
def clean_text1(text):  
    text = re.sub(r'@\S+', '', text) # remove mentions  
    return text
```

*Code 2 Remove Mentions*

### 2.4.3 Remove RT

It was unnecessary to know who was being tagged so retweets or RT would be eliminated. Note that “cc” cannot be removed because of words such as “accident”.

```
def clean_text2(text):  
    text = re.sub(r'RT|^cc', '', text) # remove RT  
    return text
```

*Code 3 Remove RT*

#### 2.4.4 Remove Extra Whitespace and Special Symbols

Extra whitespace and special symbols were being removed because they did not have a significant impact on the outcome.

```
def clean_text3(text):  
    text = re.sub(r'\s+$', '', text)  
    text = re.sub(r'^s+', '', text)  
    text = re.sub(r'^a-zA-Z\s+', '', text) #remove special symbols  
    return text
```

*Code 4 Removal of Extra Whitespace and Special Symbols*

#### 2.4.5 Remove Numbers

As established from 2.3.7, numbers had no influence in determining real or fake disasters, therefore they were being removed.

```
def clean_text4(text):  
    text = re.sub(r'\w*\d\w*', '', text)  
    return text
```

*Code 5 Remove Numbers*

#### 2.4.6 Expand Contractions

Contractions are the shortened versions of words like “don’t” for “do not” and “how’ll” for “how will”. These are used to reduce the speaking and writing time of words. For a better analysis of the texts, these contractions are to be expanded. The dictionary list can be found in Appendix 1.



```

# Regular expression for finding contractions
contractions_re=re.compile('%s' % '|'.join(contractions_dict.keys()))

# Function for expanding contractions
def expand_contractions(text,contractions_dict=contractions_dict):
    def replace(match):
        return contractions_dict[match.group(0)]
    return contractions_re.sub(replace, text)

# Expanding Contractions in the reviews
train['clean'] = train['clean'].apply(lambda x:expand_contractions(x))

```

*Code 6 Expanding Contractions*

#### 2.4.7 Lowercase Words

In NLP, models treat words like “Goat” and “goat” differently, even if they are the same. Therefore, to overcome this problem, words are converted to be lowercase using the code in Code 7.

```

train['clean'] = train['clean'].apply(lambda x: " ".join(x.lower() for x in x.split()))

```

*Code 7 Lowercase Words*

#### 2.4.8 Remove Stopwords

Back in 2.3.4, it has been established that stopwords did not much impact in finding real or fake news. Thus, stopwords would be removed.

```

train['clean'] = train['clean'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))

```

*Code 8 Remove Stopwords*

#### 2.4.9 Remove Punctuations

After the conclusion at 2.3.6, punctuations were not proven to be able split real and fake news. As a result, punctuations were removed.

```
train['clean'] = train['clean'].str.replace('[^\w\s]','')
```

#### *Code 9 Removing Punctuations*

##### 2.4.10 Remove Common Words

Previously, stopwords were removed because they were commonly occurring words in a general sense. Other commonly occurring words from our text data could be removed as well. The 10 most frequently occurring words in the text data were checked then deciding to remove or retain.

```
freq = pd.Series(' '.join(train['clean']).split()).value_counts()[:10]  
freq
```

```
like      345  
im        306  
amp       300  
fire      252  
get       229  
new       226  
via       220  
dont      208  
news      196  
people    195  
dtype: int64
```

#### *Code 10 Top 10 Words in the Text*

Other than “fire”, those words would be removed because they were likely to be uninformative.

```

freq = list(freq.index)

print(freq)

['like', 'im', 'amp', 'fire', 'get', 'new', 'via', 'dont', 'news', 'people']

freq.remove('fire')

print(freq)

['like', 'im', 'amp', 'get', 'new', 'via', 'dont', 'news', 'people']

train['clean'] = train['clean'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))

```

### Code 11 Remove Common Words

#### 2.4.11 Remove Rare Words

At the other end of the spectrum, words that only appeared once in the text were likely to have minimal impact in differentiating real and fake news. They were removed as well.

```

s = pd.Series(' '.join(train['clean']).split()).value_counts()

freq1 = s[s == 1].index
freq1

Index(['deceased', 'francisunderwood', 'jakartapost', 'remorseless', 'div',
      'dealer', 'lining', 'brutal', 'zach', 'hilda',
      ...,
      'trade', 'overturn', 'resemblance', 'bloom', 'sinceg', 'mba', 'muse',
      'alternative', 'epidemics', 'flakes'],
      dtype='object', length=8402)

train['clean'] = train['clean'].apply(lambda x: " ".join(x for x in x.split() if x not in freq1))

```

### Code 12 Remove Rare Words

#### 2.4.12 Spelling Correction

It is common to see tweets with a ton of spelling mistakes. Users' timelines are often filled with hasty sent tweets that are barely legible at times. In that regard, spelling

correction is a useful pre-processing step because this also will help to reduce multiple copies of words. For example, “Analytics” and “analytcs” will be treated as different words even if they are used in the same sense.

```
train['clean'].apply(lambda x: str(TextBlob(x).correct()))
```

#### *Code 13 Spelling Correction*

### 2.4.13 Lemmatisation and Tokenisation

Tokenisation refers to dividing the text into a sequence of words or sentences, while Lemmatisation is to reduce the word into its root form. At the last stage of text cleaning, all the texts would undergo these 2 processes in Code 14. In this case, “v” or verbs were chosen to be lemmatised.

```
# Init the Wordnet Lemmatizer

w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_verb_text(text):
    return [lemmatizer.lemmatize(w, 'v') for w in w_tokenizer.tokenize(text)]

train['clean'] = train['clean'].apply(lemmatize_verb_text)
```

#### *Code 14 Lemmatisation and Tokenisation*

The last step was to convert all the text into string for analysis.

