Machine Learning Engineer Nanodegree

Capstone Project

Nabanita Roy
September 2021

I. Definition

Project Overview

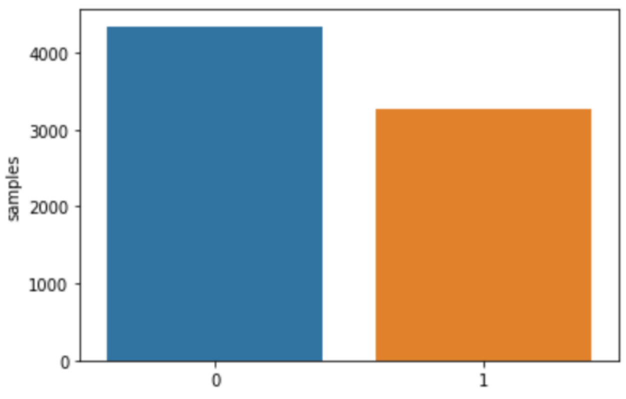
Natural Language Processing with Disaster Tweets is a Kaggle Challenge where tweets are collected with labels indicating whether the tweets are about a disaster that occurred or not. Since tweets are social media language, therefore, it is a challenge to automatically identify them. Besides, ambiguity in texts makes it more difficult to achieve automatic identification of tweets containing information on real disaster. The objective of this project is to predict using machine learning if a tweet contains information on occurrence of a real disaster or not.

Problem Statement

Given a tweet, this task is designed to identify if it contains information on occurrence of a real disaster or not.

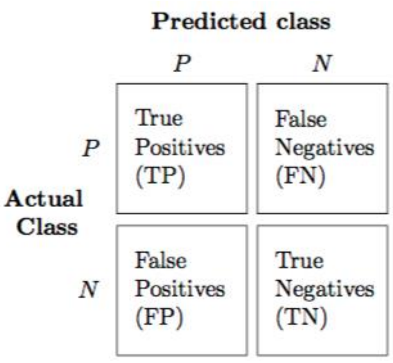
Metrics

This challenge is a classification problem therefore accuracy, precision, recall are relevant measures to assess the predictions. Since the training data is slightly imbalanced, the accuracy will not be considered as a useful metric in this case and only F1-score will be considered to avoid accuracy paradox. F1-score is helpful to monitor both precision and recall as metrics since it represents their harmonic mean. An F1-score of more than 80% on the test set of this dataset is desirable.



Performance Metrics Definitions

In a typical classification problem solved using Machine learning techniques, the outputs can be represented in a 2x2 matrix form which is called the 'Confusion matrix' which is shown in the diagram on the right. The different classification metrics based on the confusion matrix can be defined as:



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

- Accuracy = TP + TN / (TP + FP + TN + FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-Score = 2 * Precision x Recall /(Precision + Recall) = 2TP/(2TP+FP+FN)

II. Analysis

Data Exploration

This dataset has 7613 data points where each data point contains the tweet text, an unique id for the tweet, a keyword from that tweet, the location from where the tweet was posted and also if the tweet is about real disaster or not. Here is a column name and summary of the information available:

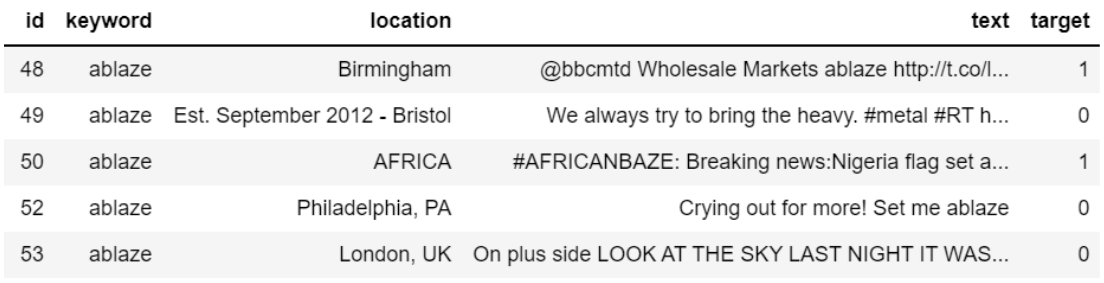
Column Name	Description
id	A unique identifier for each tweet
keyword	A keyword from that tweet
location	The location from where the tweet was posted
text	Tweet text
target	Indicates whether a tweet is about real disaster or not. Present only in training set.

This dataset contains missing values in columns 'location' and 'keyword'. Following is a summary of missing values:

Column Name	Missing Values (Percentage)
id	0.0
keyword	0.8
location	33.3
text	0.0
Target	0.0

The 'location' column is the one with the most missing values. Since Named Entity Recognition (NER) systems can detect locations, these missing values could be filled.

