

Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

4-2022

Using Machine Learning in Disaster Tweets Classification

Humaid Alhammadi
hma8842@rit.edu

Follow this and additional works at: <https://repository.rit.edu/theses>

Recommended Citation

Alhammadi, Humaid, "Using Machine Learning in Disaster Tweets Classification" (2022). Thesis.
Rochester Institute of Technology. Accessed from

This Master's Project is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

Using Machine Learning in Disaster Tweets Classification

by

Humaid Alhammadi

**A Capstone Submitted in Partial Fulfilment of the Requirements for
the Degree of Master of Science in Professional Studies: Data
Analytics**

Department of Graduate Programs & Research

**Rochester Institute of Technology
RIT Dubai
April 2022**

RIT

Master of Science in Professional Studies: Data Analytics

Graduate Capstone Approval

Student Name: **Humaid Alhammadi**

Graduate Capstone Title: **Using Machine Learning in Disaster Tweets Classification**

Graduate Capstone Committee:

Name: **Dr. Sanjay Modak**
Chair of committee

Date:

Name: **Dr. Ehsan Warriach**
Member of committee

Date:

Acknowledgments

First and foremost, I would like to praise Allah, all praises to Allah, the Most Compassionate, the Most Merciful, Glory to Allah.

Special thanks and appreciation to my mentor and advisor, Dr. Ehsan Warriach, for his continuous support, cooperation, and mentorship during the whole capstone project. I would also like to thank Dr. Sanjay Modak, Chair of Graduate Programs and Research in RIT Dubai for his advice, guidance, and feedback during the whole program.

Furthermore, I would to express special thanks to my mother and father, who always showed their support, and especially throughout the master program. Also, to my family, friends and colleagues for their continuous support throughout the research.

Abstract

People share real-time updates on social media platforms (i.e. Twitter) when disaster occurs, this information is very valuable for disaster relief and response teams as it can alert them immediately in order to prioritize tasks. Text mining and Machine learning algorithm can scan the huge generated unstructured data on social media platforms such as Twitter, to spot such information through keywords and phrases that refers to disasters. One challenge that the algorithm might face is whether a tweet text is talking about a real disaster or uses those keywords as a metaphor, which can lead to huge mislabeling of tweets. Hence, this research aims on using Natural Language Processing (NLP) and classification models to distinguish between real and fake disaster tweets. The dataset was acquired from Kaggle website, and it contain tweets that are related to real disasters, and other tweets that refers to fake disasters. Furthermore, using RStudio software, exploratory data analysis (EDA), feature selections, and data cleaning were performed prior to the data modeling, two different training to testing split were tested. In addition, four classifiers were built, which are SVM, KNN, Naïve Bayes, and XGBoost. As a result, the best accuracies achieved with 80/20 ratio split, and with using the whole dataset rather than sampling, SVM and XGBoost performed well with accuracies of 80% and 78% respectively, while KNN suffered overfitting (99% accuracy) and Naïve Bayes performed poorly (65%).

Keywords: Disaster Tweets, Machine learning, Text Classification, Text Preprocessing

Table of Contents

ACKNOWLEDGMENTS	II
ABSTRACT	III
LIST OF FIGURES	V
LIST OF TABLES.....	V
CHAPTER 1: INTRODUCTION	1
1.1 BACKGROUND INFORMATION	1
1.2 STATEMENT OF THE PROBLEM.....	2
1.3 PROJECT AIMS AND OBJECTIVES	2
1.4 RESEARCH METHODOLOGY	2
1.5 LIMITATIONS OF THE STUDY	4
CHAPTER 2: LITERATURE REVIEW	5
CHAPTER 3: PROJECT DESCRIPTION.....	10
3.1 PROJECT OVERVIEW	10
3.2 DATA SET OVERVIEW	10
CHAPTER 4: DATA ANALYSIS.....	12
4.1 DATA UNDERSTANDING	12
4.1.1 DATA SET EXPLORATION	12
4.1.2 KEYWORD AND LOCATION PRE-PROCESSING.....	15
4.1.3 TWEET TEXT EXPLORATION – REAL DISASTERS	18
4.1.4 TWEET TEXT EXPLORATION – FAKE DISASTERS	23
4.2 DATA PRE-PROCESSING	27
4.3 DATA MODELING.....	31
4.3.1 SUPPORT VECTOR MACHINE (SVM) CLASSIFIER.....	32
4.3.2 K-NEAREST NEIGHBOR CLASSIFIER.....	33
4.3.3 NAÏVE BAYES CLASSIFIER.....	35
4.3.4 XGBOOST CLASSIFIER.....	37
4.4 RESULTS	39
CHAPTER 5: CONCLUSION	42
5.1 CONCLUSION	42
5.2 RECOMMENDATIONS	43
REFERENCES	44

List of Figures

Figure 1: Importing the data set	12
Figure 2: Data set structure	13
Figure 3: Data set Summary.....	13
Figure 4: Data set null values	13
Figure 5: Missing Value Percentage	14
Figure 6: Distribution of Target Column	15
Figure 7: Dealing with null values	15
Figure 8: Null value after data pre-processing.....	16
Figure 9: Top 10 repeated keywords for each output	16
Figure 10: Missing keywords percentage per target	17
Figure 11: Top 10 recurring locations	18
Figure 12: Sub setting Real Disasters and creating a corpus	19
Figure 13: Data Cleaning - Real Disasters	20
Figure 14: Word Cloud - Real Disaster Tweets.....	21
Figure 15: Top 10 Words - Real Disasters	22
Figure 16: Sentiment Analysis of Real Disasters Words	23
Figure 17: Sub setting fake disaster tweets and creating a corpus	24
Figure 18: Data Cleaning - Fake Disasters	24
Figure 19: Word Cloud - Fake Disaster Tweets	25
Figure 20: Top 10 Words - Fake Disasters.....	26
Figure 21: Sentiment Analysis for Fake Disaster Words	26
Figure 22: Data selection - modeling part.....	29
Figure 23: Text data cleaning	30
Figure 24: Data Partition	30
Figure 25: Changing TDM to a data frame	31
Figure 26: SVM Classifier.....	32
Figure 27: SVM Confusion Matrix	33
Figure 28: KNN Classifier	34
Figure 29: KNN Confusion Matrix	34
Figure 30: Naive Bayes Classifier.....	36
Figure 31: Naive Bayes Confusion Matrix	37
Figure 32: XGBoost Classifier	38
Figure 33: XGBoost Confusion Matrix.....	38

List of Tables

Table 1: Data set Attribute Description	11
Table 2: Model Accuracies Results.....	39
Table 3: Trial 1 Model Performance Metrics	40

Chapter 1: Introduction

1.1 Background Information

In the last decade, social media platform (e.g. Twitter, Instagram, Facebook and Snapchat) became very popular among the world, and the amount of unstructured data generated by users from all are unprecedented, where people can share their opinions on different subjects and discuss certain issues and complaints (Agarwal et al. 2011). This provides an opportunity for different businesses, where by using data analytics tools and algorithms, they can know more about how people feel and obtain a sentimental reaction (Çelikutğ, 2018), which in its way can help in applications such as marketing, political campaign, crisis management, and several other fields.

One significant application is using sentiment analysis for disaster relief and response teams, people that are near a live disaster can post that on their social media accounts instantly, and the idea here is to analyze that piece of text, and if it refers to a disaster (wild fire, earthquake, etc.), that can help in mobilizing response teams immediately. Furthermore, one challenge that an algorithm might face in analyzing a text, is to differentiate whether it talks about a real disaster, or it is just a humor, irony, or a metaphor. For example, the following tweet: "Twitter is just here to destroy your childhood" ("Natural Language Processing with Disaster Tweets | Kaggle", 2021), even though the word 'destroy' was used as a metaphor, it is challenging for a machine learning algorithm to filter out that metaphor, which can cause confusion in detecting and reporting both actual and fake disasters based on those tweets and keywords.

1.2 Statement of the Problem

Social media platforms revolutionized the way people share information in an unprecedented manner. This paradigm can be utilized by emergency response teams to be aware of major disasters in advance, with the aid of advanced automated machine learning algorithm. However, algorithm can find it challenging in differentiating whether the piece of information (tweets, posts, etc....) is describing an actual event, a rumor, sarcasm, irony, or a mere metaphor. So, it is very important to develop a solution to enhance the ability of machine learning algorithms to distinguish real information from fake or metaphors.

1.3 Project Aims and Objectives

The aim of this project is to apply the knowledge and skills in the field of data analysis and machine learning to build a classification model based on supervised learning that can accurately identify whether a piece of information (tweet text) is referring to a real disaster or not. Hence, supporting disaster relief and response teams to accurately response to real disaster and not fake ones.

1.4 Research Methodology

This study will be following the CRISP-DM methodology, as follows:

Step 1: Business Understanding

First, we have to understand what are the business benefits that we are trying to achieve in this study. Moreover, in our case, creating a classification model that can classify real disaster tweets from fake one can be significant. Disaster relief teams can have an earlier

notice if they can ensure that a posted tweet is referring to a real disaster, while if the real and fake tweets cannot be distinguished, such response teams will not rely on social media platform, although they illustrated their benefit with providing real-time information.