```
train['clean'] = train['clean'].apply(lambda x: " ".join([str(word) for word in x]))
```

#### *Code 15 Converting to String*

To ensure that the all the cleaning worked correctly, a sample size of 20 was chosen at random and being shown in Figure 29 .

	text	clean
1148	Japan Marks 70th Anniversary of Hiroshima Atomic Bombing <a href="http://t.co/jzgxwRgFQg">http://t.co/jzgxwRgFQg</a>	japan mark th anniversary hiroshima atomic bomb
5201	WACKOS like #MicheleBachman predict the WORLD will SOON be OBLITERATED by a burning firey INFERNO but cant accept #GlobalWarming!! HELLO!!!	michelebachman predict world soon obliterate burn firey inferno cant accept globalwarming hello
5485	Wired: Reddit Will Now Quarantine Offensive Content - Reddit co-founder and CEO Steve Huffman has unveiled more sp... <a href="http://t.co/aByHRgsS1s">http://t.co/aByHRgsS1s</a>	wire reddit quarantine offensive content reddit cofounder ceo steve huffman unveil sp
1568	ok peace I hope I fall off a cliff along with my dignity	ok peace hope fall cliff along dignity
7592	Heat wave warning aa? Ayyo dei. Just when I plan to visit friends after a year.	heat wave warn aa plan visit friends year
4677	Beyond all bounds; till inundation rise	beyond bound till inundation rise
1124	I liked a @YouTube video <a href="http://t.co/FX7uZZXtE4">http://t.co/FX7uZZXtE4</a> Benedict Cumberbatch Gets Video Bombed	like video get video bomb
2683	Ignition Knock (Detonation) Sensor-Senso Standard fits 03-08 Mazda 6 3.0L-V6 <a href="http://t.co/c8UXkizwM6">http://t.co/c8UXkizwM6</a> <a href="http://t.co/SNxgH9R16u">http://t.co/SNxgH9R16u</a>	ignition knock detonation sensorsenso standard fit lv
1162	@ameenshaikh3 by ur. logic if bridge didnt collapse then second train engine should cross bridge then @sanjaynirupam @sureshprabhu	logic bridge didnt collapse second train engine cross bridge
5046	First time getting into #gbbo2015 and physically gasped at the cake 'mudslide' incident already way too emotionally invested...	first time get gbbo cake mudslide incident already way emotionally
294	My niece just asked me 'would you be scared if there was an apocalypse here?' ????	niece ask would scar apocalypse
5106	Bombing #Iran would result in a never-ending game of #nuclear whack-a-mole. Here's why: <a href="http://t.co/6exS23MUy3">http://t.co/6exS23MUy3</a> <a href="http://t.co/l9iDheROtj">http://t.co/l9iDheROtj</a>	bomb iran would result neverending game nuclear heres
191	<a href="http://t.co/FCqmKFfllW">http://t.co/FCqmKFfllW</a> Twelve feared killed in Pakistani air ambulance helicopter crash <a href="http://t.co/vAyaYmbNgu">http://t.co/vAyaYmbNgu</a>	twelve fear kill pakistani air ambulance helicopter crash
69	Accident on I-24 W #NashvilleTraffic. Traffic moving 8m slower than usual. <a href="https://t.co/0GHk693EgJ">https://t.co/0GHk693EgJ</a>	accident w traffic move slower usual
2648	Bomb squad set to detonate backpack Antioch Tenn. theater gunman had on him officials say - @Tennessean <a href="http://t.co/eb74lclWn">http://t.co/eb74lclWn</a>	bomb squad set detonate backpack antioch theater gunman officials say
3604	11-Year-Old Boy Charged With Manslaughter of Toddler: Report: An 11-year-old boy has been charged with manslaughter over the fatal sh...	yearold boy charge manslaughter toddler report yearold boy charge manslaughter fatal sh
4184	Seeing Hazard without the beard like... <a href="http://t.co/IPtZWVNXXP">http://t.co/IPtZWVNXXP</a>	see hazard without beard
6188	connor franta: damn sirens I hope everyone is okay. \nndan howell: can you PLEASE get MURDERED on ANOTHER STREET	damn sirens hope everyone okay dan please murder another street
1719	My @MLG and food worlds have collided in this @ijustine salmon video. #simple #Alaskaseafood #askforalaska <a href="https://t.co/2SnyGHaiVs">https://t.co/2SnyGHaiVs</a>	food worlds collide salmon video simple
4151	I don't pray harm on members of ISIS.I pray they experience the life-rebooting love of God & become 'Paul's' in Gods mind-blowing final Act	pray harm members pray experience love god become pauls gods final act

Figure 29 Samples after Cleaning

## 2.5 WordCloud

In the previous section, text cleaning was conducted. Using WordCloud can help to visualise words associated with real and fake disasters in Figure 30 and Figure 31 respectively.



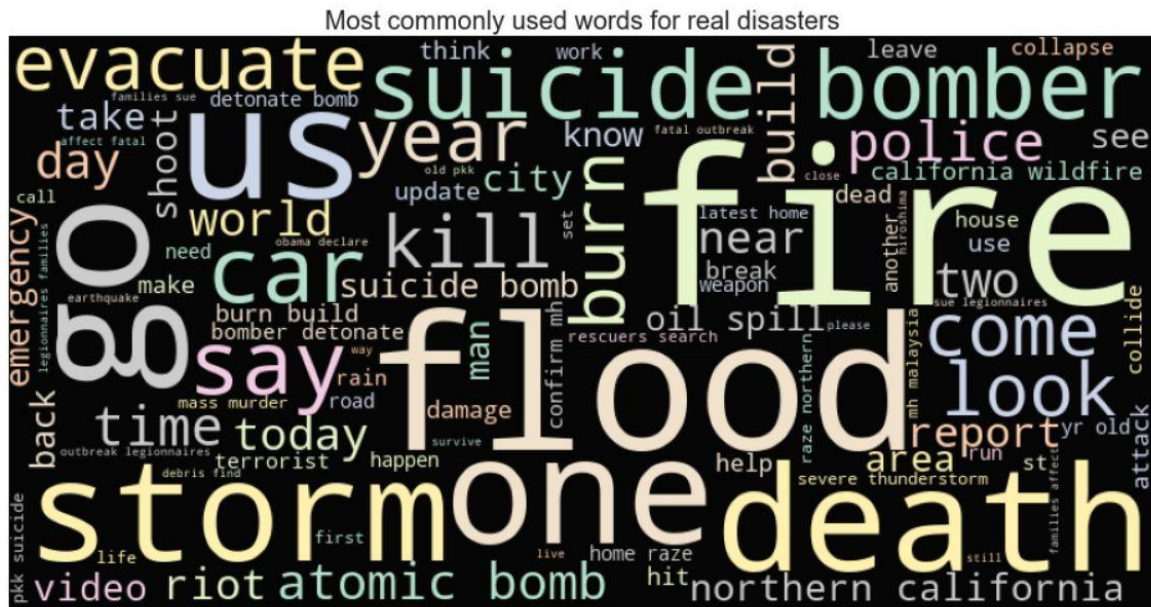


Figure 30 Most Commonly Used Words for Real Disasters

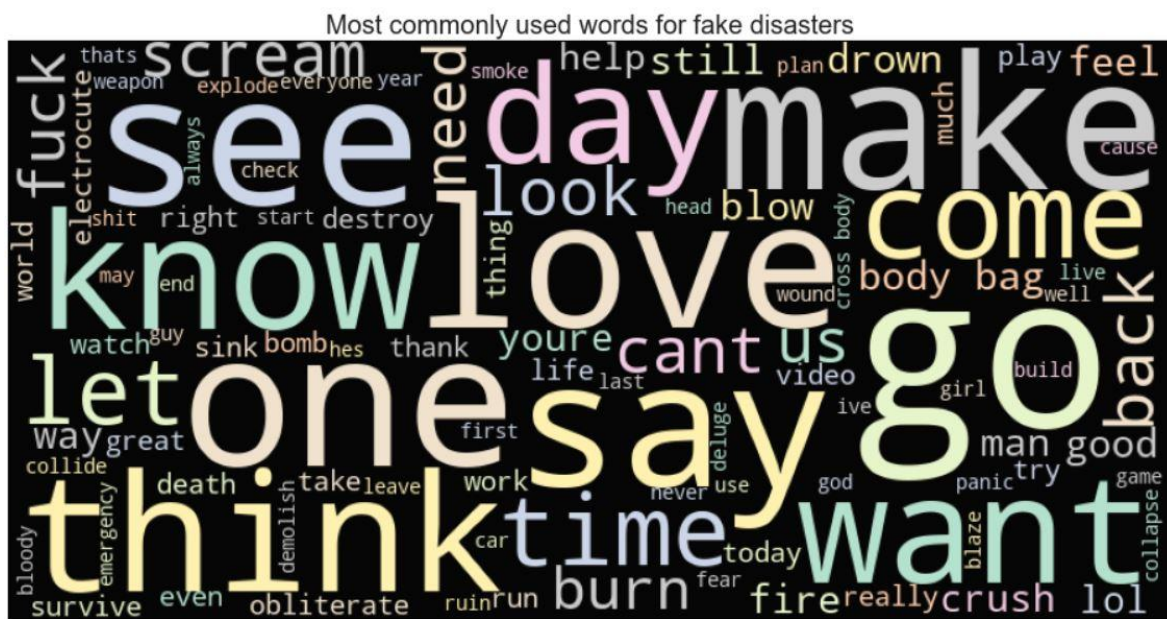


Figure 31 Most Commonly Used Words for Fake Disasters

There were words that appeared in both figures, they could be the reason if the classifiers categorised the labels wrongly at the latter stages.

## 2.6 Feature Engineering

Before doing the feature engineering, feature labels were being defined in Code 16. Stratify was used to ensure the train and test sets maintain at the 43:57 ratio between real and fake disasters shown from Figure 20.

