Final Project- Machine Learning

(Kanchan Tenginakai, Phaneesh Nagalla and Rama Mohan Reddy Bigivemula)

Natural Language Processing with Disaster Tweets

Use Case:

In the digital age, social media platforms like Twitter have emerged as crucial channels for real-time communication during emergencies. Disaster relief organizations, news agencies, and emergency responders heavily rely on Twitter to gather crucial information during crises. However, interpreting the true nature of tweets in emergency situations can be challenging, especially when individuals use language ambiguously or metaphorically.

The Challenge:

Working as a part of a prominent disaster relief organization, "Global Disaster Response,", our team has been assigned a daunting task of sifting through an overwhelming volume of tweets to identify those that genuinely report real disasters. Take, for instance, the ambiguity in tweets like the one using the word "ABLAZE" metaphorically. While humans can discern the intended meaning, it poses a challenge for machines to accurately classify such tweets.

Our organization recognizes the need for an advanced machine learning model to automate the process of distinguishing between tweets reporting actual disasters and those using similar language in a non-literal sense. By leveraging a dataset of 10,000 hand-classified tweets, this model aims to significantly enhance the organization's ability to filter and prioritize tweets during emergencies.

Value to the Company:

The machine learning model's successful implementation will yield several advantages:

Enhanced Response Time: Accurate classification of disaster-related tweets will enable our company to swiftly identify and prioritize genuine emergencies.

Resource Optimization: By automating the tweet classification process, the organization can allocate its resources more efficiently, focusing on verified disaster reports.

Improved Situational Awareness: Rapid and accurate identification of disaster-related tweets will provide a comprehensive overview of ongoing crises, aiding in better situational awareness and decision-making.

Problem Statement:

Building a ML model which will utilize natural language processing (NLP) techniques to analyze the textual content of tweets, extracting features and patterns that differentiate genuine disaster reports from non-disaster-related discussions. Through meticulous training and validation, the model aims to achieve high accuracy in classifying tweets, thereby enhancing the organization's crisis management capabilities.

Approach:

First step is to install and import necessary libraries in python such as pandas, seaborn, matplotlib, stopwords, word tokenize, logistic regression and other classifier libraries. Then we loaded the training and test dataset which we will use to train, fit and test the model.

Information of the dataset:

We have the following columns in the data:

id - a unique identifier for each tweet

text - the text of the tweet

location - the location the tweet was sent from

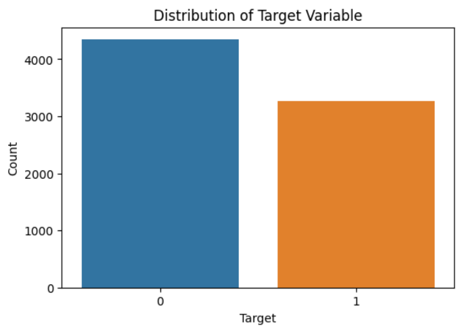
keyword - a particular keyword from the tweet

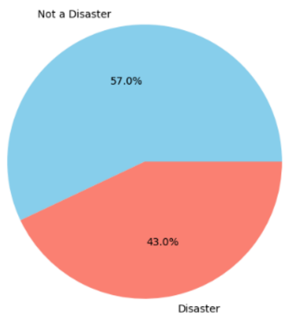
target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)

Exploratory Data Analysis:

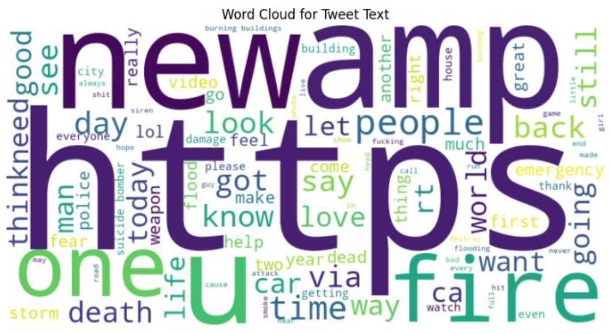
We have **7613** observations in the training dataset and **3263** observations in the test dataset.