Step 2: Data Understanding

In the second step, the dataset is to be explored, the aim is to look into the data, observe the features and records, and visualize attributes in order to have a better understanding on how they are correlated with the output results.

Step 3: Data Pre-processing

In the third step, the collected dataset will be examined and prepared, since the dataset contains tweets text, data cleaning steps will involve removing URLs, symbols, emojis, accounts mention (@user), hash-tags (#), as well as dealing with null values within the dataset. In addition, the dataset will be split into training and testing subsets. The aim is to have a clean dataset prior to the model building step.

Step 4: Modeling

In the fourth step, RStudio software will be used to build multiple supervised classification models, such as K-nearest neighbor (KNN), Support Vector Machine (SVM), Random Forest, and Logistic Regression. The goal is to test how various classifier will perform on the training subset.

Step 5: Evaluation

In the final step, the classifiers that were built based on the training subset will be evaluated and compared to the testing subset. Here the confusion matrix will be the main evaluating tool, where recall, precision, accuracy, F-score and other parameters can be calculated.

1.5 Limitations of the Study

This study was limited based on the availability of the data set, disaster related agencies usually have confidentiality that prevent them from sharing up-to-date information about how certain text can be flagged. Therefore, an offline data set collected from Twitter, that contains thousands of tweets that are labeled as real or fake disaster related tweet. Therefore, the study will be limited on a set of data that might capture a portion of disaster related tweets, and it will be on an offline tweets.

Chapter 2: Literature Review

According to Rani et al. (2016), Tweets can be taken as a raw data for opinion extraction to help in providing business intelligence, and assist in strategic decision making, and can be used in various fields, such as analyzing customer satisfaction, all of that is finally related to sentiment analysis/opinion mining. Sentiment analysis is method of determining the sentiment (opinion) of a user on a certain topic (e.g. Positive, Negative) (Samuels et al., 2013). Furthermore, in our case, the problem that we are dealing with is to determine if the use of a certain words is referring to a real disaster or if it is being figuratively. According to Patterson (2013) metaphors lacks the conventional referents of a word, while Ghosh et al. (2015) considered three classes of figurative language, and they are: Irony, Sarcasm, and Metaphor. In addition, they also considered the use of this language to be a challenge for Natural Language Processing (NLP) tools, where it does not take into account the way of using such words creatively, and that figurative language could be used generally in any genre, especially that social media platforms are being used by all kind of people with different interest and that it is more of casual rather than a professional platform.

Tweets text content contains text, links, symbols, emojis and shapes, and prior any further analysis, it usually requires some data pre-processing to bring it to a 'standard' form. Madichetty et al. (2021), Arun et al. (2017), and Nikam et al. (2020) performed similar data pre-processing steps, and they were:

- Changing tweets letters to lower case.
- Removing stop words, URLs, user mentions (@user), hash-tags (#), symbols, emojis.
- Dividing the words into tokens (Tokenization).

- Stemming.

Moreover, Magliani et al. (2016) stated the importance of removing tabs and line breaks, and that the idea in data pre-processing is to make the tweet text uniform and of less noise, and they illustrated the importance of data pre-processing and how it can affect the classifier performance. Moreover, the lack of available corpora for foresight demonstrating is a huge stumbling block in developing credible algorithms to detect fake news. Rubin et al. (2016) provided a model for detecting parody and comedy in news items as well. They examined and evaluated 360 sarcastic news pieces in four categories: civics, science, business, and "delicate news". They proposed an SVM model based on their analysis of the sarcastic news, which consisted primarily of 5 features. Absurdity, Humor, Grammar, Negative Affect, and Punctuation are the five highlights. Their highest level of accuracy, 90 percent, was achieved by combining only three elements: absurdity, grammar, and punctuation. Furthermore, Horne and Adali (2017) demonstrated how easy it is to distinguish between fake and authentic content. Fake news titles, in their opinion, contain fewer stop words and things, but more things and action words. They separated the numerous components into three categories as follows:

- Intricacy highlights draw attention to the text's complexity and consistency.
- The quantity of emotion words and easygoing words, for example, are used in brain research to define and assess the mental interaction and individual anxieties concealed inside the compositions.
- Extensive features, such as the number of action words and the number of objects, reflect the writers' style and the text's linguistic structure.

Conroy et al. (2015) described the Linguistic Cue Approaches with Machine Learning, Bag of words method, Rhetorical Structure and discourse analysis, Network Examination Draws Near, and SVM classifiers. These models are solely text-based, with very little or no augmentation for current approaches. In addition, Chen et al. (2015) discusses that click baiting in online platform are the equivalent of journals tabloid. They've depicted click baiting as a form of rapid dissemination of rumors and lies over the internet. Expected tactics for programmed location of deceptive material have been mentioned by the authors to be tricky. They carried out lexical and semantic levels of investigation when creating content signals. *Zhang, J. et al. (2018)* looked into the criteria, tactics, and computations used for identifying misleading news items, as well as topics from online social networking sites, and evaluating their reach and execution. The paper also presented research challenges based on the invisible characteristics of fake news and various connections between news items, authors, and subjects. The authors also looked at the Fake News Detector model, which is a programmed fake news derivation model. It is based on literary order and employs a deep neural network model to become acquainted with the depictions of news articles, producers, and issues all at once.

In term of the supervised classification methods, Cobo et al. (2015) worked on classification of relevant and non-relevant words to an earth-quake in Chile, they did it on two selected features, the user-based where they look at the account followers and profile, and the content-based, where text mining is done, also, they used multiple classifiers, such as Logistic Regression, Random Forest, Support Vector Machine, Naive Bayes, and to evaluate the models they looked at precision, recall, F-score, area under the curve (AUC). In addition, Ashktorab et al. (2014) and Imran et al. (2016) followed similar models and evaluation measures, however, Ashktorab et al.

(2014) used K nearest neighbor (KNN) as well. Additionally, Hörtenhuemer et al. (2020) used a multi-aspect classification ensemble to profile Real from Fake news, their goal were to automatically detect if a text is faking news in social media platforms, the tweets where either assigned that it were written by fake news spreaders or, written by truth-tellers. They extracted multiple features from the text, such as Bag of Words (BOW), N-grams, Term Frequency-Inverse document frequency (TF-IDF), and several other techniques to aid in better setting the tweets features.

In contrast, for the unsupervised classification methods, Kumar et al. (2020) worked in prioritizing tweets over others to aid the disaster relief organizations and help them in decision making and in arranging their task, they classified tweets as either informative or non-informative, informative tweets are those where people are asking for help, or provided information about damage on infrastructure, injured or dead people, while the other type (non-informative) that can be such as prayers, thanking messages, etcetera. To classify tweets text, they relied on long-short-term-memory (LSTM) and deep neural network.

All of the above authors agreed to some degree on the limitation while classifying tweets, and those limitations were: tweets size (140 characters earlier, now 280 characters), non-standard abbreviation, and grammatical errors.

To sum up, tweets text classifications is very common in the literature, with different applications varying brand marketing to political campaigns and others. Scholars perform necessary data preparation to upbring the quality of the dataset text, and then introduce it to several potential classifiers such as Random Forest, Naïve Bayes, KNN, and SVM, and accordingly, evaluating the

models based on common parameters such as F-score, accuracy, recall, and precision, that are calculated using the confusion matrix.

Chapter 3: Project Description

3.1 Project Overview

The project will go through multiple steps to eventually achieve the primary objective of building a classification model that can classify real from fake text. After selecting the dataset, the project will start by performing exploratory data analysis (EDA) and data visualization for the selected data set. Moreover, data pre-processing steps will be performed to clean the data, deal with empty cells, and try to improve the quality of the data set before reaching the data modeling step. Upon building the model based on the clean data set, the models will be tested based on the proper metrics to evaluate its performance. Furthermore, RStudio software will be used to perform all data related tasks.

3.2 Data Set Overview

The data set is obtained from Kaggle website, which is a popular site and an online community for data scientist and machine learning (ML) practitioner, which provides a wide variety of data science and ML problems. The problem am tackling is “Natural Language Processing with Disaster Tweets”, which is a competition posted by the website, it contains two datasets that consist of 10,876 tweets, a training dataset (train.csv, 7,613 rows), and testing dataset (test.csv, 3,262 rows), the difference between the two datasets is that the training dataset contains a target attribute while the testing dataset does not, and that it because it is a competition where they want to test competitors. Consequently, for this project I will be

selecting the training dataset only as the main dataset, and divide it between a training and testing subsets, in order to be able to evaluate the model after all.

Dataset Link: <https://www.kaggle.com/c/nlp-getting-started/data>

The train.csv dataset contains five attributes, as shows in the below table:

No	Attribute	Attribute Description	Format
1	id	A unique id identifier to each tweet	double
2	keyword	A particular keyword from the tweet text	chr
3	location	The location the tweet was sent from	chr
4	text	The text of the tweet	chr
5	target	The target output, whether a tweet is about real disaster (1) or not (0)	double

Table 1: Data set Attribute Description

The data set contains some null values, and they are on the keyword and location attributes, which might require data pre-processing steps to deal with them.