```
# Features and Labels
X = train['clean']
y = train['target']

# split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42, stratify = y)
```

*Code 16 Feature Labels and Train Test Split*

### 2.6.1 Machine Learning

#### 2.6.1.1 CountVectoriser

```
# create a matrix of word counts from the text
count_vect = CountVectorizer(token_pattern = r'\w{1,}')
```

```
# do the actual counting
A = count_vect.fit_transform(X_train, y_train)
```

```
# Transform documents to document-term matrix.
X_train_count = count_vect.transform(X_train)
X_test_count = count_vect.transform(X_test)
```

*Code 17 CountVectoriser as Feature*

### 2.6.1.2 TF-IDF

#### 2.6.1.2.1 Word Level

```
# word level tf-idf
tfidf_vect = TfidfVectorizer(analyzer = 'word',
                             token_pattern = r'\w{1,}',
                             max_features = 5000)

print(tfidf_vect)

B1 = tfidf_vect.fit(X_train, y_train)
X_train_tfidf = tfidf_vect.transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

*Code 18 TF-IDF Word Level as Features*

#### 2.6.1.2.2 N-Gram Level

```
# ngram level tf-idf
tfidf_vect_ngram = TfidfVectorizer(analyzer = 'word',
                                    token_pattern = r'\w{1,}',
                                    ngram_range = (2, 3),
                                    max_features = 5000)

print(tfidf_vect_ngram)

B2 = tfidf_vect_ngram.fit(X_train, y_train)
X_train_tfidf_ngram = tfidf_vect_ngram.transform(X_train)
X_test_tfidf_ngram = tfidf_vect_ngram.transform(X_test)
```

*Code 19 TF-IDF N-Gram as Features*

### 2.6.2 Keras

To begin with, some parameters were defined that would be used throughout the experiment. The batch size is the number of samples processed before the model is updated. The number of epochs is the number of complete passes (iteration) through the training dataset.



```

NB_WORDS = 10000 # Parameter indicating the number of words we'll put in the dictionary to create word vector
VAL_SIZE = 1000 # Size of the validation set
NB_START_EPOCHS = 20 # Number of epochs we usually start to train with
BATCH_SIZE = 512 # Size of the batches used in the mini-batch gradient descent

```

### Code 20 Setting Parameters

Next, the words being put into the dictionary to create word vectors were tokenised.

```

tk = Tokenizer(num_words=NB_WORDS,
               filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
               lower=True,
               split=" ")
tk.fit_on_texts(X_train)

```

### Code 21 Tokenise the Words in Dictionary

Subsequently they were converted into sequences.

```

X_train_seq = tk.texts_to_sequences(X_train)
X_test_seq = tk.texts_to_sequences(X_test)
print("{}" is converted into {}'.format(X_train[1], X_train_seq[1]))

```

"forest fire near la canada" is converted into [1333, 2394, 4538, 955, 585, 777, 378]

### Code 22 Convert to Sequences

One hot encoding is a binary indicator of whether a feature or word exists in the training documents. It is used in this scenario instead of displaying the positional index of a particular word, it indicates 1 or 0 if the word is present in the document.

```

def one_hot_seq(seqs, nb_features = NB_WORDS):
    ohs = np.zeros((len(seqs), nb_features))
    for i, s in enumerate(seqs):
        ohs[i, s] = 1.
    return ohs

```

### Code 23 One Hot Encoding

Using Code 23, the words were converted to one hot encoding as shown in Code 24 below.

```

X_train_oh = one_hot_seq(X_train_seq)
X_test_oh = one_hot_seq(X_test_seq)

print("{} is converted into {}".format(X_train_seq[1], X_train_oh[1]))
print('For this example we have {} features with a value of 1.'.format(X_train_oh[1].sum()))

"[1333, 2394, 4538, 955, 585, 777, 378]" is converted into [0. 0. 0. ... 0. 0. 0.]
For this example we have 7.0 features with a value of 1.

