Real or Fake News? Using NLP with Disaster Tweets

Lim Zheng Wei

INSTITUTE OF DATA Capstone Project

Abstract

More people are spending time on the internet and social media which leads to more reliance getting news online. The evolution of communication also gave rise to fake news. The motivations behind fake news include making profits, political influence, and propaganda. In addition, people can post easily, there is no verifying the authenticity of the news. Because people shape their opinions based on information that they receive, having fake news can impact lives on a financially, socially, politically, and more.

This report presents an experimental study on detecting real and fake disasters by classifying Twitter texts based on Natural Language Processing (NLP) and classifiers. The experiment is conducted on Python Jupyter notebooks. To evaluate the effectiveness of the classifiers, the accuracy, precision, recall and receiver operation characteristics (ROC) and area under curve (AUC) was studied. The dataset was obtained from Kaggle, pre-processing and text cleaning occurred before splitting the texts into training and test set. 20% of the tweets were used as a control group, and the remaining data were utilised as a training set to train the model and evaluate the effects between different classifiers and detecting real and fake disasters.

The best model to classify the tweets is using Regularised model from Keras, with an accuracy score of 92.7%, a precision score of 94.6%, a recall score of 88.5% and a ROC_AUC score of 92.2%.

Acknowledgment

The author would like to take this opportunity to extend his gratitude to his trainers Dr. Tham and Mrs. Nastaran Faraji for their guidance and valuable inputs during the entire duration of this project, as well as arranging for the necessary resources required for the capstone project.

The author would like to convey his deepest appreciation to his fellow course mates Ms Danielle Chua, Mr. Jeremy Lee, Mr. John Choo, and Mr. Max Se for asking insightful questions to the trainers during class, which in turn benefitted everyone. They also made this course enjoyable as well.

The author would like to take the chance to thank the staffs of the Institute of Data, namely Mr. Andrew Campbell, Ms Jessica Haines for the guidance relating to non-data science segments, as well as Mr. Herman Fung and Ms Jessica Tochtermann with regards to administrative issues, ensuring a smooth and enjoyable experience with the course.

The author would also like to thank his family and friends for their support and understanding during the course as he was occupied by the assignments and presentations.

Table of Contents

Tables of Figures

List of Tables

Appendix

1. Introduction

The Internet’s history goes back to a few decades ago, emails existed in the 1960s, file sharing in the 1970s. However, it was the creation of internet in 1989 that changed how humans communicate forever when English scientist Tim Berners-Lee created a system to share information through a network of computers.

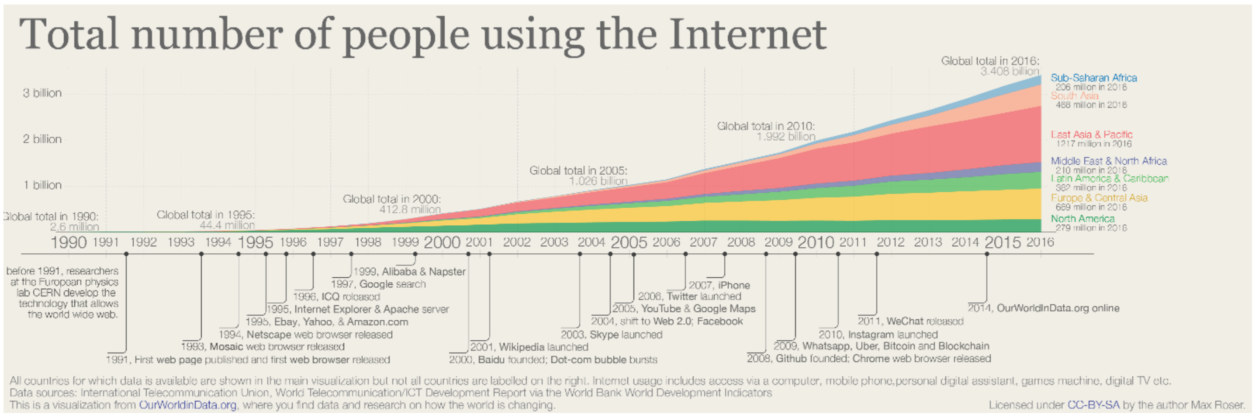


Figure 1 Total Number of People using the Internet [1]

Figure 1 showed the history of the number of Internet users since its creation. Only about 0.5% of the world population were online in 1990. 26 years later, over 3 billion people were utilising the net [1].

With more usage of the internet came the rise of social media. In 2004, MySpace became the first social media site to reach a million active users. Fast forward to 2018 and Facebook was the most popular social media platform with 2.3 billion users [2]. Social media was a big reason for the human’s increasing digitisation of lifestyle.

With people spending more time online, they were also more reliant on using online platform to receive news and information. A survey conducted by Reuters Institute showed that people aged between 18 and 34 were more likely to get their news from their devices. From those groups of people, they were more likely to read the news via social media first [3]. Because of how easy people could post online without need any verification, others could perceive fake news as legitimate. A survey conducted showed that 4 in 5 Singaporeans were confident in spotting fake news, but 91% of them were wrongly classified a least 1 fake news as genunine [4]. Fake news could ruin a person or entity’s reputation, or cause unnecessary alarm to the public. Therefore, there is a need to filter out what is real and what is not.

This project is targeted to differentiate between real and fake news using Natural Language Processing (NLP). The experiment was conducted during the dataset found at Kaggle, using different classifiers to separate the data. The efficiency of the classifier will be determined by the accuracy score, recall score, precision score and ROC_AUC score.

1.1 Social Media Statistics in Singapore

Singapore is one of the countries with the most advanced technology in the world. Indeed, Singaporeans spend most of their time on social media platforms, and the two have come hand in hand.

For that reason, approximately 88% of the population have access to the Internet and spends an average of 6 hours and 48 minutes every day online, of which 2 hours and 8 minutes are spent on social media. This means that Singaporeans spend nearly 40% of their lives on the Internet since the average sleeping time is 8 hours. The ease of access to the internet allows them to engage in content sharing, online shopping, access gaming sites, and social media platforms at a younger age.

As of January 2020, there are more than 4.6 million active social media users. Most Singaporeans engage in social networking or search for information using their smartphones. As a result, the number of scammers, impersonators, and cyber-attacks has jumped nine folds in the last three years [5].

2. Literature Review

2.1 Fake News

2.1.1 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 [6]. 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 [7]. 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.

For a detailed information on fake news such as motivations, refer to "Real of Fake News Using NLP with Disaster Tweets Addendum".

2.1.2 Implications of Fake News

[8]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.

To find out how fake news impact lives, how to identify them and current mitigation methods refer to “Real of Fake News Using NLP with Disaster Tweets Addendum”.

2.2 Feature Engineering

[9]Feature engineering is a process of transforming the given data into a form which is easier 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 process of making data transparent for the machine learning models can be a hussle as different models often require different approaches for the different kinds of data.

The text must be parsed to remove certain words before doing predictive modelling – this process is named tokenization. These words have to be encoded as floating-point values or integers before they can be used in machine learning algorithms. This process is called feature extraction (or vectorisation).

For more information on Feature Engineering, refer to “Real of Fake News Using NLP with Disaster Tweets Addendum”.

2.3 Text Classification

[10]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 hare broadly grouped into three different categories:

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

To know more about the systems, as well the algorithms for text classification, refer to "Real of Fake News Using NLP with Disaster Tweets Addendum".