Chapter 4: Data Analysis

4.1 Data Understanding

To begin, we first must learn and study the working data set, this step is very important to try and highlight some features and characteristics about our data set. For example, the five attributes must be looked at individually in term of structure, type, occurrence, null values, and so. In addition, exploratory data analysis (EDA) will include the generation of word cloud, estimating sentiments, plotting attributes to learn more about them such as keyword and location features.

The following steps were performed is RStudio software.

4.1.1 Data set exploration

First, we will load all required libraries to perform exploratory data analysis (EDA), visualization, data pre-processing, data modeling and evaluation, as well as the working data set, into RStudio software:

A screenshot of the RStudio console window. The code is as follows:

```
#Importing the dataset
##{r}
data <- read.csv("C:/Users/User/OneDrive/RIT/Courses/PROF770.607.CapstoneProposal/Dataset/t
rain.csv", header=T, na.strings=c("", "NA"))
data$text <- iconv(data$text, 'UTF-8', 'ASCII')
data <- as.data.frame(data)
head(data)
```

The code is color-coded: comments are blue, function names like `read.csv` and `as.data.frame` are green, and variable names and operators are black. The console window has a light gray background and a vertical scrollbar on the right.

Figure 1: Importing the data set

Then, the structure, summary, and null value of the data set were found:

```
#Dataset summary and structure
##{r}
str(data)

'data.frame': 7613 obs. of 5 variables:
 $ id      : int  1 4 5 6 7 8 10 13 14 15 ...
 $ keyword : chr   NA NA NA NA NA ...
 $ location: chr   NA NA NA NA NA ...
 $ text    : chr  "Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all"
               "Forest fire near La Ronge Sask. Canada" "All residents asked to 'shelter in place' are
               being notified by officers. No other evacuation or shelter in pla"| __truncated__ "13,000
               people receive #wildfires evacuation orders in California " ...
 $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

Figure 2: Data set structure

```
##{r}
summary(data)

      id      keyword      location      text
Min.   : 1      Length:7613      Length:7613      Length:7613
1st Qu.:2734      Class :character      Class :character      Class :character
Median :5408      Mode  :character      Mode  :character      Mode  :character
Mean   :5442
3rd Qu.:8146
Max.   :10873
      target
Min.   :0.0000
1st Qu.:0.0000
Median :0.0000
Mean   :0.4297
3rd Qu.:1.0000
Max.   :1.0000
```

Figure 3: Data set Summary

```
#Checking the null values within the dataset
##{r}
colSums(is.na(data))

      id keyword location      text      target
0         61      2533      697         0
```

Figure 4: Data set null values

From the figures below, we can see that our data set consists of 7,613 observations of 5 variables. The first variable, “id”, is a unique identification number (integer) that is associated with each record. “Keyword” is a character variable, that is about a keyword in each tweet. “location” is also of char type, and it refers to the location where the tweets were posted from. “text” is the main attribute in this dataset, it is of char type as well, and it is the body of the tweet. Finally, the last variable is “target” and it is an integer of either “1” or “0”, target refers to the output and the label of each record, with “1” referring to yes, that the tweet is referring to a real disaster,

while “0” is referring to fake disaster tweets. Furthermore, from checking the null values, we can see that “id” and “target” does not have any null values, while “Keyword” has 61 missing values, “location” has 2533 null values, and “text” have 697 null values, as shown in the below figure”

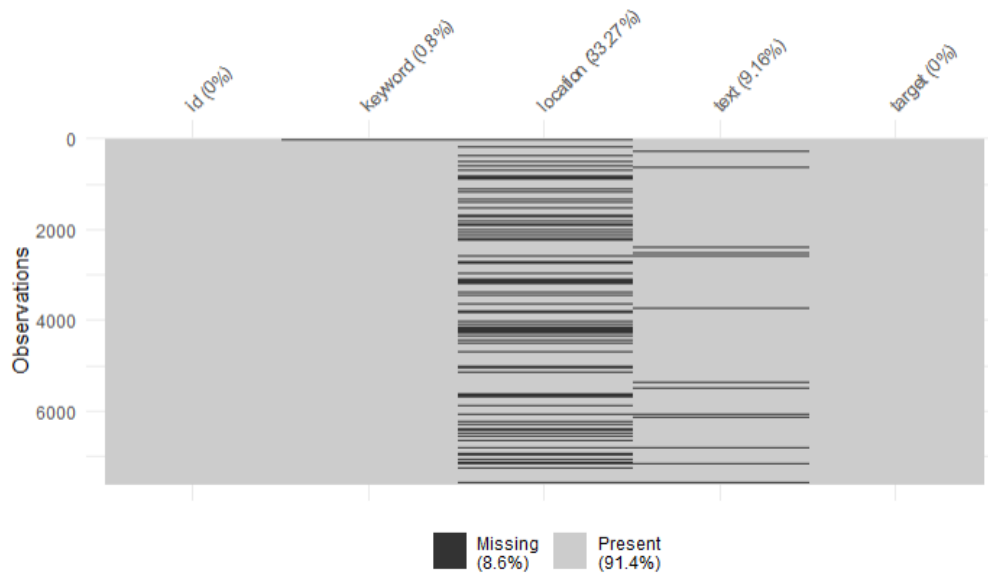


Figure 5: Missing Value Percentage

It is clear that the train.csv data set contains missing null values of 8.6%, and this is will be considered during the coming steps.

Next, “target” variable was visualized, as shown in the pie chart below:

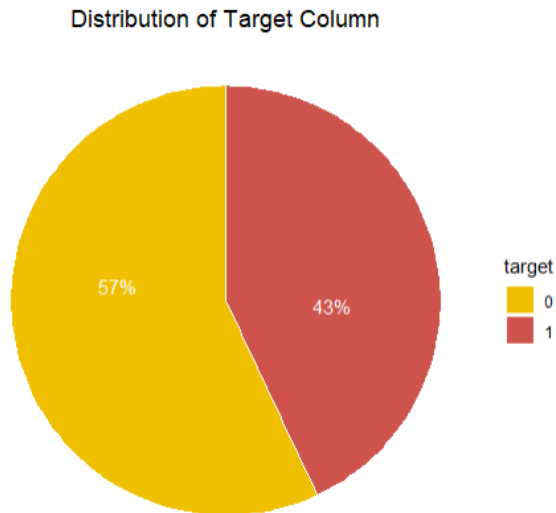


Figure 6: Distribution of Target Column

It is observed that the data set contains more records of “0” label, which refers to fake disaster tweets than the number of “1” label records, with the first being of 57% of the total data set. One solution that is going to be tested is to sample an equal amount of tweets for both labels in order to have a balance subset that consist of 50% for each target variable.

4.1.2 Keyword and Location pre-processing

Then, we assigned “no_keyword” for the empty cells in the “Keyword” column, and “no_location” for the null “location” variable, as shown in the figures below:

```
```{r}
df <- data %>%
 mutate(keyword = str_replace(keyword, "%20", "; "),
 keyword = ifelse(is.na(keyword), "no_keyword", keyword),
 location = ifelse(is.na(location), "no_location", location),
 real = ifelse(target == 1, "yes", "no") %>%
 as_factor())

df %>%
 select(id, real, location, keyword, text)
```

Figure 7: Dealing with null values

Also, we created a new variable “real” that is either “yes” or “no” which refers if the tweet is about a real disaster or not.

```
{r}
colSums(is.na(df))
```

id	keyword	location	text	target	real
0	0	0	697	0	0

Figure 8: Null value after data pre-processing

After performing data pre-processing for keyword and location features, we can see that they contain to null values.

Then, we checked the most re-occurring keywords for each target variable, as shown in the figure below:

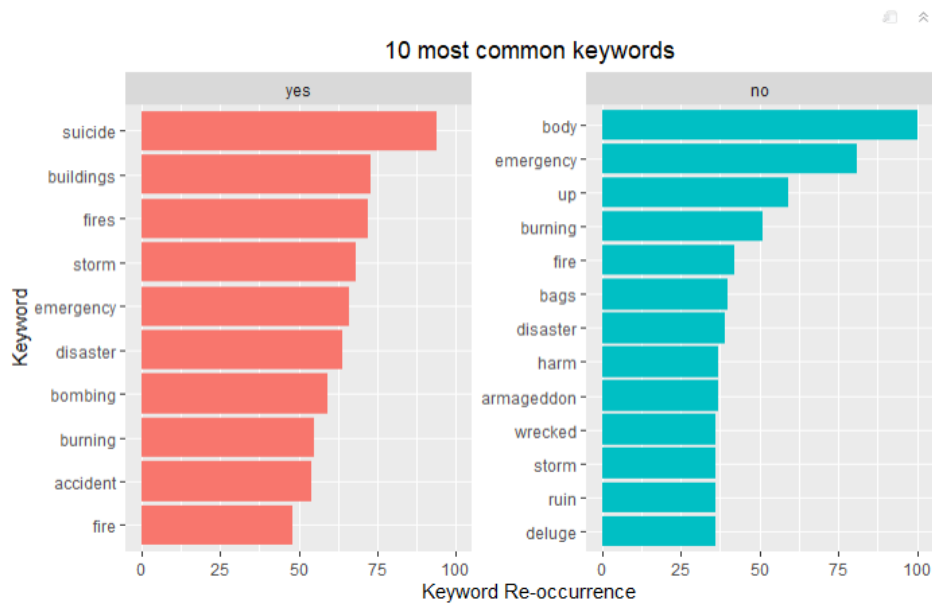


Figure 9: Top 10 repeated keywords for each output

We can see that some of the top 10 re-occurring words are shared between both target, such as “emergency”, “fire”, “storm” and “disaster”. Words such as “suicide”, “bombing”, “burning”, and “buildings” are mostly associated with real disaster. In contrast, words such as “body”, “up”, “Armageddon”, and “ruin” are associated with fake tweets. The shared tweets clearly shows a challenge for the classifier to distinguish the labeling of tweet, because as discussed earlier, some

words are used as a metaphor or an irony to describe an event rather than using those words literally.