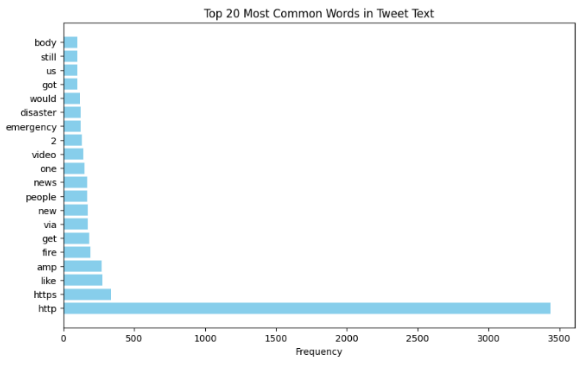
Out of 7613 observations we see that, 4342 are target 0 and 3271 are target 1.





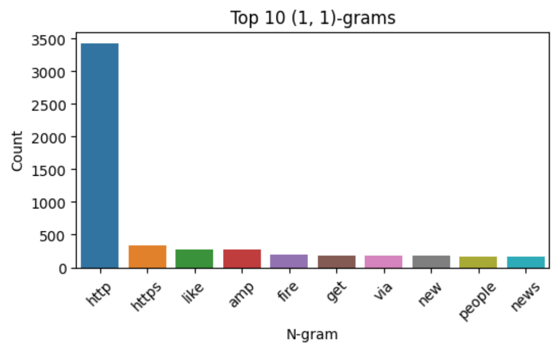
Using WordCloud library, we create a visual representation of the most frequent words in the collection of tweets. It helps visualize the distribution of words in the tweet text data, with larger words indicating higher frequency such as new, amp, https, fire, etc. Natural Language Toolkit (NLTK) also helps in finding the top 20 most frequent words along with their frequencies.

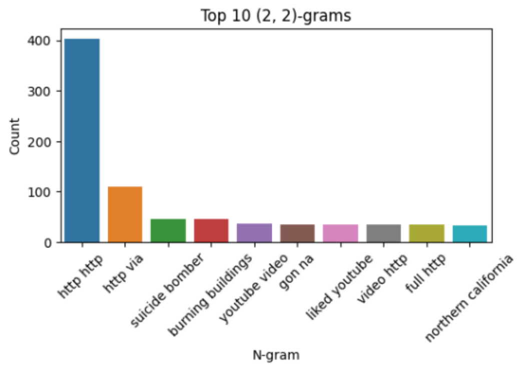




N-grams are contiguous sequences of n items (words in the context of text analysis). In the context of natural language processing (NLP) and text analysis, n-grams are used to understand the relationships between words in a sequence. The most common types of n-grams are unigrams (1-grams), bigrams (2-grams), trigrams (3-grams), and so on. Analyzing unigrams helps understand the most frequently occurring words in the dataset and bigrams provides insights into common word pairs or phrases.

The purpose of N-gram analysis is to identifying key phrases, recognizing common language patterns and expressions and generating features for machine learning models based on the occurrence of n-grams.





We used natural language processing (NLP) techniques, such as text preprocessing and tokenization, using the NLTK library. Specifically, the preprocess_text function performed the following NLP-related tasks:

Tokenization: splits the text into individual words using the word_tokenize function from the nltk.tokenize module

Stopword removal: removes common English stop words (e.g., "the," "is," "and") to focus on meaningful words

Utilizing the train_test_split function from scikit-learn, we split the dataset into training and validation sets which helps evaluate how well the trained model generalizes to unseen data during the validation phase.

Additionally, we modified the TF-IDF vectorizer to include n-gram features by setting ngram_range=(1, 2). This allows the model to capture not only individual words but also pairs of consecutive words.