3 Methodology

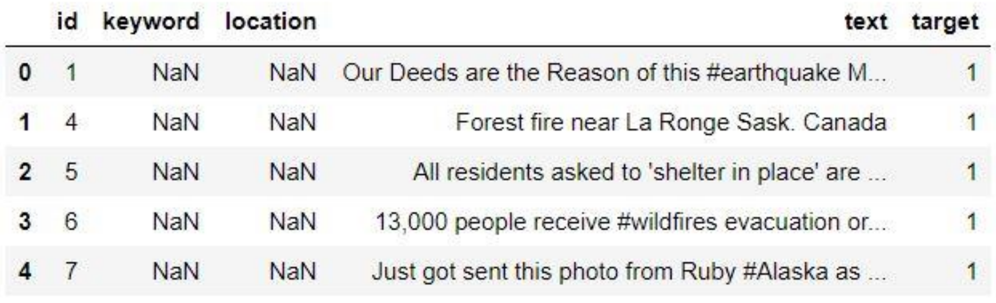
3.1 The Setup

This project is inspired by one of the competitions on Kaggle. The dataset was obtained from their website [11]. The experiments were carried out using Python version 3.7.8. More details of the setup can be found in "Real of Fake News Using NLP with Disaster Tweets Addendum".

3.2 Experimental Design

3.2.1 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 2 The first 5 columns of dataset

From Figure 2, 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 3. Next, the dataset was investigated deeper by finding out the total amount missing values in all column.

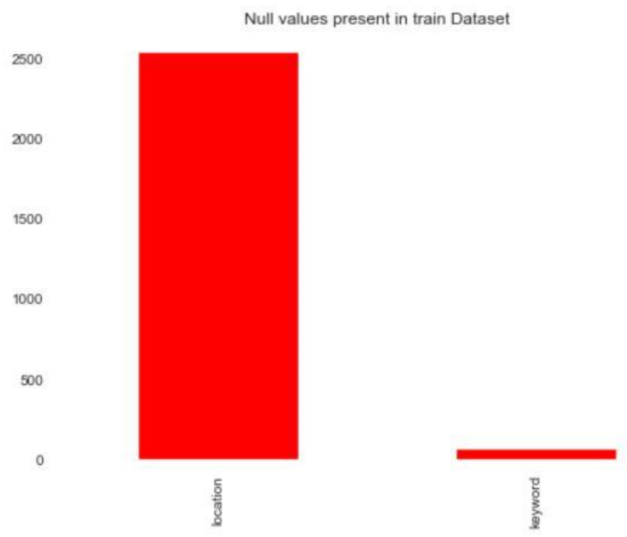


Figure 3 Null Values in the Dataset

From Figure 3, 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.

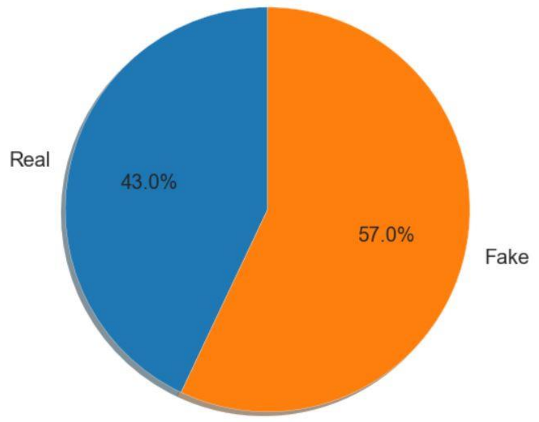


Figure 4 Proportion of Real and Fake Disaster

There was a total of 4342 fake disasters and 3271 real disasters. Figure 4 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%.

3.2.2 Basic Feature Extraction

Basic feature extractions 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.

For more in-depth analysis, refer to "Real of Fake News Using NLP with Disaster Tweets Addendum".

3.2.3 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.

To view all the steps taken to clean text, refer to "Real of Fake News Using NLP with Disaster Tweets Addendum".

3.2.4 WordCloud

In the previous section, text cleaning was conducted. Using WordCloud can help to visualise words associated with real and fake disasters in Figure 5 and Figure 6 respectively.

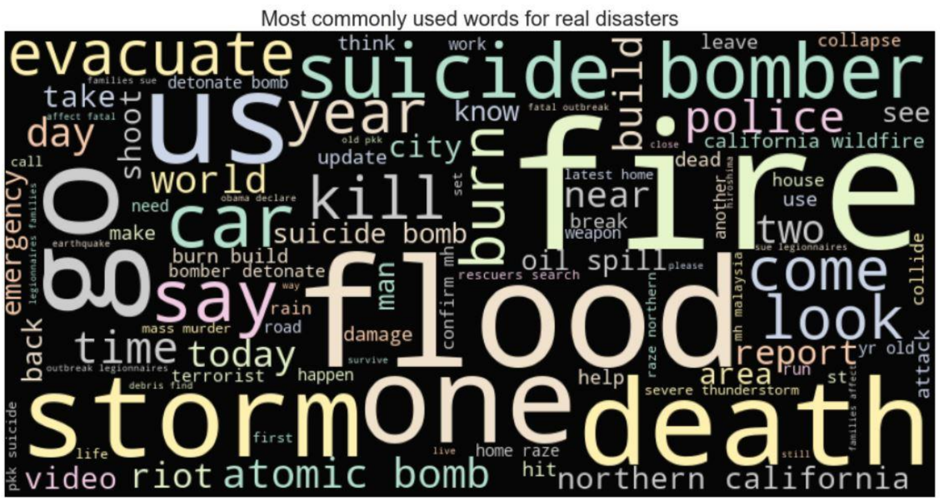


Figure 5 Most Commonly Used Words for Real Disasters

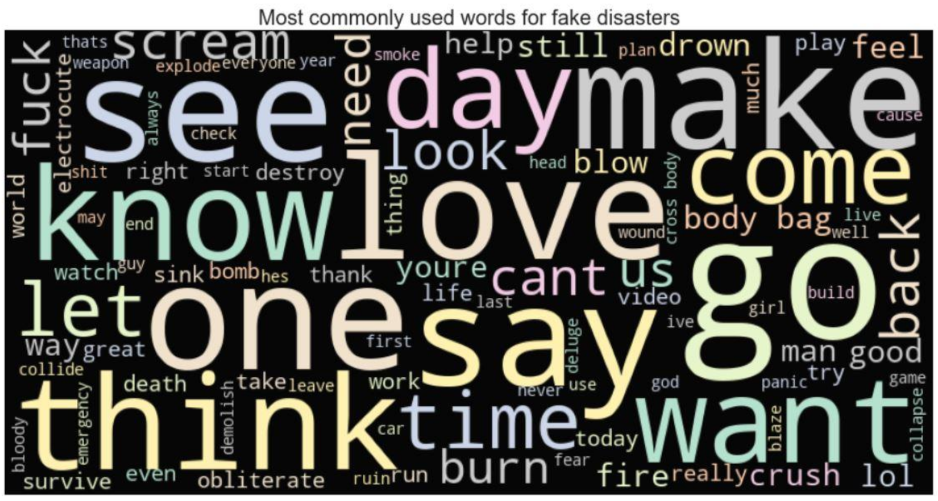


Figure 6 Most Commonly Used Words for Fake Disasters

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

3.2.5 Feature Engineering

Refer to "Real of Fake News Using NLP with Disaster Tweets Addendum" for all the steps taken such as Feature Labels.