Also, an analysis was performed on the missing keyword, as shown in the graph below:

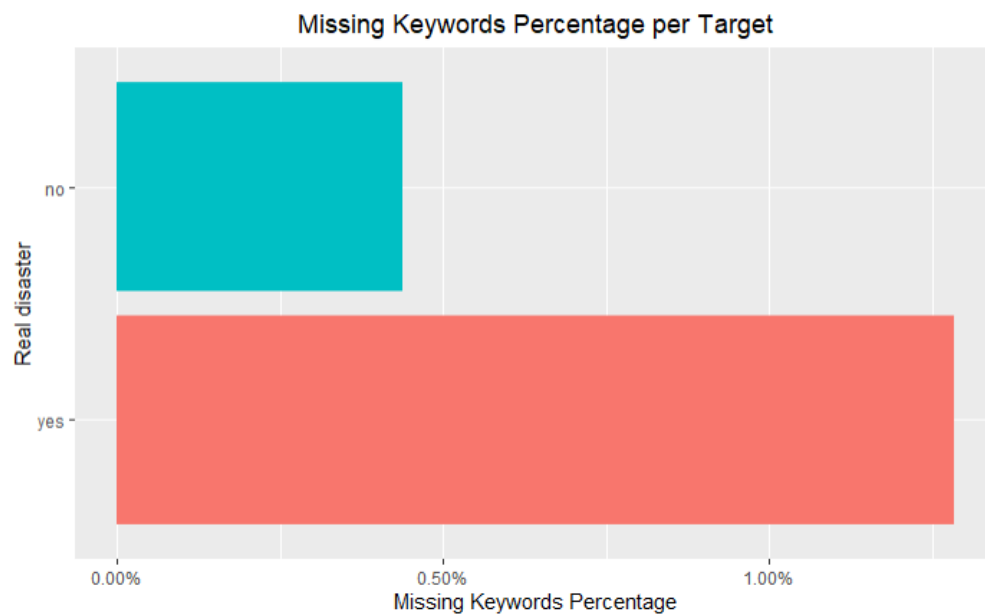


Figure 10: Missing keywords percentage per target

It is clearly shown that there are more missing keyword entries in real disaster when compared to fake disaster, and this is spite being the "0" target consisting of more records in the whole data set. However, both are less than 2% of total records.

The same steps were performed to "location" variable, the top 10 re-occurring locations per target were plotted:



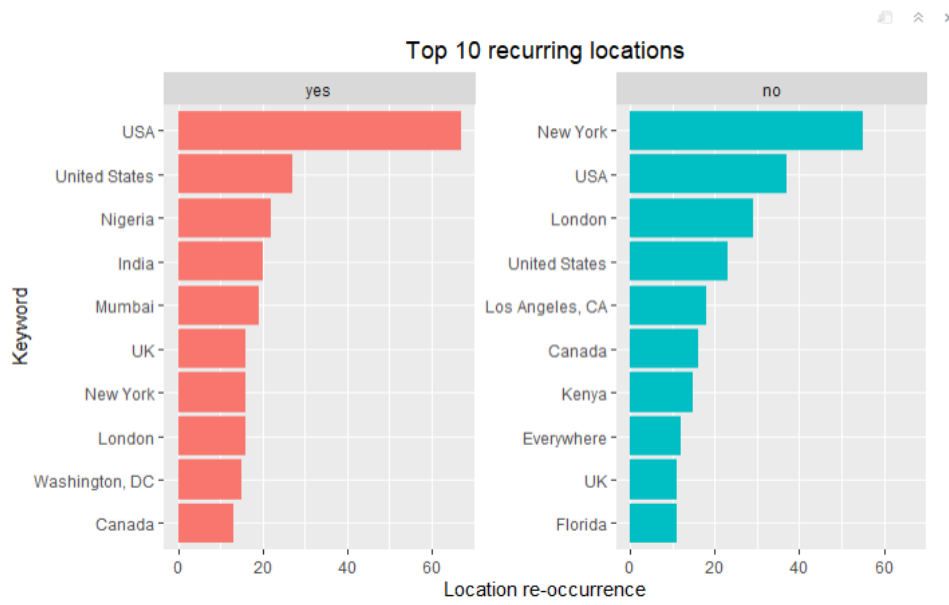


Figure 11: Top 10 recurring locations

It is clear that the data set location data is not very precise, where it is observed that the location is just a characters entries, we can see entries such as USA, and United States that have the exact meaning. In addition, lots of the entries are referring to States or Cities within the United States, such as “New York”, “Washington, DC”, “Los Angeles, CA”, and “Florida”. In general, most tweets are from the US for both targets, we can also see that real disasters took place in “Nigeria”, “India”, “UK” and “London”, while for the fake disaster, also “London” and “UK” are recurring entries.

#### 4.1.3 Tweet text exploration – Real Disasters

In this subsection, the main interesting variable in the data set, “text”, which is the tweet text itself is going to be explored more, and pre-processed in order to have it ready for the modeling step.

First, the data set is split into two subsets, based on the target variable (1 and 0), and for each subset, a corpus will be created, and then it will undergo data cleaning, then a term document matrix (TDM) will be created based on it, which will be visualized into a word cloud, and the sentiment of the TDM will be calculated. Also, a text file “stop\_words\_english.txt” is imported, the text file includes lots of stop words (854 words) that are to be removed from the tweets.

This section will focus mainly on Real Disaster subset.

```
##[r, warning=FALSE, echo=FALSE]
Creating a corpus.
Target_1 <- df %>%
 filter(target == 1)
text_1 <- Target_1$text
docs_1 <- Corpus(VectorSource(text_1))
```

Figure 12: Sub setting Real Disasters and creating a corpus

In the above step, the subset for real disaster and a corpus for the text variable is created, then, the figure below shows the code of the data cleaning:

```

Data Cleaning
```{r, echo=FALSE, warning=FALSE}

toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs_1 <- tm_map(docs_1,
  toSpace,
  "/"")
docs_1 <- tm_map(docs_1,
  toSpace,
  "@")
docs_1 <- tm_map(docs_1,
  toSpace,
  "\\|")

docs_1 <- tm_map(docs_1,
  content_transformer(tolower))

docs_1 <- tm_map(docs_1,
  removeNumbers)

docs_1 <- tm_map(docs_1,
  removeWords,
  stopwords("english"))

docs_1 <- tm_map(docs_1,
  removeWords,
  x)

docs_1 <- tm_map(docs_1,
  removePunctuation)

docs_1 <- tm_map(docs_1,
  stripWhitespace)

docs_1 <- tm_map(docs_1,
  function(x) iconv(enc2utf8(x), sub = "byte"))
```

```

Figure 13: Data Cleaning - Real Disasters

Here, we removed symbols, transform text into lower case, removed numbers, stop words, stop words from the imported text file, punctuations, and white spaces. A TDM is created based on the clean text data. The TDM includes two attributes, “word” and “freq”, so after performing data cleaning, we are left with the TDM that include all remaining words, with the frequency of their occurrence. Then, a word cloud is created based on the TDM, the term “word cloud” refers to the visualization of the occurrence of words. The larger the word in the word cloud means it is more frequent in the text. Word cloud are popular way that are used in text material, and they are used in all relevant applications. They can be used to study Twitter content and evaluate the opinions of people on certain topic or to stop hate speech. The figure below is the generated word cloud for tweets of real disasters:



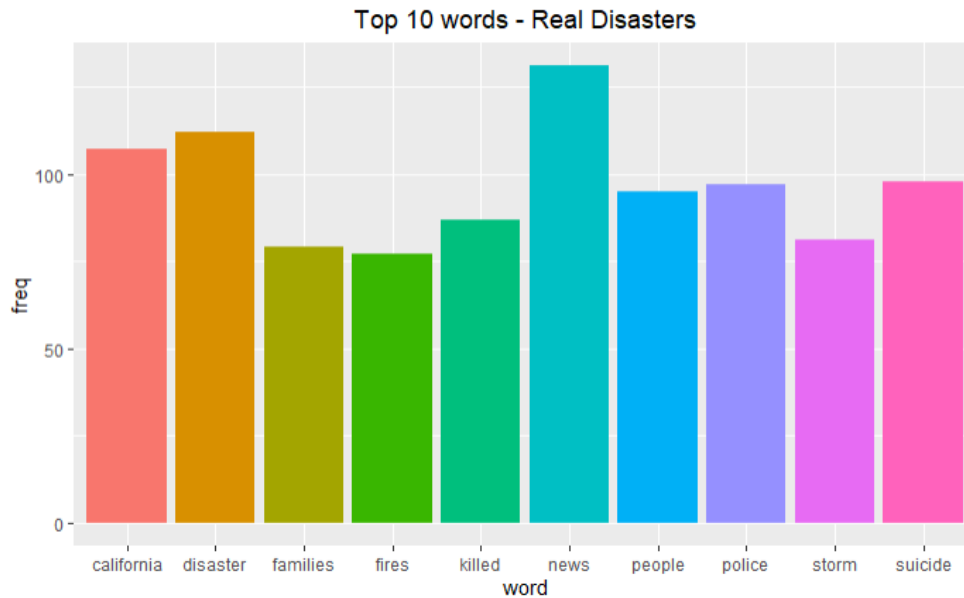


Figure 15: Top 10 Words - Real Disasters

Then, the sentiment of the real disaster TDM was calculated using NRC Dictionary, where we can have a closer look about the general feeling from those tweets. The NRC Dictionary mainly have 10 sentiments, and they are: “Anger”, “Anticipation”, “disgust”, “fear”, “joy”, “sadness”, “surprise”, and “trust”. The bar plot below shows the calculated estimate of the real disasters:

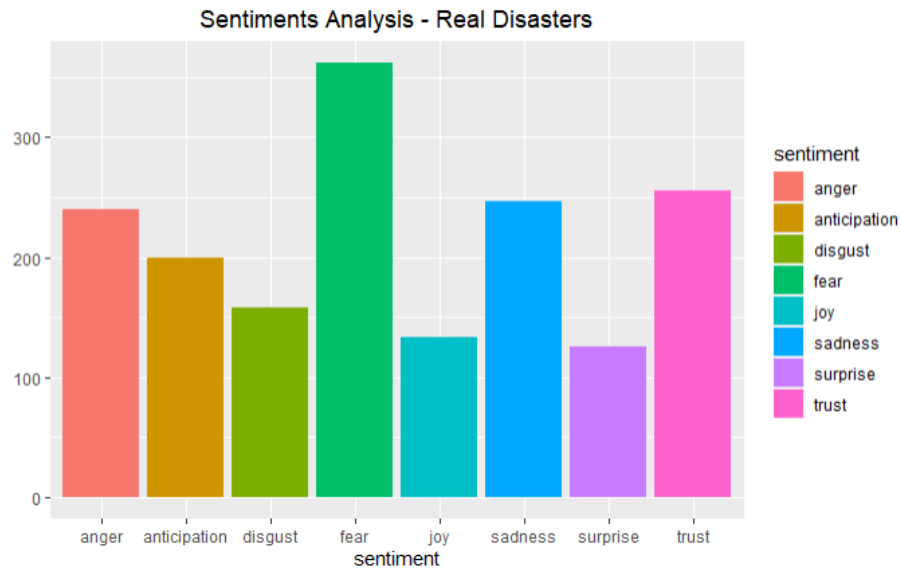


Figure 16: Sentiment Analysis of Real Disasters Words

Figure 16 illustrates that most words sentiments are linked with “fear”, followed by “trust”, “sadness” and then “anger”. It is very clear that most words are giving a negative impression. While the least sentiments are “joy”, and “surprise”, which means that negative sentiments are dominating the real disaster tweets.

#### 4.1.4 Tweet text exploration – Fake Disasters

In this subsection, we will perform similar steps as the previous section, but the focus is going to be on the fake disaster tweets. So the subset will contain only the fake disaster tweets, and data cleaning, TDM, and visualization will be performed to it.

the main interesting variable in the data set, “text”, which is the tweet text itself is going to be explored more, and pre-processed in order to have it ready for the modeling step.

First, the data set is split into two subsets, based on the target variable (1 and 0), and for each subset, a corpus will be created, and then it will undergo data cleaning, then it a term document

matrix (TDM) will be created based on it, which will be visualized into a word cloud, and the sentiment of the TDM will be calculated. Also, a text file “stop\_words\_english.txt” is imported, the text file includes lots of stop words (854 words) that are to be removed from the tweets.

This section will focus mainly on Real Disaster subset.

```
Corpus for target 0 text
```{r, warning=FALSE, echo=FALSE}
Target_0 <- df %>%
  filter(target == 0)
text_0 <- Target_0$text
docs_0 <- Corpus(VectorSource(text_0))
```
```

Figure 17: Sub setting fake disaster tweets and creating a corpus

In the above step, the subset for fake disaster and a corpus for the text variable is created, then, the figure below shows the code of the data cleaning:

```
Data Cleaning
```{r, echo=FALSE, warning=FALSE}

toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs_0 <- tm_map(docs_0,
  toSpace,
  "/"
)
docs_0 <- tm_map(docs_0,
  toSpace,
  "@"
)
docs_0 <- tm_map(docs_0,
  toSpace,
  "\\| "
)

docs_0 <- tm_map(docs_0,
  content_transformer(tolower))

docs_0 <- tm_map(docs_0,
  removeNumbers)

docs_0 <- tm_map(docs_0,
  removeWords,
  stopwords("english"))

docs_0 <- tm_map(docs_0,
  removeWords,
  x)

docs_0 <- tm_map(docs_0,
  removePunctuation)

docs_0 <- tm_map(docs_0,
  stripWhitespace)

docs_0 <- tm_map(docs_0,
  function(x) iconv(enc2utf8(x), sub = "byte"))
```
```

Figure 18: Data Cleaning - Fake Disasters

The steps performed are the exact same data cleaning steps that were performed on the real disaster tweets subset. The text is cleaned and a TDM is created for the fake disaster tweets, which contains the two attributes, “word” and “freq”. So, a word cloud will also be generated for fake disaster tweets TDM:



Figure 19: Word Cloud - Fake Disaster Tweets

A clear observation of the word cloud can tell us that the most common words are “body”, “love”, “people” and “video”, followed by “emergency”, “youtube”, and “time”. When comparing it to the real disaster word cloud, we can see that the word “people” is appearing in both word clouds, while the remaining are not. Then, we will plot the top 10 recurring words within the word cloud:



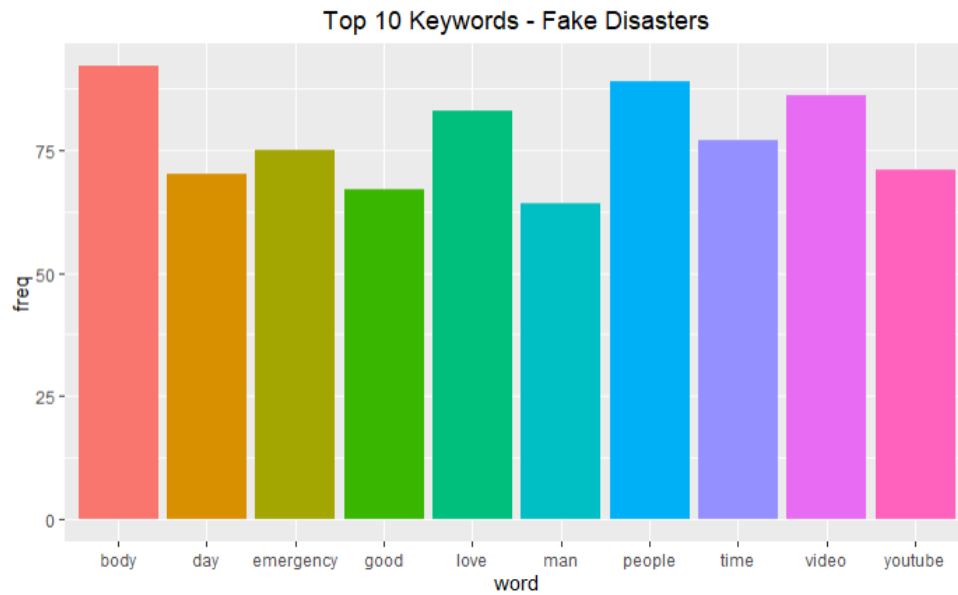


Figure 20: Top 10 Words - Fake Disasters

Also, we can see that the most recurring words in fake disaster does not exceed 100 time, while multiple words frequency passed 100 in the real disaster tweets, despite the fake disaster tweets being of more records than the real disaster tweets. Then, we will also plot the sentiment plot for the fake disaster, as it was done for the real disaster tweets:

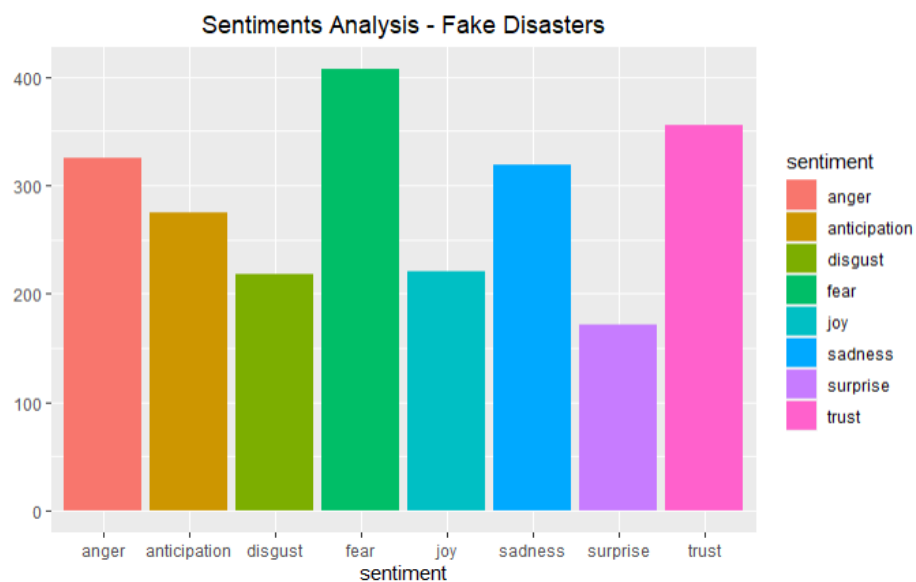


Figure 21: Sentiment Analysis for Fake Disaster Words

The sentiment plot shows that “fear” is the dominant sentiment, followed by “trust”, “anger”, and “sadness”, while “joy” and “surprise” are the least sentiments estimated. This is similar to the trend found in Sentiment plot for the real disaster tweets, which can tell us that sentiments analysis by itself cannot distinguish between if a certain word is expressed in a negative way or just been used as a metaphor, and this is expected where the method used just takes the TDM in consideration and not the whole tweet (context).