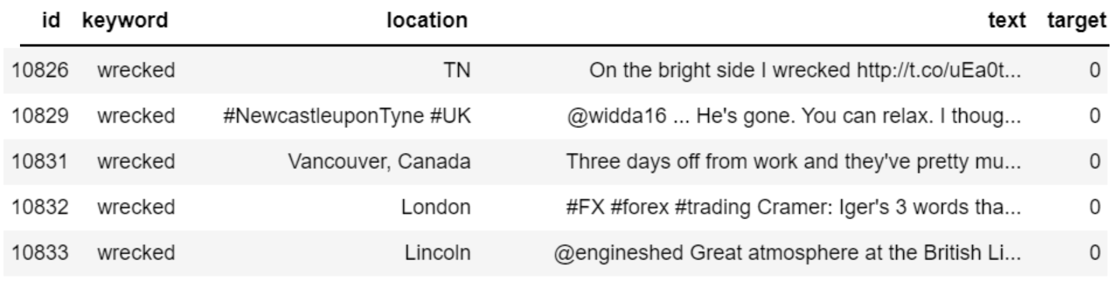
Following is a sample of the dataset:



id	keyword	location	text	target
48	ablaze	Birmingham	@bbcmtd Wholesale Markets ablaze http://t.co/l...	1
49	ablaze	Est. September 2012 - Bristol	We always try to bring the heavy. #metal #RT h...	0
50	ablaze	AFRICA	#AFRICANBAZE: Breaking news:Nigeria flag set a...	1
52	ablaze	Philadelphia, PA	Crying out for more! Set me ablaze	0
53	ablaze	London, UK	On plus side LOOK AT THE SKY LAST NIGHT IT WAS...	0

In the column ‘text’, typical social media texts are observed which includes tweet mentions, hashtags, urls and inconsistent casing. Besides, in the ‘location’ column, the location information is also in consistent. It could contain a city name, a country name, name of a state or province, or a combination of these. It could also contain other non-spatial information like ‘Est. September 2021’ as in the second row of the data table. The target column contains 0 or 1, where 1 indicates that the tweet is about a real disaster and 0 indicates that the tweet is not about a real disaster.

Below is another example of the data instances. Clearly, the ‘location’ column contains hashtags as well and state abbreviations. The ‘text’ column contains contractions and incorrect punctuations.

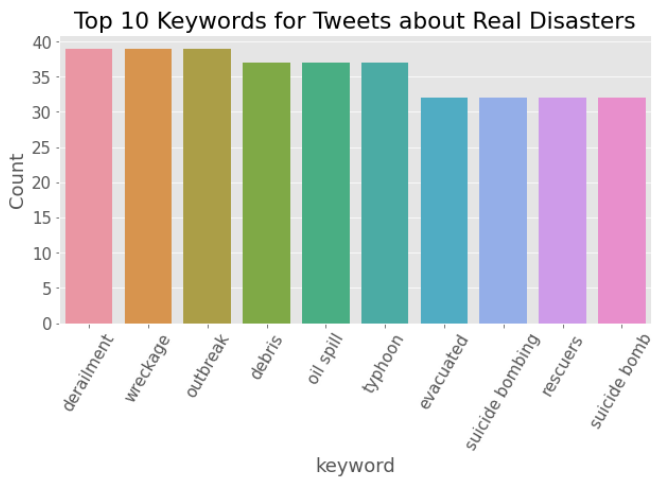


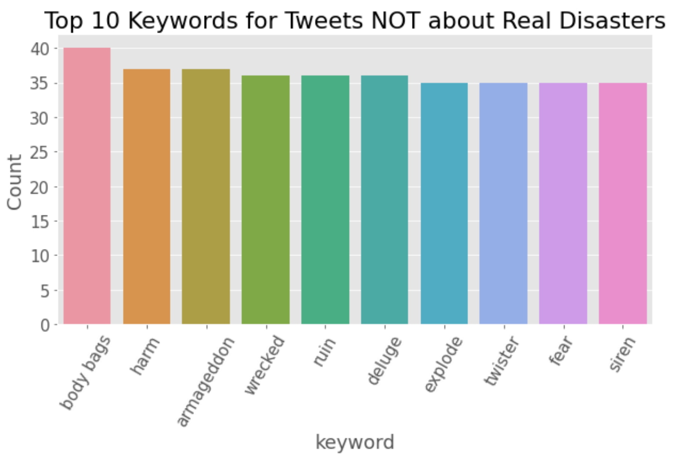
id	keyword	location	text	target
10826	wrecked	TN	On the bright side I wrecked http://t.co/uEa0t...	0
10829	wrecked	#NewcastleuponTyne #UK	@widda16 ... He's gone. You can relax. I thoug...	0
10831	wrecked	Vancouver, Canada	Three days off from work and they've pretty mu...	0
10832	wrecked	London	#FX #forex #trading Cramer: Iger's 3 words tha...	0
10833	wrecked	Lincoln	@engineshed Great atmosphere at the British Li...	0

Finally, 57% of the dataset contains information on tweets about real disasters and 42% contains information on tweets not about real disasters. Therefore, the dataset is almost balanced and no further balancing the target datapoints is required.

Exploratory Visualization

In this dataset, 'keywords' is an important information about the tweets. Following is a bar chart about the top 10 keywords in the tweets about real disasters and those not about real disasters.