<u>Training and testing the dataset:</u>

We will initialize logistic regression model from scikit-learn with random_state=42 that sets a seed for randomization to ensure reproducibility and train the model using the training data (X_train_tfidf) and their corresponding labels (y_train). We then apply the trained model to validation set to predict the labels based on the input features. We will process the test data using the same TF-IDF vectorizer applied to the training data and predict the target labels using a trained model.

<u>Evaluation of the model:</u>

Evaluation and comparison of model is based on F1 score and accuracy between the predicted and expected answers. F1 is calculated as follows:

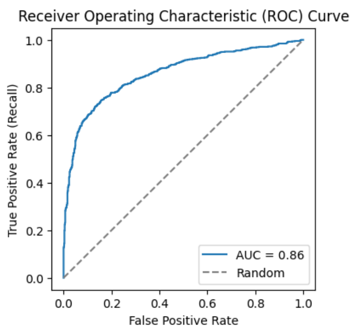
F1=2 $* \frac{precision*recall}{precision+recall}$; where, precision = $\frac{TP}{TP+FP}$ and recall = $\frac{TP}{TP+FN}$

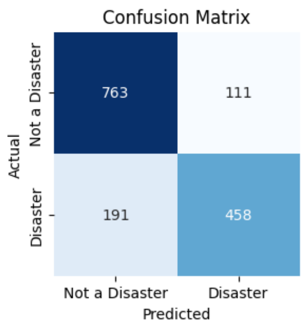
True Positive [TP] = your prediction is 1, and the actual value is also 1

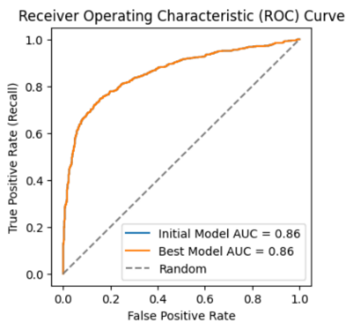
False Positive [FP] = your prediction is 1, and the actual value is 0

False Negative [FN] = your prediction is 0, and the actual value is 1

Logistic Regression model has a reasonable performance with an overall **accuracy of 80%**. The **F1-score** for Class 1 (Disaster Tweet) is **0.75**, indicating a good balance between precision and recall. However, there's room for improvement, especially in terms of recall for Class 1







An **AUC value of 0.86** indicates that the model has good discriminatory power in distinguishing between positive and negative instances. The higher the AUC, the better the model's ability to rank true positives higher than false positives across various probability thresholds. In this case, an AUC of 0.86 suggests a strong performance in terms of the receiver operating characteristic.

True Negatives (TN): 763, that is, number of instances correctly predicted as "Not a Disaster Tweet." False Negatives (FN): 191, that is, number of instances incorrectly predicted as "Not a Disaster Tweet" when they are actually "Disaster Tweets." True Positives (TP): 458, that is, number of instances correctly predicted as "Disaster Tweets." False Positives (FP): 111, that is, number of instances incorrectly predicted as "Disaster Tweets" when they are actually "Not a Disaster Tweet." The low counts of FP and FN indicate a balanced performance, suggesting that the model is making relatively few mistakes in both falsely identifying non-disaster tweets as disasters (FP) and falsely identifying disaster tweets as non-disasters (FN).

Precision (0.805): This indicates that out of all instances predicted as disaster tweets, approximately 80.5% were indeed about a real disaster. It measures the accuracy of positive predictions.

Recall (0.706): This indicates that the model correctly identified about 70.6% of all actual disaster tweets. It measures the ability of the model to capture all relevant instances of disasters.