## 4.2 Data Pre-processing

In this section, after performing data exploration and visualization for the working data set, then comes the pre-processing part in the CRISP-DM process. In the previous section, we performed data pre-processing and cleaning while exploring the data set, and similar steps will be followed here prior to the model building part.

In any machine learning problem, data preprocessing is equally as important as model construction, and usually it requires more time and effort than any other step. Text data is one of the unstructured types of data accessible, and dealing with human language is quite challenging. Text preprocessing is a method for cleaning and preparing text data to be used in a model. In addition to other elements, text information incorporates elements such as sentiments, accentuation, and text in an alternate case. When it comes to Human Language, there are many ways to say the same thing (irony, humor, metaphor), and this is challenging to the machines, as they cannot understand words and context.

Data pre-processing for the text data will be as follow:

- Computer deciphers lower case and capitalized letters differently, and to avoid such problems, all word letters will be transformed to lower case,
- Remove words and numbers that contain digits: When words and digits are combined in a text, it creates a challenge for machines to comprehend. As a result, words and digits that are combined, such as game57 or game5ts7, should be avoided. Because this type of phrase is difficult to understand, it is preferable to discard it or replace it with an empty string.
- Stop words are the most common terms in a text that provide no useful information. Words like they, there, this, there, and others are included in stop words. A list of prevent words was acquired from the internet and utilized to eliminate the stop words. Different words, such as http and https that were previously omitted from our stop word list were manually added.
- Special characters that were present in the tweets which includes @, /, \$ and emoji's were removed from the dataset.
- Dataset is divided into two sets, one for training the model and another one for testing the model. The training dataset is used by model to learn while the performance of the trained model is evaluated on the testing set created in this step.

The following are the steps performed in RStudio:

```
```{r, warning=FALSE, echo=FALSE}

corpus0 <- df %>%
  select(text, target)

corpus0$text <- iconv(corpus0$text, 'UTF-8', 'ASCII')
corpus0 <- as.data.frame(corpus0)
corpus0 <- na.omit(corpus0)
corpus0$target <- factor(corpus0$target)

corpus <- corpus0

pander(table(corpus$target), caption="Fake and Real Count in Raw Data")
```
```

| 0    | 1    |
|------|------|
| 3994 | 2922 |

Table: Fake and Real Count in Raw Data

Figure 22: Data selection - modeling part

In the chunk above, “text” and “target” are the only feature selected, as we will train our output variable, “target”, based on the input variable, “text”, and we omitted any null values in “text” column, recalling figure 4, where we learned that the data set contains 697 null values in “text” column.

```

{r}
corpus2 <- Corpus(VectorSource(corpus$text))

toSpace2 <- content_transformer(function(x , pattern) gsub(pattern, " ", x))

corpus2 <- tm_map(corpus2, toSpace2, "/")
corpus2 <- tm_map(corpus2, toSpace2, "@")
corpus2 <- tm_map(corpus2, toSpace2, "\\|")

corpus2 <- tm_map(corpus2, content_transformer(tolower)) #lowercase
corpus2 <- tm_map(corpus2, removeWords, stopwords("english")) #removing stopwords
corpus2 <- tm_map(corpus2, removeWords, x) #remove imported stopwords
corpus2 <- tm_map(corpus2, stripWhitespace) #removing whitespaces
corpus2 <- tm_map(corpus2, removePunctuation) #remove punctuation
corpus2 <- tm_map(corpus2, removeNumbers) #remove numbers

corpus2 <- tm_map(corpus2,
 function(x) iconv(enc2utf8(x), sub = "byte"))

dtm <- DocumentTermMatrix(corpus2)

```

Figure 23: Text data cleaning

In figure 23, we performed data cleaning to the text data, changing all letters to lower case, removing stop words, whitespaces, punctuations, numbers, and symbols.

```

{r}
set.seed(1223)
train_idx <- createDataPartition(corpus$target, p=0.80, list=FALSE)

train1 <- corpus[train_idx,]
test1 <- corpus[-train_idx,]

train2 <- corpus2[train_idx]
test2 <- corpus2[-train_idx]

```

Figure 24: Data Partition

Then, a data partition was created with a ratio of 80/20 training to testing subset, and we performed the partition on the original and the cleaned subsets.

```

```{r}
dict2 <- findFreqTerms(dtm2, lowfreq=10)

news_train <- DocumentTermMatrix(train2, list(dictionary=dict2))
news_test <- DocumentTermMatrix(test2, list(dictionary=dict2))

convert_counts <- function(x) {
  x <- ifelse(x > 0, 1, 0)
}

news_train <- news_train %>% apply(MARGIN=2, FUN=convert_counts)
news_test <- news_test %>% apply(MARGIN=2, FUN=convert_counts)

news_train <- as.data.frame(news_train)
news_test <- as.data.frame(news_test)

news_train1 <- cbind(target=factor(train1$target), news_train)

news_test1 <- cbind(target=factor(test1$target), news_test)

news_train1<-as.data.frame(news_train1)
news_test1<-as.data.frame(news_test1)
```

```

Figure 25: Changing TDM to a data frame

Then, the training and testing subset corpuses are to be transformed to a data frame, in order to be ready to be used in the data modeling part.

Finally, after applying all the above data preprocessing steps, we got a cleaned data, a training, and a testing subset, ready to be used in the modeling part, the training dataset consist of 5,534 observations, while the testing dataset consist of 1,382 observations, of total of 1,023 variables, one is “target” while the rest are the TDM.


## 4.3 Data Modeling

In the modeling section, we will use the pre-processed training subset to train the machine learning model, and then we will build a confusion matrix to check how the model perform against the unseen testing subset.

### 4.3.1 Support Vector Machine (SVM) Classifier

Support Vector Machine (SVM) is a useful model that sorts data with different features. SVM applies linear function hypothesis in high dimensional space, it creates a  $(p-1)$  dimensional hyperplane in  $p$ -dimensional space, in order to separate classes. Also, SVM applies margin maximization, which is a technique to select the optimum hyperplane that splits classes. (James, et. al, 2013). SVM and Naive Bayes have been considered as one of the most successful and extensively used text classification methods among all classification approaches. SVMs have the unique quality of being able to learn regardless of the dimensionality of the feature space. This suggests that if our data is separable with a large margin using functions from the hypothesis space, we can generalize even in the presence of numerous features. The optimum parameter selection is one that yields the hypothesis with the smallest VC- Dimension. This eliminates the need for costly cross-validation and allows for entirely automated parameter tweaking (Z. Wang et. al, 2006).

In RStudio, we used the SVM function and confusion matrix, as seen in the following chunk:

A screenshot of the RStudio code editor. The code is written in a light blue font on a light gray background. It includes a comment line '### SVM Classifier', a line with curly braces '{}', and a line defining 'fit1' as an SVM model using 'target~.' and 'data=news\_train1'. The code ends with three backticks '```'.

```
SVM Classifier
```{r}

fit1 <- svm(target~., data=news_train1)
fit1
```
```

Figure 26: SVM Classifier

The model dependent output variable “target”, while the rest of the TDM are the independent variables. After building the model, using the prediction and confusion matrix, the accuracy was estimated:

```

```{r}
fit1.pred <- predict(fit1, news_test1)
```

```{r}
confMatrix1 <- confusionMatrix(fit1.pred, news_test1$target)
confMatrix1
```

Confusion Matrix and Statistics

 Reference
Prediction 0 1
0 722 206
1 76 378

 Accuracy : 0.7959
 95% CI : (0.7737, 0.8169)
 No Information Rate : 0.5774
 P-Value [Acc > NIR] : < 2.2e-16

 Kappa : 0.569

 Mcnemar's Test P-Value : 1.568e-14

 Sensitivity : 0.9048
 Specificity : 0.6473
 Pos Pred Value : 0.7780
 Neg Pred Value : 0.8326
 Prevalence : 0.5774
 Detection Rate : 0.5224
 Detection Prevalence : 0.6715
 Balanced Accuracy : 0.7760

 'Positive' Class : 0

```

Figure 27: SVM Confusion Matrix

We can see that the accuracy estimated through the confusion matrix is 79.59%.