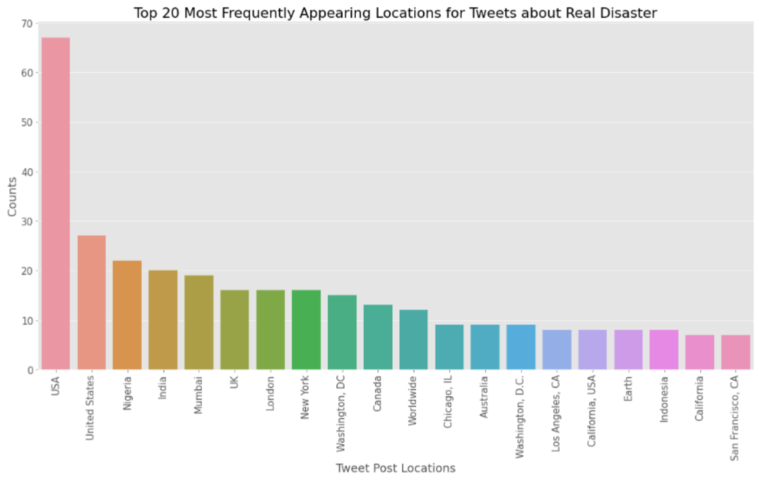
It is observed that derailment, wreckage and outbreak are the top three keywords in the tweets which are about real disasters. In the tweets not about real disasters, body bags, harm, armageddon are the top three keywords. The keywords – 'wreckage' and 'wrecked', both originating from the work 'wreck', occur in both target categories. Hence, one of the steps in pre-processing would be using stemming or lemmatizing in this use-case.

Following are the probabilities of the keywords occurring in real disaster tweets, where 'derailment', 'debris' and 'wreckage' have a 100% probability. This means that in the training set, every tweet having these three keywords were about real disasters. Similarly, the tweets having the keyword 'outbreak', 97% of them were about real disasters. Therefore, it could be assumed that if these keywords, with over 90% probability occur in a tweet, the probability of them being about a real disaster is much higher than any other keyword. However, this conclusion is based on non-normalised keywords data because of which keywords like 'suicide bombing' and 'suicide bomber' are occurring separately in this list.

Keyword	Probability the Keyword is in a Tweet about Real Disaster
derailment	1.000000
debris	1.000000
wreckage	1.000000
outbreak	0.974359
typhoon	0.972973

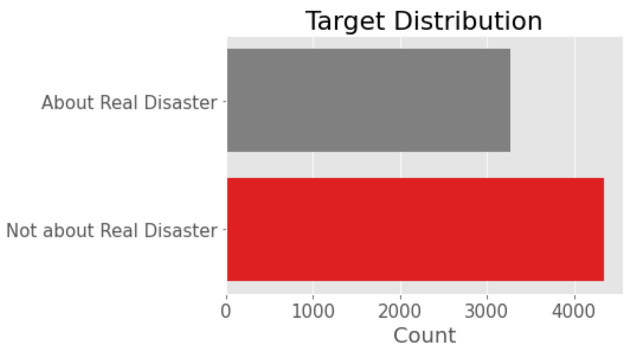
oil spill	0.972973
suicide bombing	0.968750
suicide bomber	0.966667
bombing	0.925926
rescuers	0.906250

The next column is ‘location’, for which a bar chart has been used to plot the top 20 most frequently occurring locations.



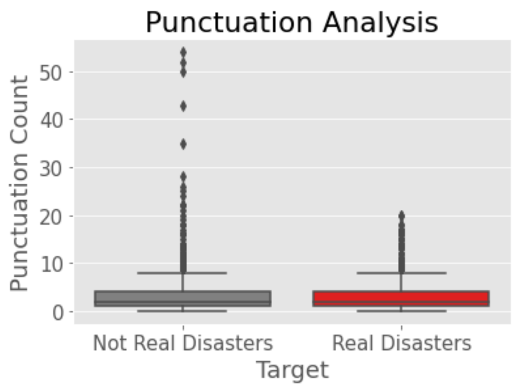
As in this bar chart, USA appeared most in tweet locations. Again, the inconsistency in location texts is evident because ‘USA’, states within USA and ‘United States’ are all appearing in the top 20 list. Therefore, several locations from the USA has appeared most number of times here in general, followed by Asian countries, UK, Nigeria, Canada and Australia, if aggregated.

Finally, the ‘target’ column distribution is of interest to check if the data is balanced. Following is the bar plot of the counts of data instances for tweets about and not about real disasters.



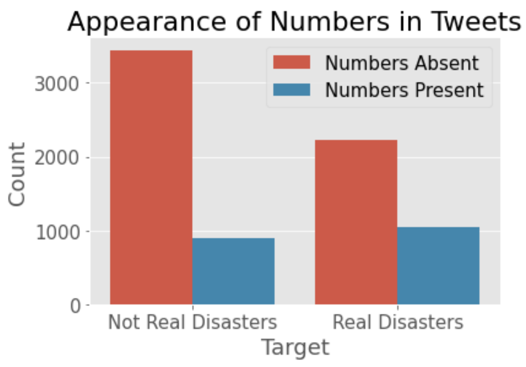
Apart from these techniques, other textual features were explored in the tweets, such as:

1. Number of punctuations: Punctuations could be overused in tweets which could be a supportive indicator of discerning the target classes. Therefore, a punctuation count analysis was conducted, as shown below:



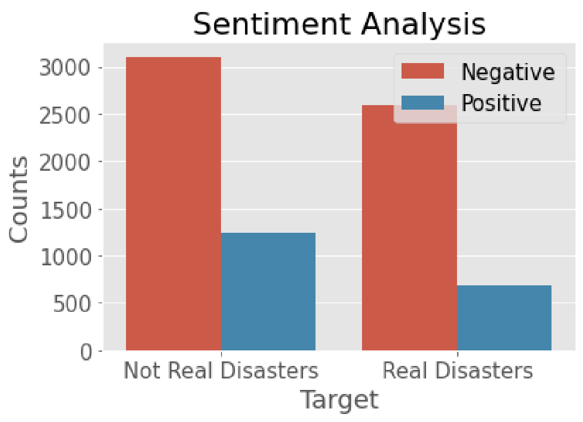
It is evident in this box plot that the IQR of the boxplots are comparable but for tweets not about real disasters have way more outliers of higher orders than those about real disasters.

2. Appearance of numbers in Tweets: Numbers of casualties are often mentioned in texts about disasters.



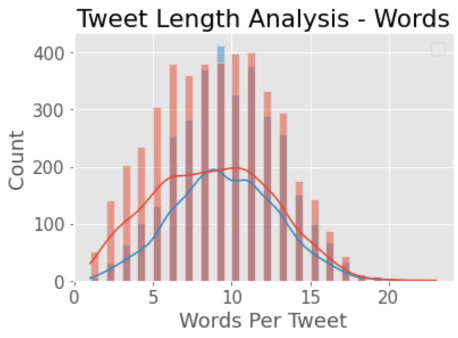
In the bar chart on the left, it is evident that the probability of finding numbers in real disaster tweets is higher.

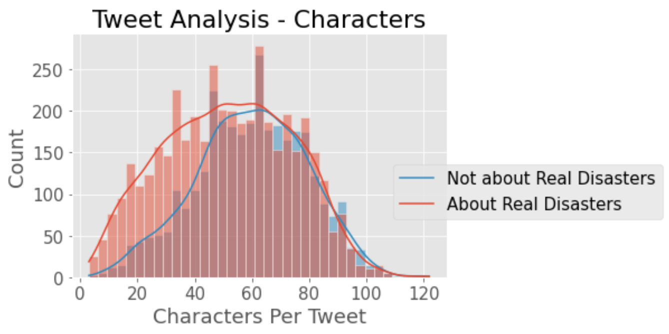
3. Sentiment Analysis of Tweets: Tweets about real disasters are more likely to have less words about positive emotions than negative ones.



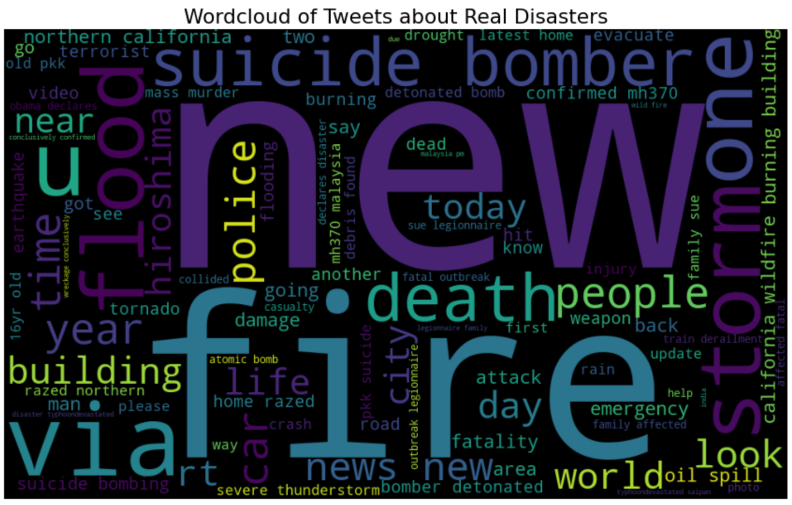
The bar chart on the left clearly shows that even though negative emotions are dominant in the entire corpus, the percentage of positive emotions in tweets about real disasters is about 50% lower than the tweets not about real disasters.

4. Tweet Length Analysis: The following analysis is conducted to verify if there is any distinguishable element in the lengths of the texts. Hence, I have compared the lengths of the texts, in terms of words and characters per tweet, between the two types of tweets. The distribution of the tweets about real disasters are more like a symmetric distribution while the one not about real disaster are somewhat left skewed.