4. Results

For data visualisation of the results, refer to the Appendix.

4.1 Naïve Bayes

	CountVectoriser	TF-IDF Word	TF-IDF N-Gram
Accuracy	0.810	0.821	0.750
Precision	0.806	0.856	0.885
Recall	0.734	0.702	0.480
ROC_AUC	0.863	0.865	0.747

Table 1 Naive Bayes

The results for Naïve Bayes were shown in Table 1. In general, all 3 models were performing well. The exception was the TF-IDF N-Gram which the recall score was suboptimal. However it has the highest precision score. TF-IDF word has the best accuracy and ROC score, while the CountVectoriser has the highest recall.

4.2 Linear Classifier

	CountVectoriser	TF-IDF Word	TF-IDF N-Gram
Accuracy	0.807	0.814	0.752
Precision	0.805	0.845	0.892
Recall	0.726	0.694	0.482
ROC_AUC	0.857	0.866	0.748

Table 2 Linear Classifier

From Table 2, generally all 3 models can classify accurately, with TF-IDF word the best with a score of 0.814. All 3 models are also precise, with TF-IDF N-Gram the highest at 0.892. As for recall, the models perform moderately well with TF-IDF N-Gram the lowest at 0.482. TF-IDF word has the highest ROC score at 0.866.

4.3 Support Vector Classifier

	CountVectoriser	TF-IDF Word	TF-IDF N-Gram
Accuracy	0.768	0.814	0.744
Precision	0.738	0.825	0.871
Recall	0.711	0.720	0.476
ROC_AUC	0.832	0.858	0.728

Table 3 Support Vector Classifier

With reference to Table 3, all 3 models were mainly accurate, with TF-IDF Word the best with a score of 0.814. All 3 were precise also, TF-IDF N-Gram the highest at 0.871. CountVectoriser and TF-IDF word did fine for the recall, but TF-IDF N-Gram was the worst with 0.476. All 3 had a relatively high score for ROC curve, with TF-IDF word at the top with 0.858.

4.4 Bagging Models (Random Forest)

	CountVectoriser	TF-IDF Word	TF-IDF N-Gram
Accuracy	0.776	0.793	0.741
Precision	0.758	0.793	0.840
Recall	0.703	0.702	0.491
ROC_AUC	0.834	0.845	0.743

Table 4 Random Forest

Assessing from Table 4, all 3 models performed well in the accuracy category, with TF-IDF word the highest with 0.793. TF-IDF N-Gram has the highest precision with 0.840 but has the lowest recall with 0.491. TF-IDF word has the highest ROC curve.

4.5 Boosting Models (Gradient Descent)

	CountVectoriser	TF-IDF Word	TF-IDF N-Gram
Accuracy	0.753	0.762	0.690

Precision	0.868	0.861	0.925
Recall	0.502	0.532	0.303
ROC_AUC	0.820	0.809	0.672

Table 5 Gradient Descent

Judging from Table 5, all 3 models fared relatively well in terms of accuracy, with TF-IDF word the best with 0.762. All 3 performed very well in the precision category, with TF-IDF N-Gram the best with 0.925. Just like the previous results, TF-IDF N-Gram also scored the worst in recall in 0.303. It was worth noting that the other 2 models did not score well for recall as well. CountVectoriser was the best performer for ROC with 0.820.

4.6 Keras

	Baseline	**Reduced**	**Regularised**	**Dropout**
Accuracy	0.897	0.893	0.920	0.891
Precision	0.915	0.916	0.947	0.911
Recall	0.844	0.834	0.868	0.834
ROC_AUC	0.891	0.887	0.915	0.885

Table 6 Keras

Looking at Table 6, generally all 4 models did very well, with the average scores of every category higher than any other classifiers from the previous 5 sections. Regularised leads all models for every category.

5. Discussions

Looking at all the machine learning methods, both CountVectoriser and TF-IDF classify real and fake disasters relatively well. TF-IDF word fare the best in terms of accuracy and ROC score, TF-IDF N-Gram has the best precision and the worst recall score. This shows that TF-IDF is a better feature extractor. This is an expected outcome considering how CountVectoriser and TF-IDF works. TF-IDF N-Gram can pick up true positive, group real disasters well but misclassify too many fake disasters as genuine, which explains the high precision score and low recall score across all classifiers. The disparity could be explained by looking at the WordClouds in Figure 5 Most Commonly Used Words for Real Disasters

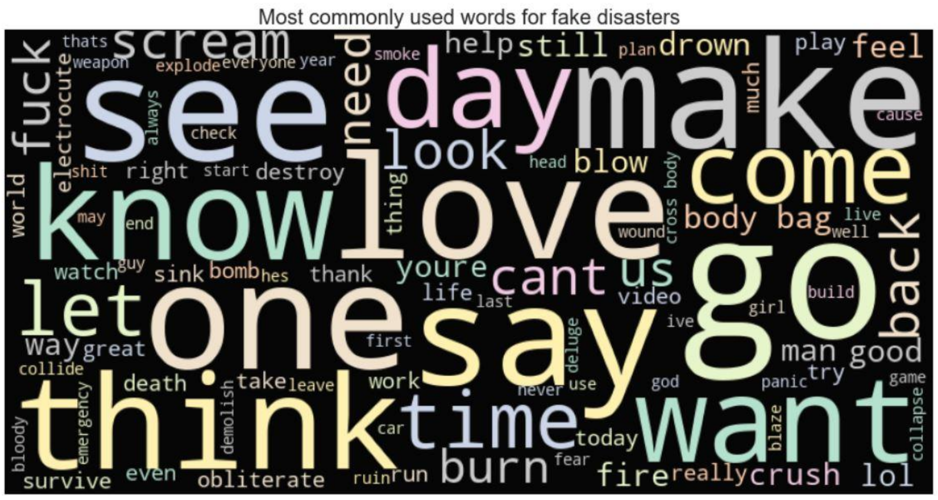


Figure 5 and Figure 6. Looking at the words from Figure 5, some of the words were meant to come together. Words such as “suicide bomber”, “severe thunderstorm”, “suicide bomb” make sense when they are together, explaining why TF-IDF N-Grams have high precision score. Some words appeared in both figures such as “fire”, “think”, “go”, “collapse”, “death”, which might cause the classifier to misclassify.

While comparing the 4 models based on Keras, all of them performed better than all the machine learning methods in all aspects. This is reflective to how deep learning works, for which they learn from their own mistakes. Considering that they are many more texts out there in the world and a limited amount of data for the experiment, this shows deep learning is the better than machine learning when it comes to classifying real and fake news.

5.1 Word Importance

Based on Figure 5 and Figure 6, Table 7 shows us the likelihood of the authenticity of disaster according to these set of words.

	Words
Fake News	Think, One, Say, Go, Love, See, Day, Make, Want, Time, Come, Know
Real News	Evacuate, Suicide, Bomber, Flood, Storm, Riot, Kill, Emergency, Northern California, Wildfire, Shoot, Damage, Severe, Thunderstorm

Table 7 Word Importance

6. Conclusion

In this study, Python Jupyter notebooks were used to execute experiments to evaluate the effects of different classifiers on grouping real and fake disaster tweets. This study aims to investigate how different classifiers segregate the data into real and fake news. The data were being pre-processed and undergo text cleaning before proceeding to conduct the experiment. One-fifth of the train set was assigned to be the control group, while the remaining 80% was tasked to train the model. The key criteria to assess the effectiveness of a classifier are accuracy, precision, recall and ROC_AUC score.