```

#### *Code 24 Convert to One Hot Encoding*

Following one hot encoding, label encoding was performed to the y variable or the outcome as shown in Code 25.

```

le = LabelEncoder()
y_train_le = le.fit_transform(y_train)
y_test_le = le.transform(y_test)
y_train_oh = to_categorical(y_train_le)
y_test_oh = to_categorical(y_test_le)
print("{} is converted into {}".format(y_train[1], y_train_le[1]))
print("{} is converted into {}".format(y_train_le[1], y_train_oh[1]))

"1" is converted into 0
"0" is converted into [1. 0.]

```

#### *Code 25 Label Encoding*

To complete the process, train test split was performed.

```

X_train_rest, X_valid, y_train_rest, y_valid = train_test_split(X_train_oh, y_train_oh, test_size=0.2, random_state=42)

```

#### *Code 26 Train Test Split after OHE and LE*

## 2.7 Text Classification

### 2.7.1 Naïve Bayes

For this machine learning method, Naïve Bayes would be implemented along with CountVectoriser and TF-IDF mentioned in 2.6.1.1 and 2.6.1.2 respectively.

```

model_5_A = MultinomialNB()

# fit the training dataset on the classifier

A1 = model_5_A.fit(X_train_count, y_train)

# predict the labels on validation dataset

predictions_A1 = model_5_A.predict(X_test_count)
predict_proba_A1 = model_5_A.predict_proba(X_test_count)[: ,1]

```

#### Code 27 Naive Bayes CountVectoriser

```

A2 = model_5_A.fit(X_train_tfidf, y_train)

predictions_A2 = model_5_A.predict(X_test_tfidf)
predict_proba_A2 = model_5_A.predict_proba(X_test_tfidf)[: ,1]

```

#### Code 28 Naive Bayes TF-IDF Word Level

```

A3 = model_5_A.fit(X_train_tfidf_ngram, y_train)

predictions_A3 = model_5_A.predict(X_test_tfidf_ngram)
predict_proba_A3 = model_5_A.predict_proba(X_test_tfidf_ngram)[: ,1]

```

#### Code 29 Naive Bayes TF-IDF N-Gram

### 2.7.2 Linear Classifier

```

model_5_B = LogisticRegression(solver = 'lbfgs', max_iter = 100)

# fit the training dataset on the classifier

B1 = model_5_B.fit(X_train_count, y_train)

# predict the labels on validation dataset

predictions_B1 = model_5_B.predict(X_test_count)
predict_proba_B1 = model_5_B.predict_proba(X_test_count)[: ,1]

```

#### Code 30 Linear CountVectoriser

```
B2 = model_5_B.fit(X_train_tfidf, y_train)

predictions_B2 = model_5_B.predict(X_test_tfidf)
predict_proba_B2 = model_5_B.predict_proba(X_test_tfidf)[: ,1]
```

*Code 31 Linear TF-IDF Word Level*

```
B3 = model_5_B.fit(X_train_tfidf_ngram, y_train)

predictions_B3 = model_5_B.predict(X_test_tfidf_ngram)
predict_proba_B3 = model_5_B.predict_proba(X_test_tfidf_ngram)[: ,1]
```

*Code 32 Linear TF-IDF N-Gram*

### 2.7.3 Support Vector Classifier

```
# define model

model_5_C = SVC(kernel='linear', probability=True)

# fit the training dataset on the classifier

C1 = model_5_C.fit(X_train_count, y_train)

# predict the labels on validation dataset

predictions_C1 = model_5_C.predict(X_test_count)
predict_proba_C1 = model_5_C.predict_proba(X_test_count)[: ,1]
```

*Code 33 SVC CountVectoriser*

```
C2 = model_5_C.fit(X_train_tfidf, y_train)

predictions_C2 = model_5_C.predict(X_test_tfidf)
predict_proba_C2 = model_5_C.predict_proba(X_test_tfidf)[: ,1]
```

*Code 34 SVC TF-IDF Word Level*



```

C3 = model_5_C.fit(X_train_tfidf_ngram, y_train)

predictions_C3 = model_5_C.predict(X_test_tfidf_ngram)
predict_proba_C3 = model_5_C.predict_proba(X_test_tfidf_ngram)[: ,1]

```

*Code 35 SVC TF-IDF N-Gram Level*

#### 2.7.4 Bagging Models (Random Forest)

```

# define model

model_5_D = RandomForestClassifier(n_estimators = 100)

# fit the training dataset on the classifier

D1 = model_5_D.fit(X_train_count, y_train)

# predict the labels on validation dataset

predictions_D1 = model_5_D.predict(X_test_count)
predict_proba_D1 = model_5_D.predict_proba(X_test_count)[: ,1]

```

*Code 36 Random Forest CountVectoriser*

```

D2 = model_5_D.fit(X_train_tfidf, y_train)

predictions_D2 = model_5_D.predict(X_test_tfidf)
predict_proba_D2 = model_5_D.predict_proba(X_test_tfidf)[: ,1]

```

*Code 37 Random Forest TF-IDF Word Level*

```

D3 = model_5_D.fit(X_train_tfidf_ngram, y_train)

predictions_D3 = model_5_D.predict(X_test_tfidf_ngram)
predict_proba_D3 = model_5_D.predict_proba(X_test_tfidf_ngram)[: ,1]