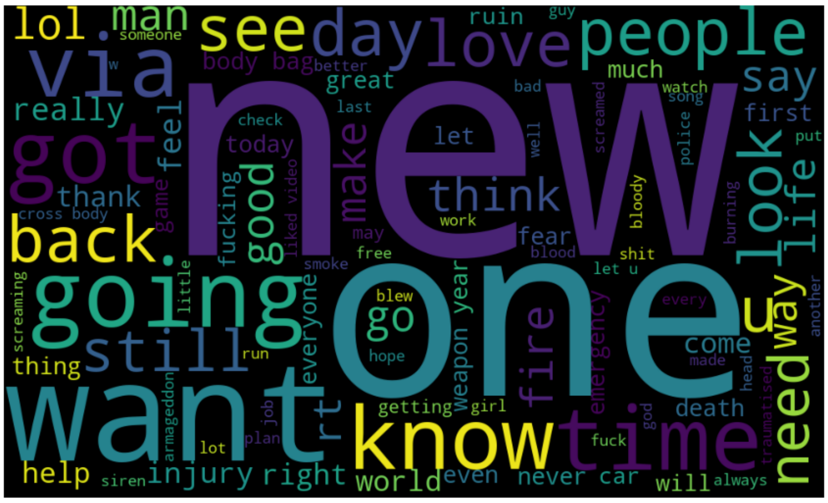




5. Term Frequency Analysis using Wordcloud: Wordcloud is the most used visualization technique for natural language texts to search for frequently occurring words. On the left is a wordcloud of most frequently occurring words in tweets about real disasters. Words about disasters are primarily present here. From an unbiased view, the words like fire, death, wildfire, storm, burning, flood, damage, casualty, police, bomb, evacuate, drought, earthquake – the collection of these words could be attributed to the topic of natural disasters. On the contrary, in the below wordcloud about tweets not about real disasters, the words cannot be easily assigned a particular topic. They are more random. Hence, it could be safely assumed that machine learning techniques based on term frequencies would be a great fit to use on this corpus.



Wordcloud of Tweets NOT about Real Disasters



Algorithms and Techniques

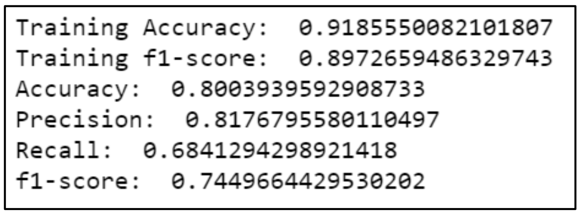
To predict if a tweet is about real disasters or not, using Bag of Words (BoW) approach of using term-frequencies, followed by a classifier like Logistic Regression seems promising. Natural Language Processing techniques to extract features from texts was also used in addition to the BoW model. For predictions, while Logistic Regression was used for benchmarking, the following algorithms were also used to test the performances:

- Support Vector Machines
- K- Nearest Neighbours
- Decision Tree
- Random Forest
- Gradient Boosting Algorithm

Benchmark

The benchmark for this project is designed as:

1. Term Frequency – Inverse Document Frequency approach, with uni-grams and bi-grams only
2. Logistic Regression with default parameters



```
Training Accuracy:  0.9185550082101807
Training f1-score:  0.8972659486329743
Accuracy:  0.8003939592908733
Precision:  0.8176795580110497
Recall:  0.6841294298921418
f1-score:  0.7449664429530202
```

The result of running the benchmarking pipeline on the dataset is in the image above. The training and test scores look good with the basic configurations.

III. Methodology

Data Pre-processing

This dataset contains texts from social media and referring to the nature of the data explained in Section II, the following procedure for cleaning and pre-processing was created –

1. Remove URLs from text
2. Remove HTML entities
3. Convert to lower case
4. Detect the Word News – In general, in tweets people tend to use the word news either to inform that something has happened or to tag news channels. Hence, this function is designed to detect is the word news is present or not. If present, then add the word to end of the tweet text since TF-IDF approach is used and therefore, the order of words is insignificant.
5. Remove social media tags – In this function, the social media tags which occurs with an '@' in-front is removed. Hashtags are preserved but the sign '#' is removed. This is because #earthquake is important information which needed to be preserved.
6. Basic Pre-processing – This includes
    a. keeping only alphabets, digits and spaces,
    b. tokenizing
    c. expanding contractions
    d. lemmatization
    e. removing stopwords
    f. stripping white space

The tweets go through these steps and the final version of the tweets is of data type string.

Besides pre-processing the tweets, other features were also extracted. Below is a list of those features along with a description of how the features were extracted.

1. Punctuation Count – Using Regular Expressions, the number of punctuations in a text was calculated.
2. Appearance of Numbers in Tweets– Using Regular Expressions, the appearance of numbers in a text was identified. If a number was present, then the output was 1, else 0.
3. Sentiment Score – A sentiment score was obtained using TextBlob python library. If the score is more than 0, then the sentiment score used in this use-case is 1, else 0, where 1 indicates positive emotion and 0 indicates negative emotion.
4. Words Per Tweet – An estimate of the number of words in a tweet.
5. Characters Per Tweet - An estimate of the number of characters in a tweet.

6. Present in BoW – A Bag of Words of words about disasters, that were identified from the wordcloud of tweets about real disasters was curated. If any word from that bag was found in a tweet, then the function return 1, else 0.
7. Identifiable Location – Lastly, if a proper location in a tweet was found , then this function returned 1, else 0.

Implementation

To predict if a tweet is about real disasters or not, using Bag of Words approach of using term-frequencies, followed by a classifier like Logistic Regression seems promising.

Bag of Words (BoW) Model

A bag-of-words is a representation of text that describes the occurrence of a set of words, called the vocabulary, within a document and is accompanied by a measure of the presence of those words. The model only considers whether those words occur in a document or not. It ignores the sequence of the words in which they occur in the document and hence the word 'bag' is used to describe the model.