In the tests conducted, comparing Naïve Bayes classifier, TF-IDF were found to have a better score across the 4-evaluation metrics except recall score for TF-IDF N-Gram. The same trend was also observed across the other 4 machine learning methods. The 2 WordClouds that depicted the most commonly used words for real and fake disaster might explain the findings for high precision but low recall score by TF-IDF N-Gram. As for Keras, all 4 models did well in all 4 criteria, showing that deep learning is the way to go when classifying fake and real news.

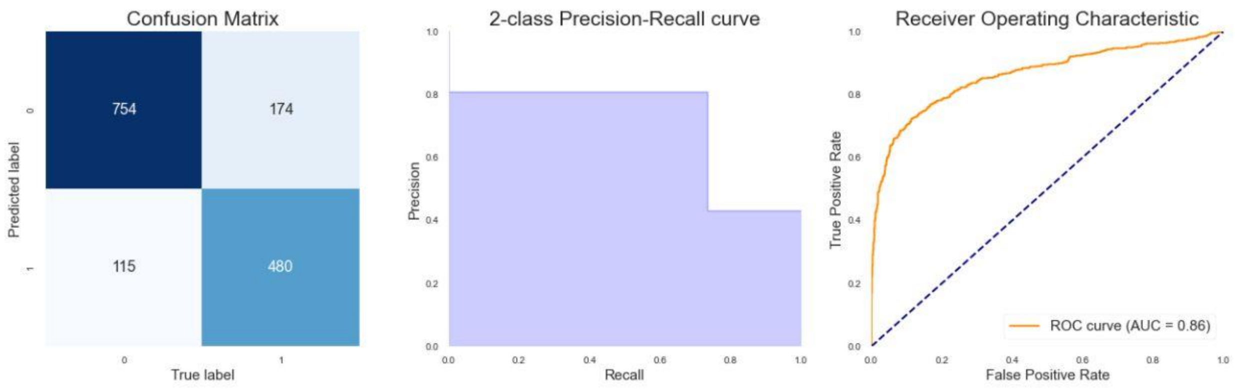
For future work, this study can be conducted using different deep learning methods such as Bert since only 1 deep learning method is being used.

Appendix

Appendix 1 NB CountVectoriser

```
show_summary_report(y_test, predictions_A1, predict_proba_A1)
```

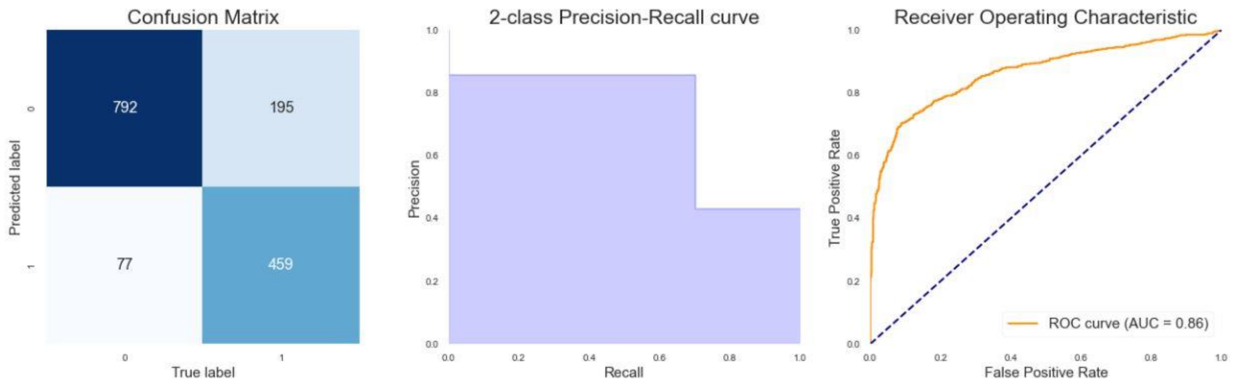
```
Accuracy : 0.8102 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8067 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7339 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8628                                                                      Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 2 NB TF-IDF

```
show_summary_report(y_test, predictions_A2, predict_proba_A2)
```

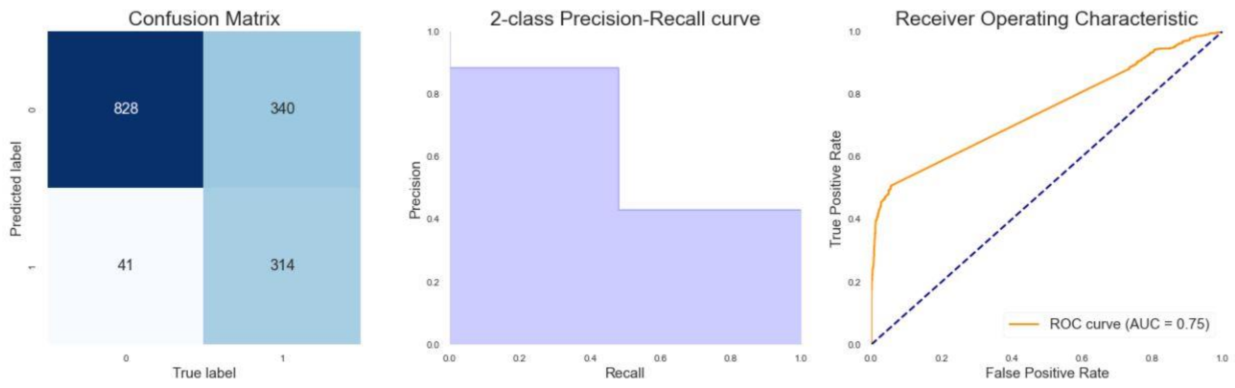
```
Accuracy : 0.8214 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8563 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7018 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8645                                                                      Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 3 NB TF-IDF N-Gram

```
show_summary_report(y_test, predictions_A3, predict_proba_A3)
```

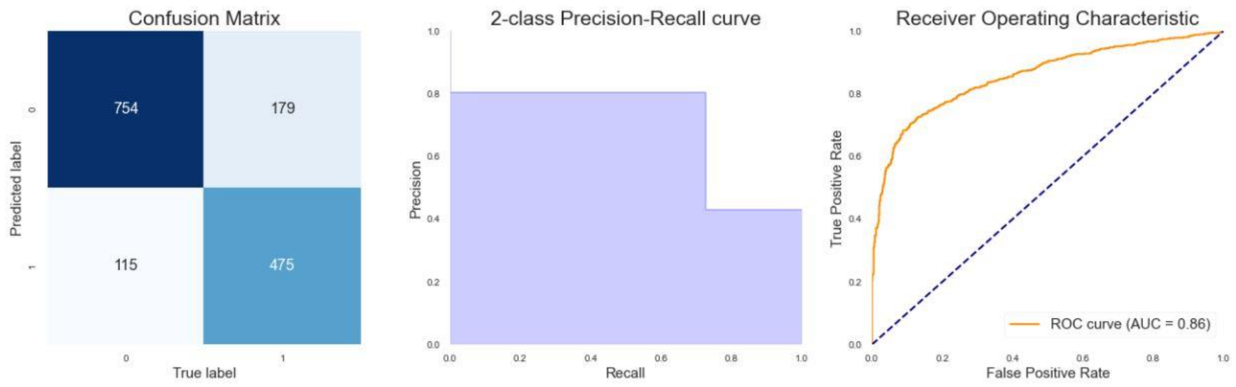
```
Accuracy : 0.7498 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8845 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.4801 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.7473                                                                     Best: 1, Worst: < 0.5
--------------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 4 Linear CountVectoriser

