Classification of Disaster Tweets

Authored By: Sharon Colson, Thomas Robertson, Caleb Smith, Tania Flores

1. Problem Statement and Background

Twitter has become an essential platform for real-time communication, particularly in times of emergency. The widespread use of smartphones allows individuals to instantly share information about unfolding disasters. As a result, organizations such as disaster relief agencies and news outlets seek to systematically monitor Twitter to detect and respond to critical situations.

However, not all tweets mentioning disaster-related terms pertain to actual emergencies. Many tweets use such language metaphorically or in unrelated contexts. This creates a significant challenge in accurately distinguishing tweets that report real disasters from those that do not. The primary goal of this project is to develop a model for classifying disaster-related tweets that balances accuracy, efficiency, and adaptability. A tool such as this would ensure more reliable disaster detection, helping emergency teams to respond faster and more effectively.

Data Source & Characteristics:
The dataset used for this project is sourced from Kaggle and originates from Figure-Eight (formerly hosted on their 'Data for Everyone' website). It consists of tweets labeled as either related to a real disaster or not. The dataset is structured to train machine learning models capable of predicting the likelihood of a tweet being disaster-related. The key characteristics of the dataset include:

- Tweets collected from various sources, potentially covering different languages, locations, and contexts.
- Presence of user-reported metadata such as location and keywords.
- Binary classification labels indicating whether a tweet is reporting a disaster or not.

Success Measures:
We will use accuracy (ratio of correct predictions to total predictions) as our baseline evaluation for our classification models. However, going forward, we will use more sophisticated success measures that are more pertinent to the problem at hand.

Background and Related Work:
Disaster detection on social media is crucial for emergency response teams, disaster relief organizations, and news agencies. Twitter provides real-time information, but the sheer volume of posts and informal language make efficiently extracting relevant details a challenge. Traditional Natural Language Processing (NLP) methods like bag-of-words and Term Frequency Inverse-Document Frequency (TF-IDF) offer basic classification but struggle with nuance. Deep learning models, including Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-Trained Transformers (GPT), improve accuracy but require substantial

labeled data and computational power. Rule-based approaches, while interpretable, lack adaptability to evolving language trends.

Studies have applied NLP and machine learning to classify disaster tweets. Traditional models like Naïve Bayes and Support Vector Machine (SVM) work well for basic classification but fail to capture the complexity of informal social media language. Deep learning models such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and transformers (e.g., BERT) outperform traditional methods but require significant data and resources.

Hybrid models combining rule-based filtering with machine learning offer interpretability but struggle with evolving terminology. Feature engineering, including sentiment analysis and keyword extraction, has improved classification but still faces challenges like noise filtering and multilingual tweets.

This project builds on prior work by integrating traditional NLP with advanced machine learning techniques to develop a scalable, high-accuracy disaster tweet classification model.

Project Goal:
The objective of this project is to develop a model that accurately differentiates disaster-related tweets from unrelated content. By improving classification accuracy, this work contributes to the field of NLP and provides emergency management agencies with a reliable tool to rapidly filter and respond to critical information.

2. Data and Exploratory Analysis

The dataset, hosted on Kaggle as a prediction competition, consists of various independent features and one target.

Data Dictionary:

1. id - a unique identifier for each tweet
2. text - the text of the tweet
3. location - the location the tweet was sent from (user input, may be blank)
4. keyword - a particular keyword from the tweet (may be blank)
5. target - denotes whether a tweet indicates a real disaster (1) or not (0). This is intended for model training and validation. The "**target**" feature is removed in **test.csv**, the testing data.

Data Cleaning and Preprocessing Methods:
The data cleaning and preprocessing steps were designed to address both the unique challenges posed by tweet text and the general need for consistency across datasets. The methods implemented were aimed at enhancing the quality of the data for downstream Natural Language Processing (NLP) tasks.