```

*Code 38 Random Forest TF-IDF N-Gram*

### 2.7.5 Boosting Models (Gradient Descent)

```
# define model
model_5_E = GradientBoostingClassifier()

# fit the training dataset on the classifier
E1 = model_5_E.fit(X_train_count, y_train)

# predict the labels on validation dataset
predictions_E1 = model_5_E.predict(X_test_count)
predict_proba_E1 = model_5_E.predict_proba(X_test_count)[: ,1]
```

*Code 39 Gradient Descent CountVectoriser*

```
E2 = model_5_E.fit(X_train_tfidf, y_train)

predictions_E2 = model_5_E.predict(X_test_tfidf)
predict_proba_E2 = model_5_E.predict_proba(X_test_tfidf)[: ,1]
```

*Code 40 Gradient Descent TF-IDF Word Level*

```
E3 = model_5_E.fit(X_train_tfidf_ngram, y_train)

predictions_E3 = model_5_E.predict(X_test_tfidf_ngram)
predict_proba_E3 = model_5_E.predict_proba(X_test_tfidf_ngram)[: ,1]
```

*Code 41 Gradient Descent TF-IDF N-Gram*

### 2.7.6 Dense Feedforward Classifier (Keras)

The model used is a sequential model with 2 intermediate hidden layers. In the input layer, there are 64 neurons (what is described as a perceptron). As there are 10 000 words, total number of parameters in the first layer are  $64 \times 10000 + 64$  (for constants) which is 640064.

In the 2nd layer, this takes the output from the 1st layer (64 outputs from the 64 neurons) Total number of parameters are  $64 \times 64 + 64$  which is 4160.

There are 2 outputs (shape) from the last output layer which correspond to probabilities of the 2 classes - real and fake. Total number of parameters are  $64 \times 2 + 2$  which is 130.

### 2.7.6.1 Baseline Model

```
base_model = models.Sequential()
base_model.add(layers.Dense(64, activation='relu', input_shape=(NB_WORDS,)))
base_model.add(layers.Dense(64, activation='relu'))
base_model.add(layers.Dense(2, activation='softmax'))
base_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	640064
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 2)	130

Total params: 644,354  
Trainable params: 644,354  
Non-trainable params: 0

*Code 42 Baseline Model*

### 2.7.6.2 Reduced Model

Reduced model refers to the removal of several neurons and hidden layers.

```
reduced_model = models.Sequential()
reduced_model.add(layers.Dense(32, activation='relu', input_shape=(NB_WORDS,)))
reduced_model.add(layers.Dense(2, activation='softmax'))
reduced_model.summary()

reduced_history = deep_model(reduced_model)
```

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 32)	320032
dense_4 (Dense)	(None, 2)	66

=====  
Total params: 320,098  
Trainable params: 320,098  
Non-trainable params: 0

*Code 43 Reduced Model*



### 2.7.6.3 Regularised Model

Another method to reduce over-fitting is adding a penalty through a regulariser, which relays the weights in each layer to the final loss function.

```
reg_model = models.Sequential()
reg_model.add(layers.Dense(64, kernel_regularizer=regularizers.l2(0.001), activation='relu', input_shape=(NB_WORDS,)))
reg_model.add(layers.Dense(64, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
reg_model.add(layers.Dense(2, activation='softmax'))
reg_model.summary()

reg_history = deep_model(reg_model)
compare_loss_with_baseline(reg_history, 'Regularized Model')
```

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 64)	640064
dense_6 (Dense)	(None, 64)	4160
dense_7 (Dense)	(None, 2)	130

Total params: 644,354  
Trainable params: 644,354  
Non-trainable params: 0

*Code 44 Regularised Model*

#### 2.7.6.4 Dropout Model

A third way to reduce over-fitting is by randomly dropping data-sets for training. In this case, an additional hidden layer is used to drop some of the intermediate training data.

```
drop_model = models.Sequential()
drop_model.add(layers.Dense(64, activation='relu', input_shape=(NB_WORDS,)))
drop_model.add(layers.Dropout(0.5))
drop_model.add(layers.Dense(64, activation='relu'))
drop_model.add(layers.Dropout(0.5))
drop_model.add(layers.Dense(2, activation='softmax'))
drop_model.summary()
```

```
drop_history = deep_model(drop_model)
```

```
compare_loss_with_baseline(drop_history, 'Dropout Model')
```

Model: "sequential\_3"

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 64)	640064
dropout (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 2)	130

```
Total params: 644,354
Trainable params: 644,354
Non-trainable params: 0
```

*Code 45 Dropout Model*

# Appendix

## Appendix 1 Contractions Dictionary