```
show_summary_report(y_test, predictions_B1, predict_proba_B1)
```

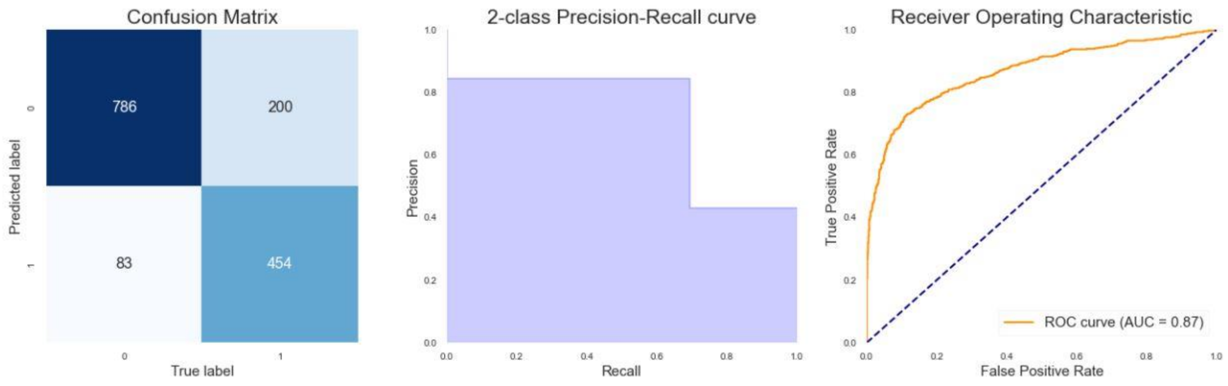
```
Accuracy : 0.8070 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8051 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7263 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8570                                                                     Best: 1, Worst: < 0.5
--------------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 5 Linear TF-IDF

```
show_summary_report(y_test, predictions_B2, predict_proba_B2)
```

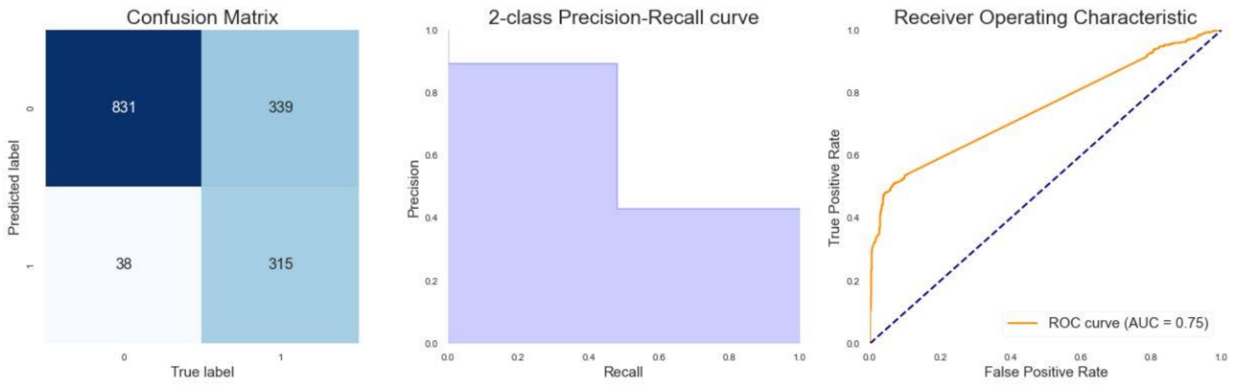
```
Accuracy : 0.8142 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8454 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.6942 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8665                                                                       Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 6 Linear TF-IDF N-Gram

```
show_summary_report(y_test, predictions_B3, predict_proba_B3)
```

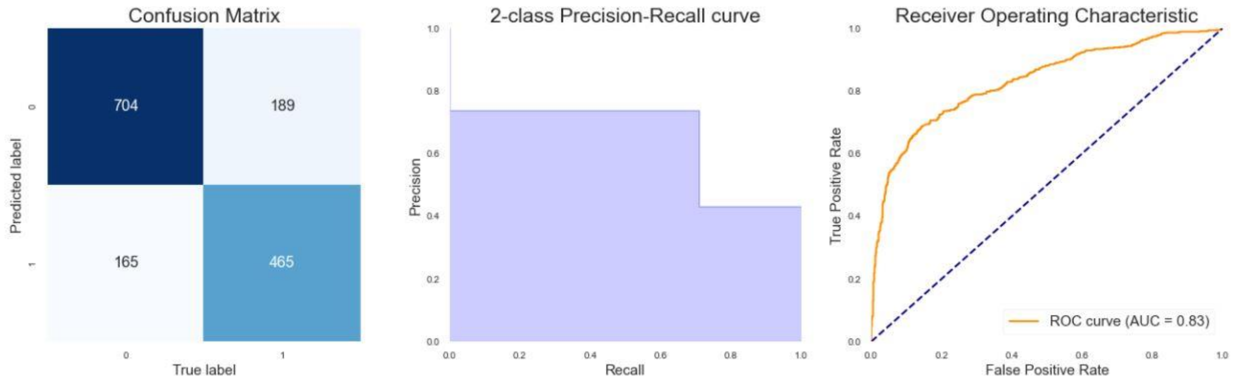
```
Accuracy : 0.7525 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8924 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.4817 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.7485                                                                       Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 7 SVC CountVectoriser

```
show_summary_report(y_test, predictions_C1, predict_proba_C1)
```

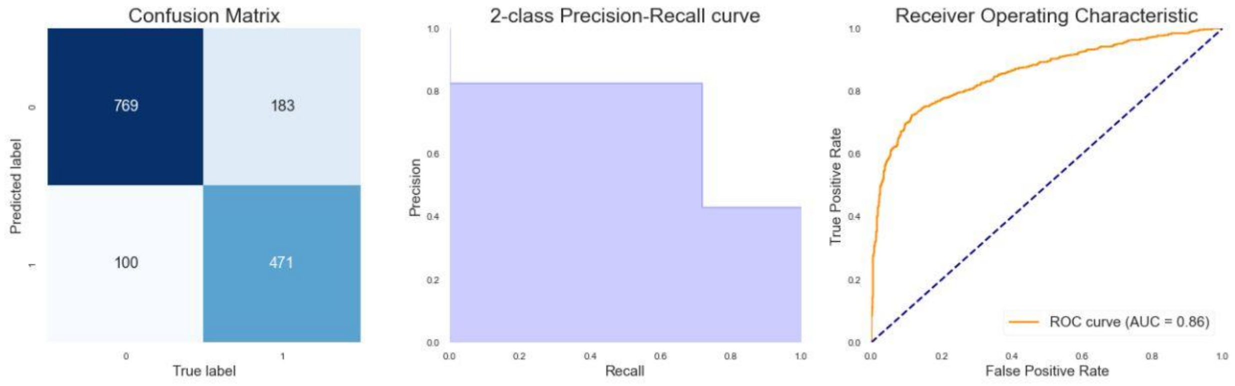
```
Accuracy : 0.7676 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.7381 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7110 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8321                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 8 SVC TF-IDF