Text Cleaning and Normalization -

We pursued two different approaches to cleaning the data. These variations of the dataset will provide us an additional way to evaluate how our models fit the data. One specific difference in the two approaches revolved around the handling of mentions and hashtags. One school of thought we had was that mentions and hashtags should be removed so that the models could focus on the core text of the tweets. An alternative school of thought is that the mentions and hashtags in a tweet might be good indicators of whether or not that tweet is disaster-related. Instead of clinging to one approach, we decided that we could run the same models on the two variations of the data, and see if one approach produced significantly better results. So far, we have only run models on the data cleaned with the first method, but in a future iteration we will also run models on the second variation and compare the results. Here is an overview of the procedures we followed:

Method 1:

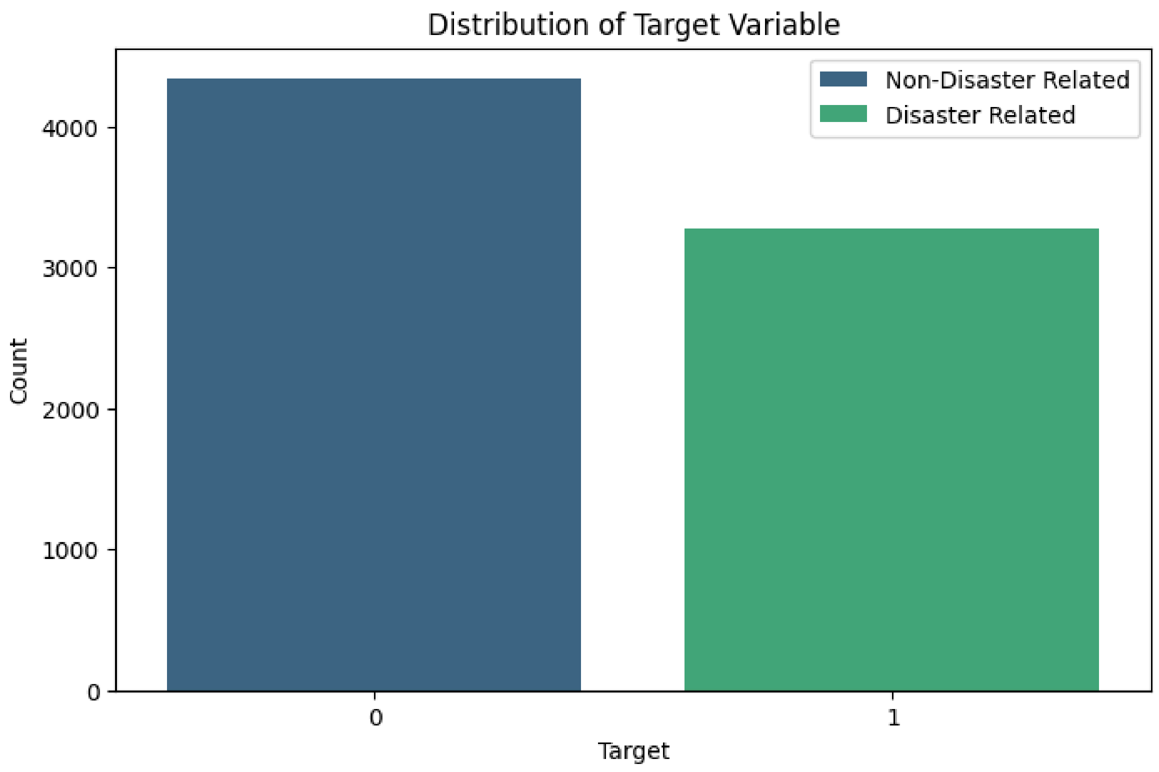
- Removed duplicate tweets to prevent bias from repeated information.
- Stripped URLs, mentions, hashtags (unless they were deemed relevant to the context), and special characters, as they do not add meaningful information for the analysis.
- Converted all text to lowercase for uniformity, eliminating discrepancies between uppercase and lowercase variations of the same word.
- Removed numbers and unnecessary punctuation, as they often introduce noise without contributing to the meaning in tweet text.
- Applied custom stopword filtering to eliminate common, irrelevant terms such as "https," "rt," and other non-essential tokens that frequently appear in tweets.