```
# Dictionary of English Contractions
contractions_dict = { "ain't": "are not", "'s": "is", "aren't": "are not",
    "can't": "cannot", "can't've": "cannot have",
    "'cause": "because", "could've": "could have", "couldn't": "could not",
    "couldn't've": "could not have", "didn't": "did not", "doesn't": "does not",
    "don't": "do not", "hadn't": "had not", "hadn't've": "had not have",
    "hasn't": "has not", "haven't": "have not", "he'd": "he would",
    "he'd've": "he would have", "he'll": "he will", "he'll've": "he will have",
    "how'd": "how did", "how'd'y": "how do you", "how'll": "how will",
    "I'd": "I would", "I'd've": "I would have", "I'll": "I will",
    "I'll've": "I will have", "I'm": "I am", "I've": "I have", "isn't": "is not",
    "it'd": "it would", "it'd've": "it would have", "it'll": "it will",
    "it'll've": "it will have", "let's": "let us", "ma'am": "madam",
    "mayn't": "may not", "might've": "might have", "mightn't": "might not",
    "mightn't've": "might not have", "must've": "must have", "mustn't": "must not",
    "mustn't've": "must not have", "needn't": "need not",
    "needn't've": "need not have", "o'clock": "of the clock", "oughtn't": "ought not",
    "oughtn't've": "ought not have", "shan't": "shall not", "shan't": "shall not",
    "shan't've": "shall not have", "she'd": "she would", "she'd've": "she would have",
    "she'll": "she will", "she'll've": "she will have", "should've": "should have",
    "shouldn't": "should not", "shouldn't've": "should not have", "so've": "so have",
    "that'd": "that would", "that'd've": "that would have", "there'd": "there would",
    "there'd've": "there would have", "they'd": "they would",
    "they'd've": "they would have", "they'll": "they will",
    "they'll've": "they will have", "they're": "they are", "they've": "they have",
    "to've": "to have", "wasn't": "was not", "we'd": "we would",
    "we'd've": "we would have", "we'll": "we will", "we'll've": "we will have",
    "we're": "we are", "we've": "we have", "weren't": "were not", "what'll": "what will",
    "what'll've": "what will have", "what're": "what are", "what've": "what have",
    "when've": "when have", "where'd": "where did", "where've": "where have",
    "who'll": "who will", "who'll've": "who will have", "who've": "who have",
    "why've": "why have", "will've": "will have", "won't": "will not",
    "won't've": "will not have", "would've": "would have", "wouldn't": "would not",
    "wouldn't've": "would not have", "y'all": "you all", "y'all'd": "you all would",
    "y'all'd've": "you all would have", "y'all're": "you all are",
    "y'all've": "you all have", "you'd": "you would", "you'd've": "you would have",
    "you'll": "you will", "you'll've": "you will have", "you're": "you are",
    "you've": "you have" }
```

## Appendix 2 List of Stopwords

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
stop.append('ur') #in case people typed ur instead of your
print(stop)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'you're', 'you've', 'you'll', 'you'd', 'your', 'yours', 'y  
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'she's', 'her', 'hers', 'herself', 'it', 'it's', 'its', 'itself', 't  
hey', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'that'll', 'these', 'those',  
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a  
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b  
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of  
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',  
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',  
's', 't', 'can', 'will', 'just', 'don', 'don't', 'should', 'should've', 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar  
en', 'aren't', 'couldn', 'couldn't', 'didn', 'didn't', 'doesn', 'doesn't', 'hadn', 'hadn't', 'hasn', 'hasn't', 'haven', 'have  
n't', 'isn', 'isn't', 'ma', 'mightn', 'mightn't', 'mustn', 'mustn't', 'needn', 'needn't', 'shan', 'shan't', 'shouldn', 'should  
n't', 'wasn', 'wasn't', 'weren', 'weren't', 'won', 'won't', 'wouldn', 'wouldn't', 'ur']
```



## Appendix 3 Modelling for Machine Learning

```
helper function to show results and charts
show_summary_report(actual, prediction, predict_proba):

    if isinstance(actual, pd.Series):
        actual = actual.values
    if actual.dtype.name == 'object':
        actual = actual.astype(int)
    if prediction.dtype.name == 'object':
        prediction = prediction.astype(int)

    accuracy_ = accuracy_score(actual, prediction)
    precision_ = precision_score(actual, prediction)
    recall_ = recall_score(actual, prediction)
    roc_auc_ = roc_auc_score(actual, predict_proba)

    print('Accuracy: %.4f [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0' % accuracy_)
    print('Precision: %.4f [TP / (TP + FP)] Not to label a negative sample as positive. Best: 1, Worst: 0' % precision_)
    print('Recall : %.4f [TP / (TP + FN)] Find all the positive samples. Best: 1, Worst: 0' % recall_)
    print('ROC AUC : %.4f Best: 1, Worst: < 0.5' % roc_auc_)
    print('-' * 107)
    print('TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples')

    # Confusion Matrix
    mat = confusion_matrix(actual, prediction)

    # Precision/Recall
    precision, recall, _ = precision_recall_curve(actual, prediction)
    average_precision = average_precision_score(actual, prediction)

    # Compute ROC curve and ROC area
    fpr, tpr, _ = roc_curve(actual, predict_proba)
    roc_auc = auc(fpr, tpr)

    # plot
    fig, ax = plt.subplots(1, 3, figsize = (18, 6))
    fig.subplots_adjust(left = 0.02, right = 0.98, wspace = 0.2)

    # Confusion Matrix
    sns.heatmap(mat.T, square = True, annot = True, annot_kws={"size": 16}, fmt = 'd', cbar = False, cmap = 'Blues', ax = ax[0])

    label_font = {'size': '15'}
    title_font = {'size': '21'}

    ax[0].set_title('Confusion Matrix', fontdict=title_font)
    ax[0].set_xlabel('True label', fontdict=label_font)
    ax[0].set_ylabel('Predicted label', fontdict=label_font)

    # Precision/Recall
    step_kwargs = {'step': 'post'}
    ax[1].step(recall, precision, color = 'b', alpha = 0.2, where = 'post')
    ax[1].fill_between(recall, precision, alpha = 0.2, color = 'b', **step_kwargs)
    ax[1].set_ylim([0.0, 1.0])
    ax[1].set_xlim([0.0, 1.0])
    ax[1].set_xlabel('Recall', fontdict=label_font)
    ax[1].set_ylabel('Precision', fontdict=label_font)
    ax[1].set_title('2-class Precision-Recall curve', fontdict=title_font)

    # ROC
    ax[2].plot(fpr, tpr, color = 'darkorange', lw = 2, label = 'ROC curve (AUC = %0.2f)' % roc_auc)
    ax[2].plot([0, 1], [0, 1], color = 'navy', lw = 2, linestyle = '--')
    ax[2].set_xlim([0.0, 1.0])
    ax[2].set_ylim([0.0, 1.0])
    ax[2].set_xlabel('False Positive Rate', fontdict=label_font)
    ax[2].set_ylabel('True Positive Rate', fontdict=label_font)
    ax[2].set_title('Receiver Operating Characteristic', fontdict=title_font)
    ax[2].legend(loc = 'lower right', prop={'size': 15})

    plt.show()

    return (accuracy_, precision_, recall_, roc_auc_)
```