```
show_summary_report(y_test, predictions_C2, predict_proba_C2)
```

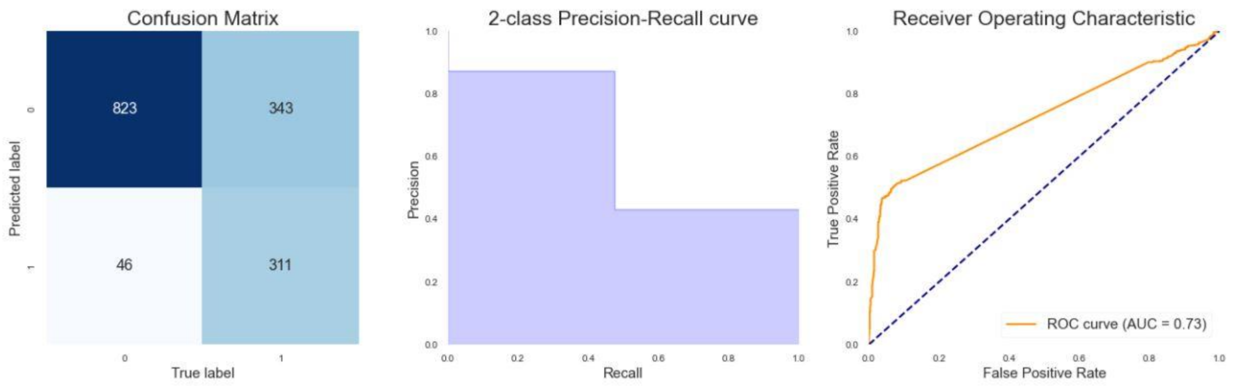
```
Accuracy : 0.8142 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8249 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7202 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8583                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 9 SVC TF-IDF N-Gram

```
show_summary_report(y_test, predictions_C3, predict_proba_C3)
```

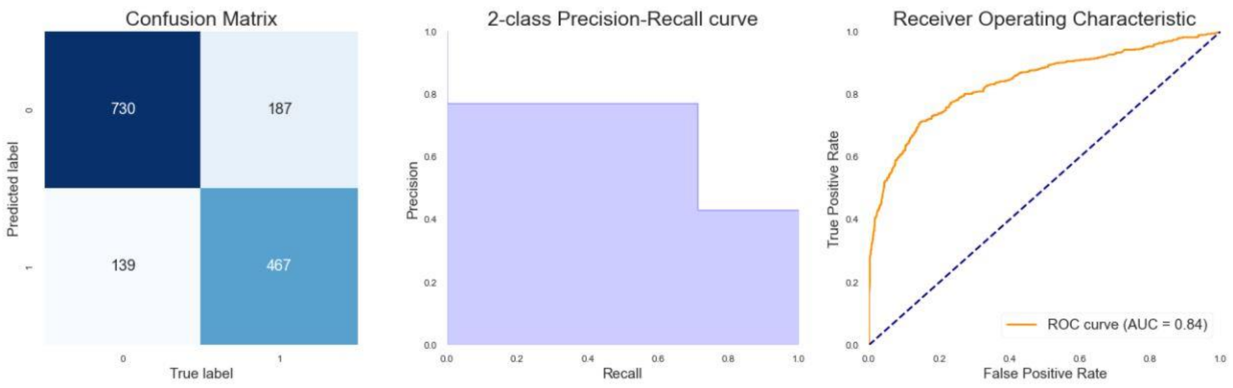
```
Accuracy : 0.7446 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8711 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.4755 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.7281                                                                      Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 10 Random Forest CountVectoriser

```
show_summary_report(y_test, predictions_D1, predict_proba_D1)
```

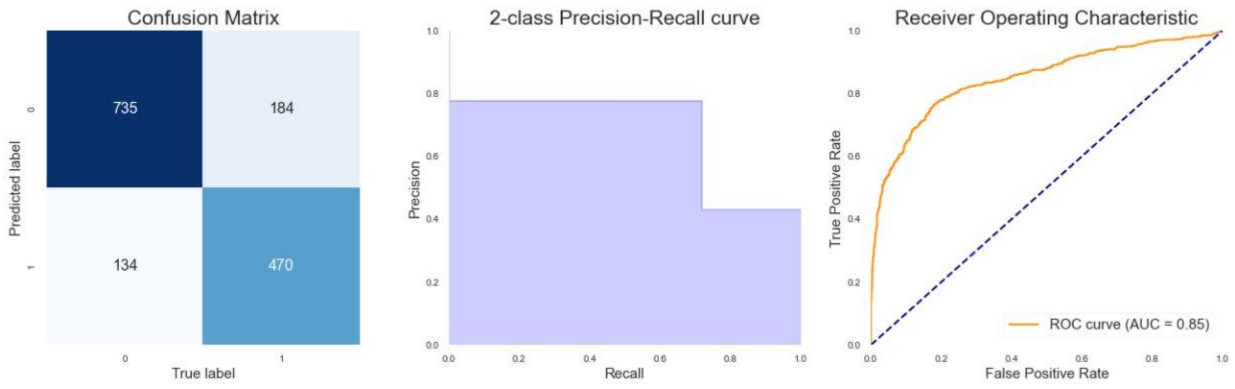
```
Accuracy : 0.7859 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.7706 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7141 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8383                                                                      Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 11 Random Forest TF-IDF

```
show_summary_report(y_test, predictions_D2, predict_proba_D2)
```

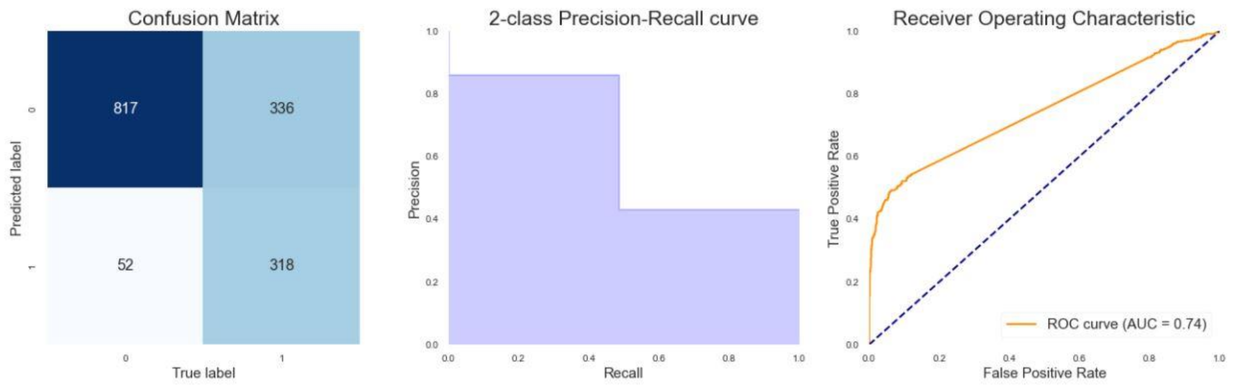
```
Accuracy : 0.7912 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.7781 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.7187 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8482                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 12 Random Forest TF-IDF N-Gram