### 4.3.2 K-Nearest Neighbor Classifier

One of the most fundamental machine learning algorithms is the K-Nearest Neighbor (KNN) algorithm. The algorithm's purpose is to classify objects into one of the machine-learned sample group's stated classes. Based on their mutual Euclidean distance, KNN identifies the most similar objects from a set of sample groups. The TF-IDF method is a numerical statistic that may be used to calculate the weighting of each term in a document. The approach's applications include NLP, information retrieval, and text mining. The method determines the weight, which is a statistic for determining the importance of phrases in a document collection. The text's prominence rises in direct proportion to the number of times it appears in the papers. The



learning step follows preprocessing and document preparation. The algorithm determines the core papers with which each new document will be compared. The algorithm analyses where a document belongs in the classification system by examining just the training documents that are the most comparable to it. The technique assumes that documents may be classified as points in Euclidean space. (B. Trstenjak, et. al 2013).

In RStudio, we used the KNN function and confusion matrix, as seen in the following chunks:

```
KNN Classifier
```{r}
classifier_knn <- knn(train = news_train1,
                     test = news_test1,
                     cl = news_train1$target,
                     k = 1)
```
```

Figure 28: KNN Classifier

```
```{r}
cm_knn <- table(news_test1$target, classifier_knn)
confusionMatrix(cm_knn)
```
```

Confusion Matrix and Statistics

| classifier_knn |     |     |
|----------------|-----|-----|
|                | 0   | 1   |
| 0              | 796 | 2   |
| 1              | 10  | 574 |

Accuracy : 0.9913  
95% CI : (0.9849, 0.9955)  
No Information Rate : 0.5832  
P-value [Acc > NIR] : < 2e-16

Kappa : 0.9822

McNemar's Test P-value : 0.04331

Sensitivity : 0.9876  
Specificity : 0.9965  
Pos Pred value : 0.9975  
Neg Pred value : 0.9829  
Prevalence : 0.5832  
Detection Rate : 0.5760  
Detection Prevalence : 0.5774  
Balanced Accuracy : 0.9921

'Positive' class : 0

Figure 29: KNN Confusion Matrix

We can clearly observe that the accuracy is 99.13%, which is very high and considered to be overfitted, and this is although we have selected the lowest possible value of  $k$ , which is one.

### 4.3.3 Naïve Bayes Classifier

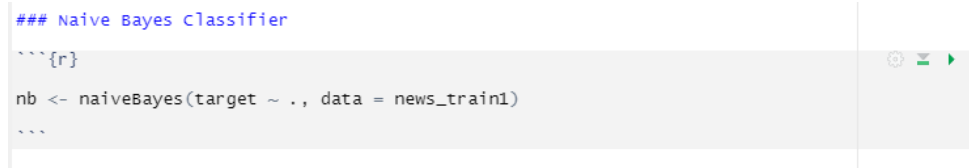
According to J. Chen, et al. (2009), Naïve Bayes model, which is based on classical mathematics theory, has a strong mathematical foundation and a consistent categorization efficiency. The model also has a wide range of applications because it only requires a few estimated parameters, is not overly sensitive to missing data, and is simple to execute. The Naive Bayes classifier may be built in two methods. The polynomial model MNB is one, while the multivariate Bernoulli model BNB, often known as the Binary independent model, is the other. The following are the primary changes between the two models:

- When representing a document, whether to take the frequency of occurrence of a phrase into account. The polynomial MNB model considers not only if a word appears, but also how frequently it appears, whereas the multivariate Bernoulli BNB model just considers whether lexical items appear.
- The words in the report do not appear to have differential arrangement effects. The obscure word thing does not play a role in classifying in the polynomial MNB model, but it should be shown in the multivariate Bernoulli BNB model, which means that the non-seeming word thing should be used as a factor in determining the back probability to determine the record classification.

Polynomial MNB classification approach provides the following benefits over multivariate Bernoulli model:

- When numerous essential characteristics are merged to create a categorization choice, Multinomial MNB can exhibit a higher classification result.
- The MNB is more resistant to idea drift and noise characteristic polynomial.
- In both the training and classification processes, the time complexity is linear. As a result, our research is based on the MNB algorithm.

In RStudio, we used the Naïve Bayes function and confusion matrix, as seen in the following chunks:

A screenshot of an RStudio code chunk. The code is written in a light gray editor with blue syntax highlighting. It starts with a comment '### Naive Bayes Classifier', followed by a blank line, then an R comment line '##{r}', then the code 'nb <- naiveBayes(target ~ ., data = news\_train1)', and ends with another blank line and an R comment line '##'. To the right of the code is a vertical toolbar with icons for running, refreshing, and other actions.

```
Naive Bayes Classifier

##{r}
nb <- naiveBayes(target ~ ., data = news_train1)

##
```

Figure 30: Naive Bayes Classifier

```

```{r}
y_pred <- predict(nb, newdata = news_test1)
cm_nb <- table(news_test1$target, y_pred)
confusionMatrix(cm_nb)
```

```

Confusion Matrix and Statistics

|   |     | y_pred |     |  |
|---|-----|--------|-----|--|
|   |     | 0      | 1   |  |
| 0 | 744 | 54     |     |  |
|   | 1   | 432    | 152 |  |

Accuracy : 0.6483  
 95% CI : (0.6225, 0.6735)  
 No Information Rate : 0.8509  
 P-Value [Acc > NIR] : 1  
  
 Kappa : 0.2109  
  
 Mcnemar's Test P-Value : <2e-16  
  
 Sensitivity : 0.6327  
 Specificity : 0.7379  
 Pos Pred Value : 0.9323  
 Neg Pred Value : 0.2603  
 Prevalence : 0.8509  
 Detection Rate : 0.5384  
 Detection Prevalence : 0.5774  
 Balanced Accuracy : 0.6853  
  
 'Positive' class : 0

Figure 31: Naive Bayes Confusion Matrix

We can clearly observe that the accuracy is 64.83%, which is very low.

#### 4.3.4 XGBoost Classifier

XGBoost stands for eXtreme Gradient Boosting, and it is an open source supervised scalable machine learning method that try to combine multiple weaker models, to achieve a better model. According to Chen, et al. (2016), XGBoost is widely used by data scientists in solving machine learning problem. They claim that it is been the dominant machine learning algorithm used in competition in Kaggle Website, including a problem of Text Web Classification.

In RStudio, we used the XGBoost function and confusion matrix, as seen in the following chunks:

```

XGBOOST

```{r}
train_control <- trainControl(method = "cv",
                             number=3,
                             verboseIter = TRUE,
                             allowParallel = TRUE)
```

```{r}
boosting1<- train(target~.,
                  data = news_train1,
                  method = "xgbTree",
                  trControl = train_control,
                  tuneGrid = expand.grid(nrounds = 500,
                                       max_depth = 3,
                                       eta = 0.2,
                                       gamma = 2.1,
                                       colsample_bytree = 1,
                                       min_child_weight = 1,
                                       subsample = 1))
```

```

Figure 32: XGBoost Classifier

```

```{r}
boost_pred <- predict(boosting1, newdata = news_test1)
cm_boost <- table(news_test1$target, boost_pred)

confusionMatrix(cm_boost)
```

```

Confusion Matrix and Statistics

|   | boost_pred |     |
|---|------------|-----|
|   | 0          | 1   |
| 0 | 726        | 72  |
| 1 | 233        | 351 |

Accuracy : 0.7793  
 95% CI : (0.7565, 0.8009)  
 No Information Rate : 0.6939  
 P-value [Acc > NIR] : 7.895e-13

Kappa : 0.5304

Mcnemar's Test P-value : < 2.2e-16

Sensitivity : 0.7570  
 Specificity : 0.8298  
 Pos Pred Value : 0.9098  
 Neg Pred Value : 0.6010  
 Prevalence : 0.6939  
 Detection Rate : 0.5253  
 Detection Prevalence : 0.5774  
 Balanced Accuracy : 0.7934

'Positive' class : 0

Figure 33: XGBoost Confusion Matrix

The confusion matrix above shows that XGBoost model scored an accuracy of 77.93%.