## Appendix 4 Modelling for Keras

```
#!/per function to show results and charts
show_summary_report(actual, prediction):

    accuracy_ = accuracy_score(actual, prediction)
    precision_ = precision_score(actual, prediction)
    recall_ = recall_score(actual, prediction)
    roc_auc_ = roc_auc_score(actual, prediction)

    print('Accuracy : %.4f [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0' % accuracy_)
    print('Precision: %.4f [TP / (TP + FP)] Not to label a negative sample as positive. Best: 1, Worst: 0' % precision_)
    print('Recall : %.4f [TP / (TP + FN)] Find all the positive samples. Best: 1, Worst: 0' % recall_)
    print('ROC AUC : %.4f Best: 1, Worst: < 0.5' % roc_auc_)
    print('-' * 107)
    print('TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples')

    # Confusion Matrix
    mat = confusion_matrix(actual, prediction)

    # Precision/Recall
    precision, recall, _ = precision_recall_curve(actual, prediction)
    average_precision = average_precision_score(actual, prediction)

    # Compute ROC curve and ROC area
    fpr, tpr, _ = roc_curve(actual, prediction)
    roc_auc = auc(fpr, tpr)

    # plot
    fig, ax = plt.subplots(1, 3, figsize = (18, 6))
    fig.subplots_adjust(left = 0.02, right = 0.98, wspace = 0.2)

    # Confusion Matrix
    sns.heatmap(mat.T, square = True, annot = True, annot_kws={"size": 16}, fmt = 'd', cbar = False, cmap = 'Blues', ax = ax[0])

    label_font = {'size': '15'}
    title_font = {'size': '21'}

    ax[0].set_title('Confusion Matrix', fontdict=title_font)
    ax[0].set_xlabel('True label', fontdict=label_font)
    ax[0].set_ylabel('Predicted label', fontdict=label_font)

    # Precision/Recall
    step_kwargs = {'step': 'post'}
    ax[1].step(recall, precision, color = 'b', alpha = 0.2, where = 'post')
    ax[1].fill_between(recall, precision, alpha = 0.2, color = 'b', **step_kwargs)
    ax[1].set_ylim([0.0, 1.0])
    ax[1].set_xlim([0.0, 1.0])
    ax[1].set_xlabel('Recall', fontdict=label_font)
    ax[1].set_ylabel('Precision', fontdict=label_font)
    ax[1].set_title('2-class Precision-Recall curve', fontdict=title_font)

    # ROC
    ax[2].plot(fpr, tpr, color = 'darkorange', lw = 2, label = 'ROC curve (AUC = %0.2f)' % roc_auc)
    ax[2].plot([0, 1], [0, 1], color = 'navy', lw = 2, linestyle = '--')
    ax[2].set_xlim([0.0, 1.0])
    ax[2].set_ylim([0.0, 1.0])
    ax[2].set_xlabel('False Positive Rate', fontdict=label_font)
    ax[2].set_ylabel('True Positive Rate', fontdict=label_font)
    ax[2].set_title('Receiver Operating Characteristic', fontdict=title_font)
    ax[2].legend(loc = 'lower right', props={'size': 15})

    plt.show()

    return (accuracy_, precision_, recall_, roc_auc_)
```

## References

- [1] T. Lee, "The global rise of "fake news" and the threat to democratic elections in the USA," 18 March 2019. [Online]. Available: <https://www.emerald.com/insight/content/doi/10.1108/PAP-04-2019-0008/full/html>. [Accessed 16 September 2020].
- [2] H. a. G. M. Allcott, "Social Media and Fake News in the 2016 Election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211-36, May 2017.
- [3] J. L. C. P. a. G. M. Juju Chang, "When Fake News Stories Make Real News Headlines," ABC News, 30 November 2016. [Online]. Available: <https://abcnews.go.com/Technology/fake-news-stories-make-real-news-headlines/story?id=43845383>. [Accessed 16 September 2020].
- [4] C. Wardle, "Fake news. It's complicated.," First Draft News, 17 February 2017. [Online]. Available: <https://firstdraftnews.org/latest/fake-news-complicated/>. [Accessed 16 September 2020].
- [5] L. Pagé, "IMPACTS OF FAKE NEWS," Dubois International Inc., 2019. [Online]. Available: <https://30secondes.org/en/module/impacts-of-fake-news/>. [Accessed 16 September 2020].
- [6] C. Young, "Here Are the Most Shared Fake Health News Stories of Last Year," Interesting Engineering, 10 January 2019. [Online]. Available:

<https://interestingengineering.com/here-are-the-most-shared-fake-health-news-stories-of-last-year>. [Accessed 16 September 2020].

[7] "The Economic Cost of Bad Actors on the Internet," CHEQ, University of Baltimore, 2019. [Online]. Available: <https://s3.amazonaws.com/media.mediapost.com/uploads/EconomicCostOfFakeNews.pdf>. [Accessed 16 September 2020].

[8] M. Jankowicz, "The coronavirus outbreak has prompted people around the world to panic buy toilet paper. Here's why.," Business Insider, 11 March 2020. [Online]. Available: <https://www.businessinsider.com/coronavirus-panic-buying-toilet-paper-stockpiling-photos-2020-3>. [Accessed 17 September 2020].

[9] K. Tsuchiya, "Japan toilet paper shortages are unconnected to coronavirus, but that's not encouraging," The Mainichi, 7 March 2020. [Online]. Available: <https://mainichi.jp/english/articles/20200306/p2a/00m/0na/028000c>. [Accessed 17 September 2020].

[1 R. K.-W. Imran Awan, "CORONAVIRUS, FEAR AND HOW ISLAMOPHOBIA SPREADS ON SOCIAL MEDIA," 20 April 2020. [Online]. Available: <https://anti-muslim-hatred-working-group.home.blog/2020/04/20/coronavirus-fear-and-how-islamophobia-spreads-on-social-media/>. [Accessed 17 September 2020].

[1 K. K. Elyse Samuels, "How misinformation on WhatsApp led to a mob killing in India," Washington Post, 21 February 2020. [Online]. Available:

<https://www.washingtonpost.com/politics/2020/02/21/how-misinformation-whatsapp-led-deathly-mob-lynching-india/>. [Accessed 17 September 2020].

[1 N. L. Shelly Banjo, "How Fake News and Rumors Are Stoking Division in Hong Kong,"

2] Bloomberg, 12 November 2019. [Online]. Available:

<https://www.bloombergquint.com/politics/how-fake-news-is-stoking-violence-and-anger-in-hong-kong>. [Accessed 17 September 2020].

[1 V. M. Munadhil Muqsith, "Effect Fake News for Democracy," *Indonesia Law Journal*, vol.

3] 7, no. 3, pp. 307-318, 19 December 2019.

[1 "Post-Truth," Cambridge Dictionary, [Online]. Available:

4] <https://dictionary.cambridge.org/dictionary/english/post-truth>. [Accessed 17 September 2020].

[1 L. R. Eugene Kiely, "How to Spot Fake News," FactCheck, 18 November 2016. [Online].

5] Available: <https://www.factcheck.org/2016/11/how-to-spot-fake-news/>. [Accessed 17 September 2020].