```
show_summary_report(y_test, predictions_D3, predict_proba_D3)
```

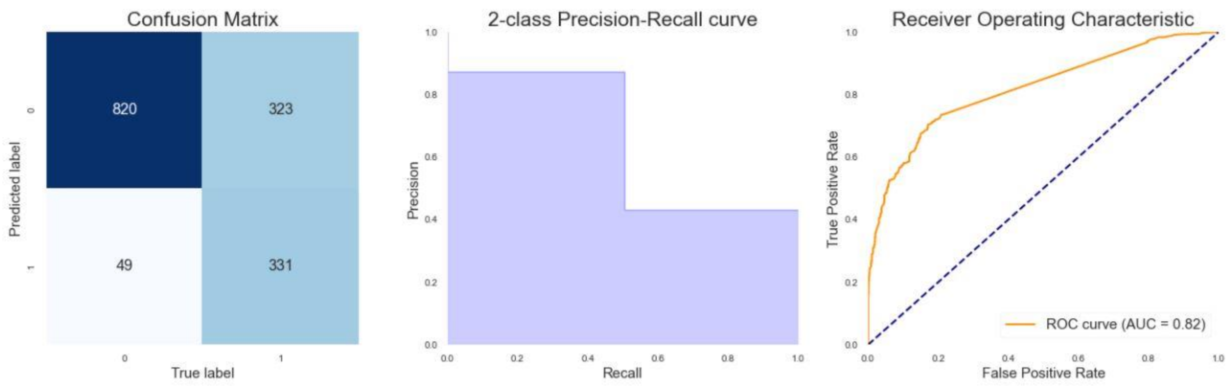
```
Accuracy : 0.7452 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8595 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.4862 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.7437                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 13 Gradient Descent CountVectoriser

```
show_summary_report(y_test, predictions_E1, predict_proba_E1)
```

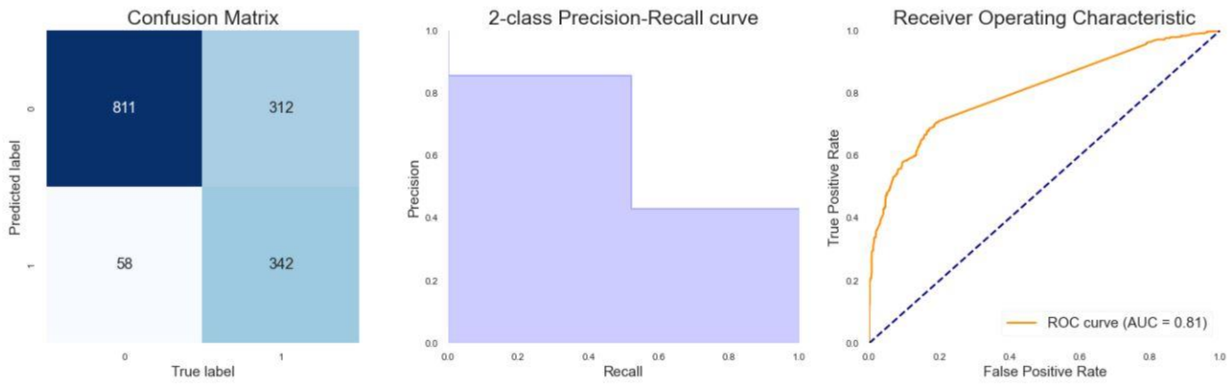
```
Accuracy : 0.7557 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8711 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.5061 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.8182                                                                    Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 14 Gradient Descent TF-IDF

```
show_summary_report(y_test, predictions_E2, predict_proba_E2)
```

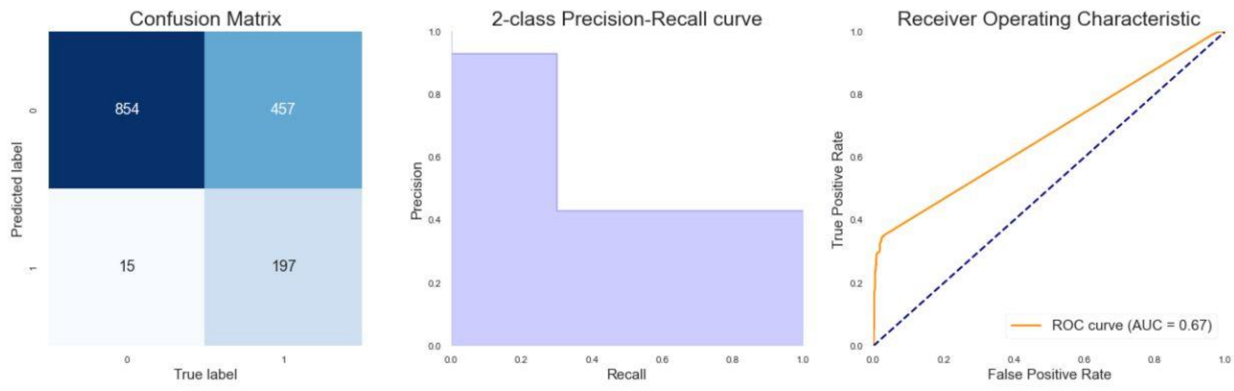
```
Accuracy : 0.7571 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8550 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.5229 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.8070                                                                    Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 15 Gradient Descent TF-IDF N-Gram

```
show_summary_report(y_test, predictions_E3, predict_proba_E3)
```

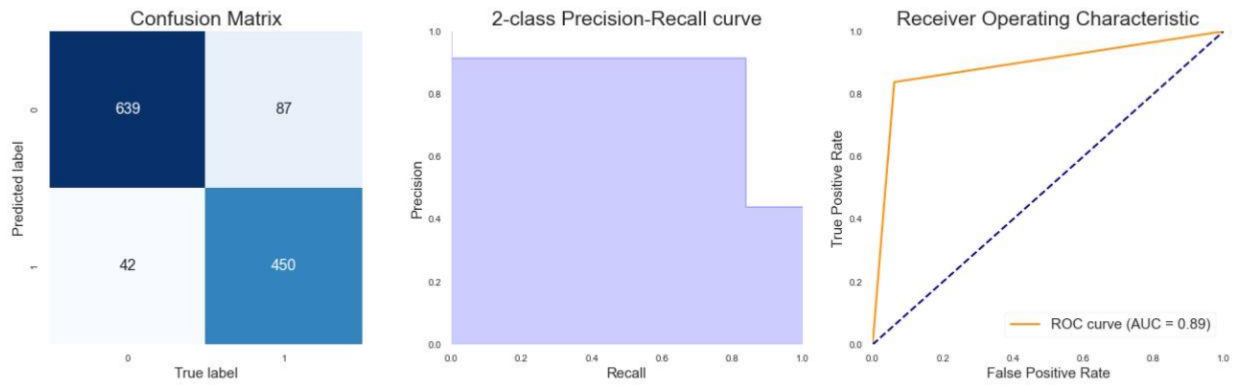
```
Accuracy : 0.6901 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.9292 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.3012 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.6711                                                                    Best: 1, Worst: < 0.5
------------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 16 Baseline Model Keras