The BoW model assumes that documents which are similar, contain similar terms. In this use-case identifying tweets about real disasters indicates identifying tweets containing words about disasters. The term-frequencies of the vocabulary of the BoW model will be considered as well to identify the words that have been occurring frequently in the tweets about real disasters. Using only **term-frequencies** emphasizes on the most frequently appearing words which may or may not contain 'information' about the document and undermines the terms which do not occur much frequently, like domain-specific words, which are important but might not be the most frequently occurring word in a document. Hence, **inverse document frequency** is used to normalize the word counts over a corpus containing multiple documents.

Logistic Regression

One of the simplest classifiers which models the probability of the target classes based on the given data points. Logistic regression is a linear method, but the predictions are transformed using the logistic function. The logistic or sigmoid function is expressed as $1 / (1 + e^{-x})$. It transforms numbers in the range of 0 and 1.

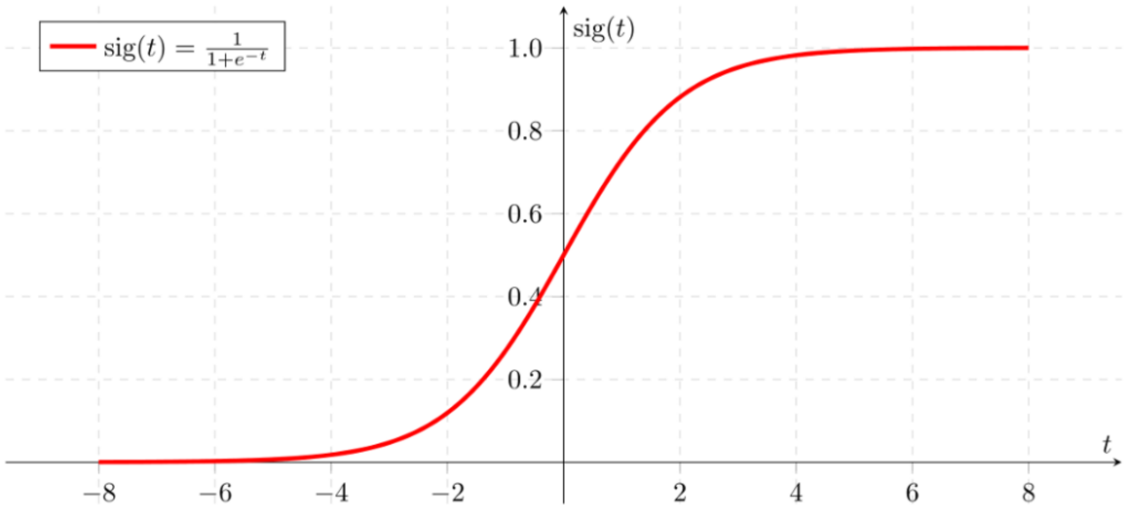


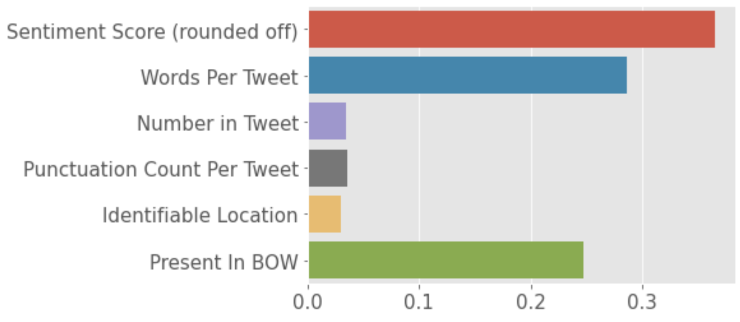
Image Source: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

To predict which class a data point belongs to, a threshold can be set, called the decision boundary. Based upon this threshold, the obtained estimated probability is classified into classes. For example, if modelling a data to detect if a transaction was fraud or genuine, if the threshold if 0.5, then if the probability obtained is greater than 0.5, then the transaction is fraud.

Feature Selection

Apart from obtaining a sparse matrix of vocabulary x term-frequencies, the other extracted features were also used after testing the benchmark with Logistic Regression. The summary of the selections and procedure is as documented below:

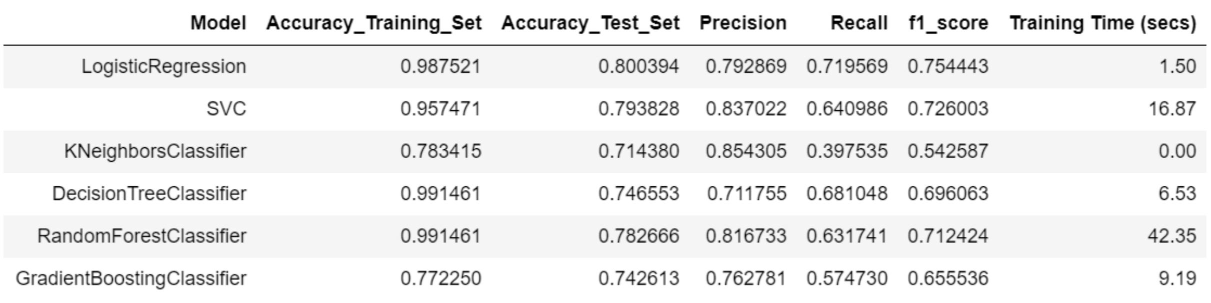
- For 'Words Per Tweet' feature, Standard Scaling was applied after experimenting with a series of scalers available using SkLearn.
- Characters per tweet was discarded since words per tweet also contain similar information with strong correlation.
- Used Random Forest to train on all the features and obtained the feature importances to estimate which features to retain. From the graph on the right, 'Sentiment Score', 'Words Per Tweet' and 'Present in BoW' were selected due to their precedence over other features.