## 4.4 Results

In the modeling part, we used four supervised machine learning models, and they are SVM, KNN, Naïve Bayes, and XGBoost. In the first trial, the training and testing split was 80/20, and we used the original subset which had more data consisting of one class over the other. In that trial, SVM and XGBoost scored a good accuracy of 79.6% and 77.9% respectively, while KNN accuracy was too high and became overfitted, and Naïve Bayes accuracy was too low (65%). Furthermore, we performed three more trials for all four models, we tried to sample equal observation for each class and ran the model. Then, we changed the split to 75/25 and ran the model based on the original subset, and on an equal amount of sampled observation of both classes. Table 2 outlines all the results accuracies:

| Model       | 80/20 Split     |                | 75/25 Split     |                |
|-------------|-----------------|----------------|-----------------|----------------|
|             | Original Subset | Sampled Subset | Original Subset | Sampled Subset |
| SVM         | 0.7959          | 0.768          | 0.7766          | 0.7774         |
| KNN         | 0.9913          | 0.994          | 0.9902          | 0.9979         |
| Naïve Bayes | 0.6483          | 0.601          | 0.6279          | 0.6055         |
| XGBoost     | 0.7793          | 0.7603         | 0.7703          | 0.7692         |

Table 2: Model Accuracies Results

It is clear that among the four different trials conditions. The first condition, 80/20 split, using the original dataset rather than an equal sample dataset, is better than all other models. To explore more, below is full performance table for that trial:

| <b>Model</b>       | <b>Accuracy</b> | <b>Kappa</b> | <b>Sensitivity</b> | <b>Specificity</b> |
|--------------------|-----------------|--------------|--------------------|--------------------|
| <b>SVM</b>         | 0.7959          | 0.569        | 0.9048             | 0.6473             |
| <b>KNN</b>         | 0.9906          | 0.9807       | 0.9876             | 0.9948             |
| <b>Naïve Bayes</b> | 0.6483          | 0.2109       | 0.6327             | 0.7379             |
| <b>XGBoost</b>     | 0.7793          | 0.5304       | 0.7570             | 0.8298             |

Table 3: Trial 1 Model Performance Metrics

In term of accuracy, KNN has the highest accuracy, but it is too high it became overfitted, then SVM and XGBoost have reasonable accuracies of approximately 79.6% and 78.0% respectively, while Naïve Bayes accuracy is very low at 65%.

Kappa is a metric that can be used to compare classifiers to each other, it ranges between 0 and 1, the higher better the classifier, the higher is Kappa. Kappa values follow similar behavior as accuracies, so the models are sorted in the same way.

Sensitivity is the percentage of positive that was correctly predicted as positive (TP) divided by the total number of positive (P). In our problem, it means that we are dividing the number of correctly identified tweets as real disaster tweets (TP), to the total number of real disaster tweets (True Positive/False Negative). By looking at table 3, we can see KNN obviously have the highest percentage of true positive prediction because of the overfitting, and Naïve Bayes have the lowest because of the poor performance of the model. However, when comparing SVM and XGBoost, they both have close accuracies, while in term of sensitivity, SVM performs better with

a value of 0.9048 while XGBoost value is 0.7570, this means that SVM is better in term of identifying the positive predictions.

Specificity is a measure of the true negative rate, so it is the fraction of true negative (TN) to the total number of negatives (N), which consist of both true negative (TN) and false negative (FN). As expected, KNN will have the highest specificity. Unlike sensitivity, here XGBoost classifier performed better than SVM, with a value of almost 0.83 while SVM value is 0.65. One interesting finding is the value of Naïve Bayes, although it had the lowest accuracy, kappa, and sensitivity, the value of it here is 0.74, so in term of true negative rate, it performed better than SVM.

To sum up, we produced one model that is overfitted (KNN), and one model with poor performance (Naïve Bayes), while SVM and XGBoost had reasonable accuracies between 78-80%, SVM is performing better in term of sensitivity (true positive rate), while XGBoost is performing better in term of specificity (true negative rate).



# Chapter 5: Conclusion

## 5.1 Conclusion

In conclusion, the project aim was to try and build a model that can classify whether a piece of text is referring to a real disaster or not, which eventually can help in identifying real disaster and filtering out fake ones, in order to help for a better response for the concerned team. The literature review covered how text analysis and classification is approached and the best techniques to be followed. The project demonstrated a realistic way to try and detect suspicious piece of text, as a binary machine learning problem. The main issue facing the machine is to be able to identify whether certain text is referring to a real disaster or not. Furthermore, the working dataset contained labelled tweets, whether they are referring to real disaster or not. CRISP-DM methodology was followed (expect for deployment phase), business understanding in our case refers to the importance of the problem we are trying to solve, then we performed data exploration, data pre-processing, data modeling and evaluation. Finally, we successfully built four supervised machine learning models, and they are KNN, SVM, XGBoost, and Naïve Bayes, the accuracies were 99%, 80%, 78%, and 65% respectively, and we concluded that the first model was overfitted, while the last one performed poorly. Also, we observed that although SVM and XGBoost performed reasonably and had accuracies of 80% and 78%, the first one was better in identifying true positive, while the latter is better in identifying true negatives.

Performances vary because of different reasons, the method used in the model building was building a TDM and build the model based on it, which does not take the context of a sentence

in consideration. The problem here is again, it will be hard for the machine to identify whether a certain word will be used with a literal meaning, or used metaphorically.

## 5.2 Recommendations

To improve the performance of the classifiers, other approaches to the text mining and classification should be tried. For instance, the method used here is to clean the text data, then creating a corpus and a TDM that contains all the remaining word and to build the model based on the relationship between the TDM and the target variable. To extend, other methods that takes context into consideration should be tested. For example, a new variable can be created which counts the number of words per tweet or number of characters, and try to relate it to the output variable. Moreover, the used model, such as XGBoost, can undergoes parameters tuning which might yield a higher accuracy, one challenge was the time it takes to build the model. Also, other models can be tested such as neural network in Keras packages and compare its result with other models.

# References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R.J. (2011). Sentiment Analysis of Twitter Data.
2. M. F. Çelikutğ, "Twitter Sentiment Analysis, 3-Way Classification: Positive, Negative or Neutral?," *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2098-2103, doi: 10.1109/BigData.2018.8621970.
3. Natural Language Processing with Disaster Tweets | Kaggle. (2021). Retrieved 30 November 2021, from <https://www.kaggle.com/c/nlp-getting-started/data>
4. Vasudha Rani, V., & Sandhya Rani, K. (2016). Twitter Streaming and Analysis through R. *Indian Journal Of Science And Technology*, 9(45). doi: 10.17485/ijst/2016/v9i45/97914
5. Samuels, Antony & McGonigal, John. (2020). News Sentiment Analysis.
6. Patterson, K. 2013. The identification of metaphor using corpus methods: Can a re-classification of metaphoric language help our understanding of metaphor usage and comprehension? Paper presented at the 7th International Corpus Linguistics Conference (CL2013), Lancaster University, Lancaster.
7. Ghosh, Aniruddha & Li, Guofu & Veale, Tony & Rosso, Paolo & Shutova, Ekaterina & Reyes, Antonio & Barnden, John. (2015). SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. 10.18653/v1/S15-2080.
8. Madichetty, S., M., S. A stacked convolutional neural network for detecting the resource tweets during a disaster. *Multimed Tools Appl* 80, 3927–3949 (2021).
9. Arun, K., Srinagesh, A., & Ramesh, M. (2017). Twitter Sentiment Analysis on Demonetization tweets in India Using R language.
10. Nikam, Shivani & Dalvi, Rupali. (2020). Machine Learning Algorithm based model for classification of fake news on Twitter. 1-4. 10.1109/I-SMAC49090.2020.9243385.

11. Magliani, Federico & Fontanini, Tomaso & Fornacciari, Paolo & Manicardi, Stefano & Iotti, Eleonora. (2016). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter.
12. Rubin, V.R., Conroy, N. J., Chen, Y. & Cornwell, S. (2016) Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News.
13. Horne, B., & Adali, S. (2017, May). This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the international AAAI conference on web and social media* (Vol. 11, No. 1, pp. 759-766).
14. Conroy, N., Rubin, V., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings Of The Association For Information Science And Technology*, 52(1), 1-4. doi: 10.1002/pr2.2015.145052010082
15. Chen, Y., Conroy, N. J., & Rubin, V. L. (2015, November). Misleading online content: Recognizing clickbait as false news. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection* (pp. 15-19). ACM
16. Zhang, J., Cui, L., Fu, Y., & Gouza, F.B. (2018). Fake News Detection with Deep Diffusive Network Model. *ArXiv, abs/1805.08751*.
17. Cobo, Alfredo & Parra, Denis & Navon, Jaime. (2015). Identifying Relevant Messages in a Twitter-based Citizen Channel for Natural Disaster Situations. 1189-1194. 10.1145/2740908.2741719.
18. Ashktorab, Z., Brown, C., Nandi, M., & Culotta, A. (2014). Tweedr: Mining twitter to inform disaster response. *ISCRAM*.
19. Imran, Muhammad & Mitra, Prasenjit & Castillo, Carlos. (2016). Twitter as a Lifeline: Human-annotated Twitter Corpora for NLP of Crisis-related Messages.
20. Hörtenhuemer, C., & Zangerle, E. (2020). A Multi-Aspect Classification Ensemble Approach for Profiling Fake News Spreaders on Twitter. *CLEF*.

21. Kumar A, Singh JP, Dwivedi YK et al (2020). A deep multi-modal neural network for informative Twitter content classification during emergencies. *Annals of Operations Research*.
22. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R* (pp. 337-343).
23. Z. Wang, X. Sun, D. Zhang, (2006), An Optimal SVM-Based Text Classification Algorithm.
24. B. Trstenjak, S. Mikac, D. Donko (2013), KNN with TF-IDF Based Framework for Text Categorization.
25. Chen, Tianqi & Guestrin, Carlos. (2016). XGBoost: A Scalable Tree Boosting System. 785-794. 10.1145/2939672.2939785.