```
show_summary_report(y_valid[:,1], y_round_pred_base)
```

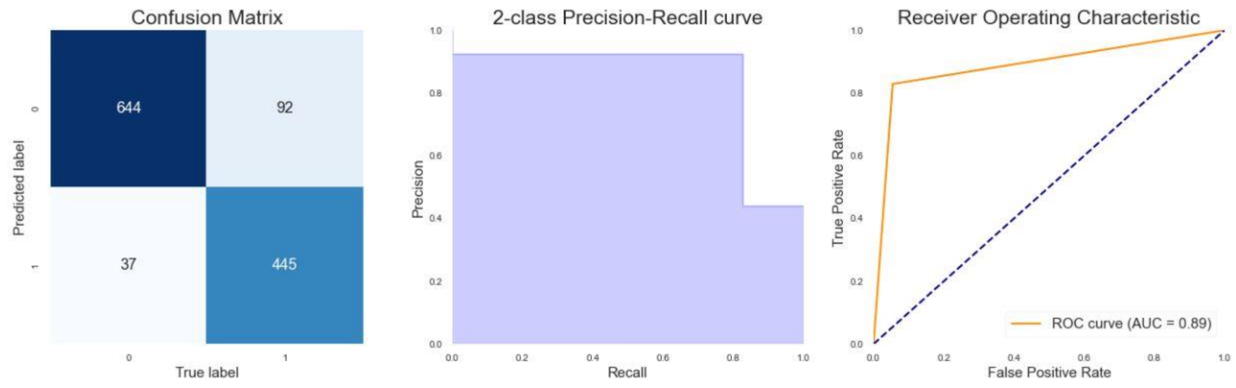
```
Accuracy : 0.8941 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.9146 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.8380 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.8882                                                                    Best: 1, Worst: < 0.5
------------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 17 Reduced Model Keras

```
show_summary_report(y_valid[:,1], y_round_pred_reduced)
```

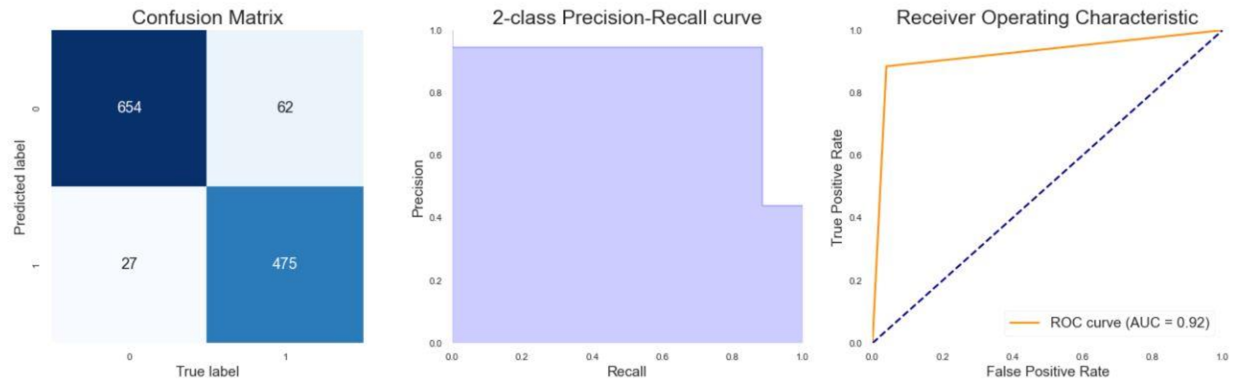
```
Accuracy : 0.8941 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.9232 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.8287 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8872                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 18 Regularised Model Keras

```
show_summary_report(y_valid[:,1], y_round_pred_reg)
```

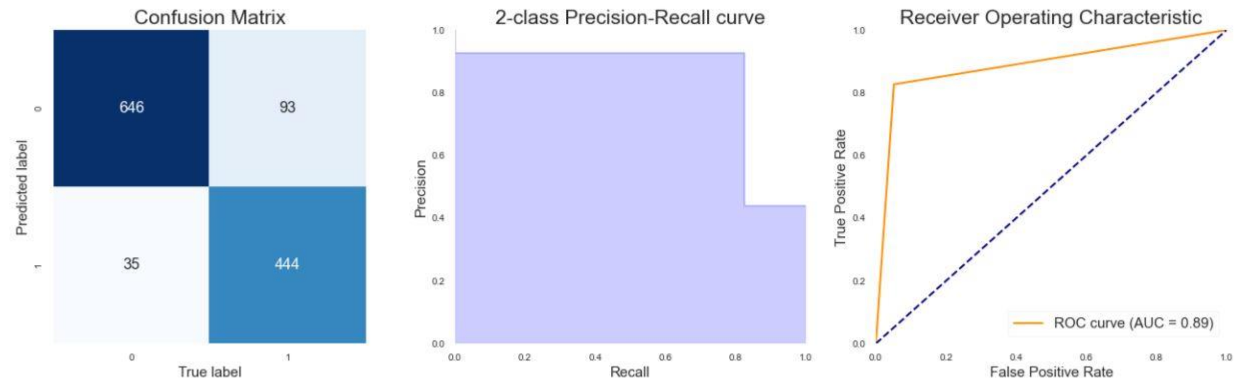
```
Accuracy : 0.9269 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.9462 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.8845 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.9224                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Appendix 19 Dropout Model Keras

```
show_summary_report(y_valid[:,1], y_round_pred_drop)
```

```
Accuracy : 0.8949 [(TP + TN) / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.9269 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.8268 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.8877                                                                      Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



References

[1] M. Roser, “The internet’s history has just begun,” Our World in Data, 3 October 2018. [Online]. Available: https://ourworldindata.org/internet-history-just-begun. [Accessed 15 September 2020].

[2] E. Ortiz-Ospina, “The Rise of Social Media,” 18 September 2019. [Online]. Available: https://ourworldindata.org/rise-of-social-media. [Accessed 16 September 2020].

[3] A. Kalogeropoulos, “How Younger Generations Consume News Differently,” Reuters Institute, 24 May 2019. [Online]. Available: http://www.digitalnewsreport.org/survey/2019/how-younger-generations-consume-news-differently/. [Accessed 16 September 2020].

[4] N. Huiwen, “4 in 5 Singaporeans confident in spotting fake news but 90 per cent wrong when put to the test: Survey,” Straits Times, 27 September 2018. [Online]. Available: https://www.straitstimes.com/singapore/4-in-5-singaporeans-confident-in-spotting-fake-news-but-90-per-cent-wrong-when-put-to-the. [Accessed 17 September 2020].

[5] F. Shahari, “10+ Interesting Social Media Statistics (2020),” CloudRock, 15 May 2020. [Online]. Available: https://cloudrock.asia/sg/blog/social-media-statistics-singapore/#:~:text=By%20January%202020%2C%20more%20than,in%20Singapore%20had%20touched%2079%25.. [Accessed 16 September 2020].

[6] 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].

[7] 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.

[8] 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].

[9] P. Grabiński, “Feature engineering, Explained,” KD Nuggets, December 2018. [Online]. Available: https://www.kdnuggets.com/2018/12/feature-engineering-explained.html. [Accessed 17 September 2020].

[10] “Text Classification,” MonkeyLearn, 2020. [Online]. Available: https://monkeylearn.com/text-classification/#:~:text=Text%20classification%20is%20the%20process,spam%20detection%2C%20and%20intent%20detection.. [Accessed 18 September 2020].

[11] “Real or Not? NLP with Disaster Tweets,” Kaggle, [Online]. Available: https://www.kaggle.com/c/nlp-getting-started/overview. [Accessed 18 September 2020].