Experimenting with Classifiers

Once the features were selected, the series of classifiers, were tested on the dataset. These classifiers are:

- Logistic Regression
- Support Vector Machines
- K- Nearest Neighbours
- Decision Tree
- Random Forest
- Gradient Boosting Algorithm



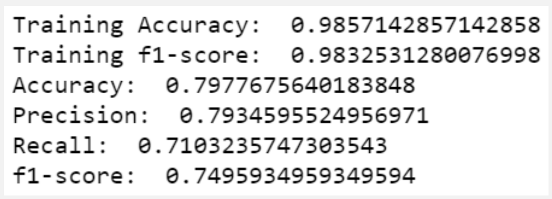
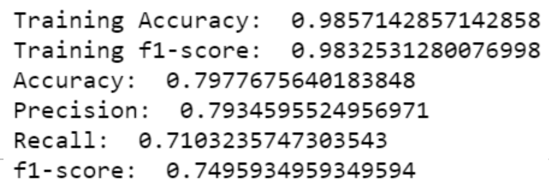
Model	Accuracy_Training_Set	Accuracy_Test_Set	Precision	Recall	f1_score	Training Time (secs)
LogisticRegression	0.987521	0.800394	0.792869	0.719569	0.754443	1.50
SVC	0.957471	0.793828	0.837022	0.640986	0.726003	16.87
KNeighborsClassifier	0.783415	0.714380	0.854305	0.397535	0.542587	0.00
DecisionTreeClassifier	0.991461	0.746553	0.711755	0.681048	0.696063	6.53
RandomForestClassifier	0.991461	0.782666	0.816733	0.631741	0.712424	42.35
GradientBoostingClassifier	0.772250	0.742613	0.762781	0.574730	0.655536	9.19

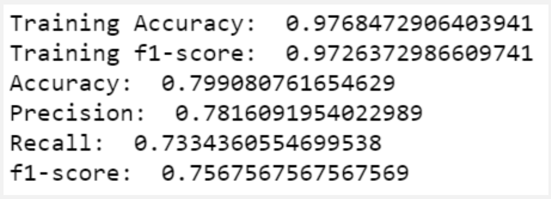
The best performance on an average is that of the Logistic Regression algorithm. Hence, the refinement task was carried on with it. Following Logistic Regression, the Support Vector Machine (SVC in the table) and Random Forest performed well too.

Refinement

The refinement of the Logistic Regression algorithm was performed in three attempts using GridSearch with cross-validation of 5 folds. Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters of an estimator.

A summary of the grids and the results are as follows:

Grid	**Best Params**	**Results**
max_iter = [100, 200, 500, 1000] **C = [0.1, 0.5, 1, 10, 50, 100]**	C: 1 max_iter: 100	 Training Accuracy: 0.9857142857142858 Training f1-score: 0.9832531280076998 Accuracy: 0.7977675640183848 Precision: 0.7934595524956971 Recall: 0.7103235747303543 f1-score: 0.7495934959349594
max_iter = [50, 100, 150] **C = [1, 5, 10]**	C: 1 max_iter: 100	 Training Accuracy: 0.9857142857142858 Training f1-score: 0.9832531280076998 Accuracy: 0.7977675640183848 Precision: 0.7934595524956971 Recall: 0.7103235747303543 f1-score: 0.7495934959349594

penalty = ['elasticnet', 'l1', 'l2']
 	
solver = ['saga', 'liblinear', 'lbfgs']
 	
max_iter = [100]
 	
C = [1]
 	
fit_intercept = [True, False]
	C: 11
	max_iter: 100
	fit_intercept:True
	penalty: l2
	solver: saga
		
		`Training Accuracy:  0.9768472906403941`
		`Training f1-score:  0.9726372986609741`
		`Accuracy:  0.799080761654629`
		`Precision:  0.781609195402298 9`
		`Recall:  0.7334360554699538`
		`f1-score:  0.7567567567567569`

IV. Results

Model Evaluation and Validation

The benchmark Logistic Regression model had a training accuracy and f1-score of 92%, and 90% while the test accuracy and f1-score are 80% and 75%.

After tuning the model, the training accuracy and f1-scores were 99% and 98% while the testing accuracy and f1-scores are 80% and 75% in attempt 1.

In attempt 3 of refinement, the training accuracy and f1-scores were 97% while the testing accuracy and f1-scores are 80% and 76%. Since the results have improved from the benchmark model. Hence, the final model selected was the result of the grid of attempt 3. The final configuration to the Logistic Regression model is:

- C: 1
- max_iter: 100
- fit_intercept: True
- penalty: l2
- solver: saga

Justification

The final results are not significantly better than the benchmark. There is plenty scopes of improvement, especially using transformers for text-classification. During the pre-processing process, smileys and emoticons were ignored, detecting which could boost the results.