[1 "How to Spot Fake News," International Federation of Library Associations and

6] Institutions, 10 July 2020. [Online]. Available:

<https://www.ifla.org/publications/node/11174>. [Accessed 17 September 2020].

[1 R. England, "Google explains how it's fighting fake news," Engadget, 19 February 2019.

7] [Online]. Available: <https://www.engadget.com/2019-02-19-google-explains-how-it-is-fighting-fake-news.html>. [Accessed 17 September 2020].



- [1] I. L. Emily Dreyfuss, "Facebook Is Changing News Feed (Again) to Stop Fake News,"
- 8] Wired, 10 April 2019. [Online]. Available: <https://www.wired.com/story/facebook-click-gap-news-feed-changes/>. [Accessed 17 September 2020].
- [1] P. Lee, "Factually website clarifies 'widespread' falsehoods," Straits Times, 2 March
- 9] 2017. [Online]. Available: <https://www.straitstimes.com/singapore/factually-website-clarifies-widespread-falsehoods>. [Accessed 17 September 2020].
- [2] K. Devadass, "Select Committee formed to study deliberate online falsehoods," Channel
- 0] NewsAsia, 11 January 2018. [Online]. Available: <https://www.channelnewsasia.com/news/singapore/select-committee-formed-to-study-deliberate-online-falsehoods-9851096>. [Accessed 17 September 2020].
- [2] "Protection from Online Falsehoods and Manipulation Bill," Singapore Statutes Online,
- 1] 1 April 2019. [Online]. Available: <https://sso.agc.gov.sg/Bills-Supp/10-2019/Published/20190401?DocDate=20190401>. [Accessed 17 September 2020].
- [2] P. Grabiński, "Feature engineering, Explained," KD Nuggets, December 2018. [Online].
- 2] Available: <https://www.kdnuggets.com/2018/12/feature-engineering-explained.html>. [Accessed 17 September 2020].
- [2] H. E. BOUKKOURI, "Text Classification: The First Step Toward NLP Mastery," Medium, 18
- 3] June 2018. [Online]. Available: <https://medium.com/data-from-the-trenches/text-classification-the-first-step-toward-nlp-mastery-f5f95d525d73>. [Accessed 18 September 2020].

- [2] E. L. a. E. K. Steven Bird, Natural Language Processing with Python, O'Reilly Media Inc, 4] 2009.
- [2] E. Team, "CountVectorizer in Python," Edpresso, 27 August 2020. [Online]. Available: 5] <https://www.educative.io/edpresso/countvectorizer-in-python>. [Accessed 17 September 2020].
- [2] M. Chaudhary, "TF-IDF Vectorizer scikit-learn," Medium, 24 April 2020. [Online]. 6] Available: <https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>. [Accessed 17 September 2020].
- [2] "Text Classification," MonkeyLearn, 2020. [Online]. Available: 7] <https://monkeylearn.com/text-classification/#:~:text=Text%20classification%20is%20the%20process,spam%20detectio n%2C%20and%20intent%20detection..> [Accessed 18 September 2020].
- [2] N. S. Chauhan, "Naïve Bayes Algorithm: Everything you need to know," KD Nuggets, 8] June 2020. [Online]. Available: <https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html>. [Accessed 18 September 2020].
- [2] M. Oleszak, "Linear Classifiers: An Overview," Towards Data Science, 21 May 2019. 9] [Online]. Available: <https://towardsdatascience.com/linear-classifiers-an-overview-e121135bd3bb>. [Accessed 18 September 2020].
- [3] B. Stecanella, "An Introduction to Support Vector Machines (SVM)," MonkeyLearn, 22 0] June 2017. [Online]. Available: <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>. [Accessed 18 September 2020].

- [3] R. Garg, "A Primer to Ensemble Learning – Bagging and Boosting," Analytics India Mag, 19 February 2018. [Online]. Available: <https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/#:~:text=Bagging%20is%20a%20way%20to,based%20on%20the%20last%20classification..> [Accessed 18 September 2020].
- [3] A. SINGH, "A Comprehensive Guide to Ensemble Learning (with Python codes)," Analytics Vidhya, 18 June 2018. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>. [Accessed 18 September 2020].
- [3] J. Budzik, "Many Heads Are Better Than One: The Case For Ensemble Learning," KD Nuggets, September 2019. [Online]. Available: <https://www.kdnuggets.com/2019/09/ensemble-learning.html>. [Accessed 18 September 2020].
- [3] B. Marr, "What Is Deep Learning AI? A Simple Guide With 8 Practical Examples," Forbes, 1 October 2018. [Online]. Available: <https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/#763eed438d4b>. [Accessed 18 September 2020].
- [3] "Data Never Sleeps 8.0," DOMO, 2020. [Online]. Available: <https://www.domo.com/learn/data-never-sleeps->

8?utm\_source=wire&utm\_medium=pr&utm\_campaign=PR\_DNS\_8&campid=7015w000001OfjDAAS. [Accessed 18 September 2020].

[3] “What Is Deep Learning? 3 Things You Need to Know,” MathWorks, [Online]. Available:

6] <https://www.mathworks.com/discovery/deep-learning.html>. [Accessed 22 September 2020].

[3] R. Raicea, “Want to know how Deep Learning works? Here’s a quick guide for

7] everyone.,” freeCodeCamp, 23 October 2017. [Online]. Available:

<https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/>. [Accessed 22 September 2020].

[3] S. Raschka, “Machine Learning FAQ Fitting a model via closed-form equations vs.

8] Gradient Descent vs Stochastic Gradient Descent vs Mini-Batch Learning. What is the difference?,” Sebastian Raschka, [Online]. Available:

<https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html>. [Accessed 22 September 2020].

[3] R. Chandra, “The What’s What of Keras and TensorFlow,” upGrad Blog, 4 April 2019.

9] [Online]. Available: <https://www.upgrad.com/blog/the-whats-what-of-keras-and-tensorflow/#:~:text=Keras%20is%20a%20neural%20networks,wrapper%20to%20TensorFlow%20or%20Theano..> [Accessed 18 September 2020].

[4] “Real or Not? NLP with Disaster Tweets,” Kaggle, [Online]. Available:

0] <https://www.kaggle.com/c/nlp-getting-started/overview>. [Accessed 18 September 2020].