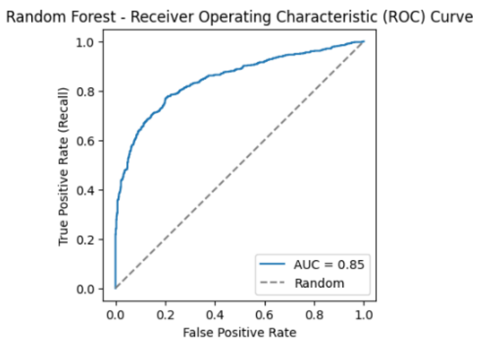
Tuning the model:

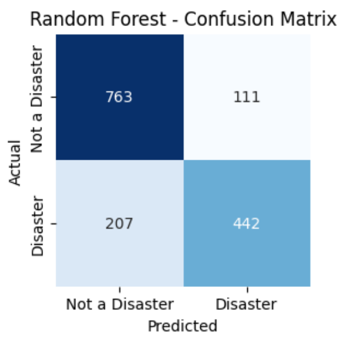
To further tune our logistic regression model, we used techniques like grid search from scikit-learn to find the best hyperparameters.

Here, we're searching over different values of the regularization parameter (C) such as 0.001, 0.01, 0.1, 1, 10 and 100. The grid search will evaluate each combination of these parameters using cross-validation and select the best combination i.e., **{'C': 1}.** We evaluated the model again using the optimal parameter value and get the **accuracy as 80%** and **F1 score as 0.75**

Building a Random Forest Model:

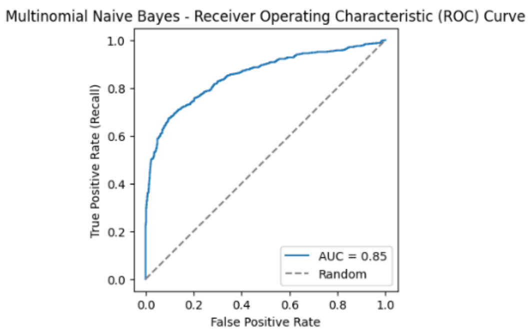
In order to check if there is a model that performs better than logistic regression model, we built a random forest classifier model. It involves hyperparameter tuning to find the best combination of hyperparameters using cross-validation (RandomizedSearchCV) and1 gives the following optimal parameters: {'n_estimators': 50, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'auto', 'max_depth': None}. After fitting and testing the model, we see that the **accuracy of the model is 79%** and **F1 score is 0.73** with an AUC value of 0.85 which is lesser compared to logistic regression.

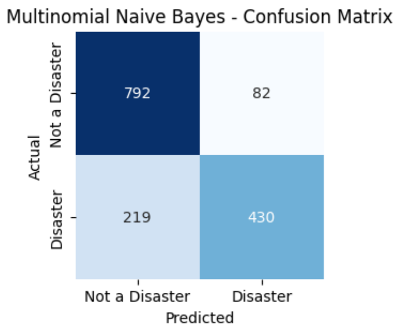




Building a Multinomial Naive Bayes Model:

Another popular classifier for NLP tasks is the Multinomial Naive Bayes classifier from sklearn.naive_bayes. It's particularly well-suited for text classification problems where the features are discrete counts such as word counts. We see that the **accuracy of the model is 80%** and **F1 score is 0.74** with an **AUC value of 0.85** which is again less compared to logistic regression.





Experimenting with a text preprocessing technique for the logistic model- Lemmatization:

Lemmatization reduces words to their base or root form. This can be useful for reducing the dimensionality of the feature space. By incorporating lemmatization into the text preprocessing pipeline, we aim to unify words with the same meaning, potentially improving the logistic regression model's performance by providing it with more standardized and normalized text data. After following a similar approach of training, fitting and testing the model, we get the **accuracy of the model as 80%** and **F1 score as 0.75**.

Conclusion:

In summary, the logistic regression model has a relatively high precision, meaning that when it predicts a tweet as a disaster, it is often correct. The final **accuracy is 80%,** the **F1-score** for Class 1 (Disaster Tweet) is **0.75** and an **AUC value of 0.86** which enhances the organization's crisis management capabilities. However, the recall is moderate, indicating that there is room for improvement in capturing a higher proportion of actual disaster tweets.