The training accuracies has been already good from the benchmark. For the tuned models, the training scores improved even more but the test scores did not improve as much. This could be because the model was overfitting on the training dataset.

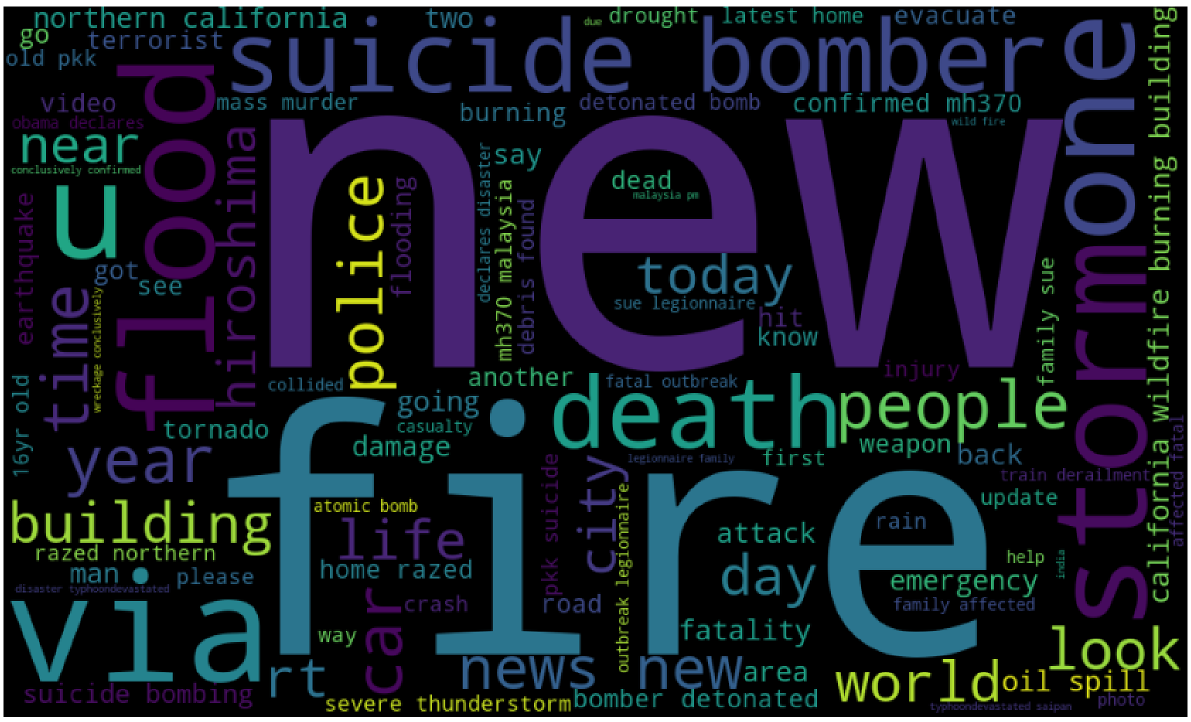
Tuning a support vector machine or a naïve bayes model could improve the performance. Going for decision trees or random trees might not be the best approach since the sparsity of the training data will impact the performance and also the training time.

V. Conclusion

Free-Form Visualization

The wordcloud of tweets about real disasters gave a strong indication of the pipeline comprising of the bag of words model with tf-idf approach followed by a probabilistic classifier would be the best combination for this project.

Wordcloud of Tweets about Real Disasters



Reflection

The project is about classifying if social texts are about real disasters or not, therefore, this project is based on natural language processing. Firstly, the texts were cleaned and pre-processed during which URLs, social media tags and hashtags were removed, stopwords were removed and words were lemmatized. Then features like sentiment, tweet length and BoW analysis was done. Next, a baseline model was selected which used pipeline comprising of a tf-idf transformer and a Logistic Regression model with default hyperparameter configurations. Next, a series of classifiers were experimented with and out of those Logistic Regression had the best performance, which was then selected to tune further. The tuned estimator was trained on features selected as a result of feature importance calculated from a random model forest. The selected features and the tweet text were then used to tune the estimator where the final set of hyperparameters are:

- C: 1
- max_iter: 100
- fit_intercept: True
- penalty: l2
- solver: saga

Improvement

The model could be improved by:

1. Using Naïve Bayes or other probabilistic models
2. Using an improved dictionary of disaster related terms
3. Using document clustering techniques for text classification
4. Using transformers for text classification

I would assume that using transformers will result into near perfect classification. If using the traditional approaches, the probabilistic models might perform better since this is a project where words about real disasters appearing frequently matters more. Random forest or decision trees might not perform well since the sparsity of the training data on using tf-idf transformer will impact the performance and also the training time, both.

References

- https://www.kaggle.com/c/nlp-getting-started/overview
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression
- https://machinelearningmastery.com/gentle-introduction-bag-words-model/
- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- https://spacy.io/
- https://blog.jkmsmkj.fyi/2018/10/confusion-matrix.html