Method 2:

- Repaired or removed corrupted characters introduced with the importation of the dataset.
- As in Method 1, removed all rows of data that were duplicated in the "text" feature.
- Tokenized the "text" column using both TweetTokenizer and word_tokenize from Natural Language Tool Kit (NLTK) to break the text up into meaningful components.
- As in Method 1, applied custom stopword filtering.
- Removed the keyword and location features as we were unable to determine if these were added before or after the target value was known.

Initial Exploratory Data Analysis

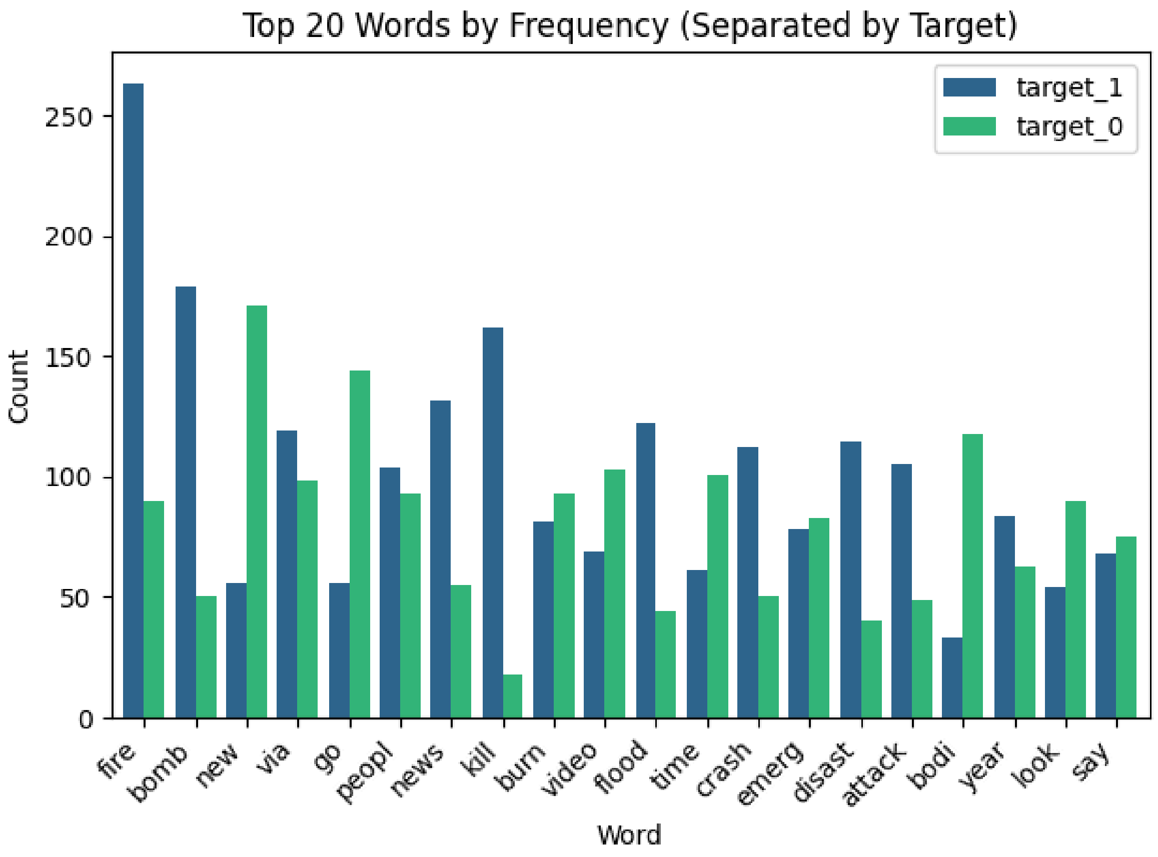
Now we will look at some of the initial findings of our exploratory data analysis. The dataset has a slight imbalance, with approximately 57% of tweets classified as non-disasters. This is important to note as it may affect the performance of our models, but with this rate distribution, balancing should not be necessary.

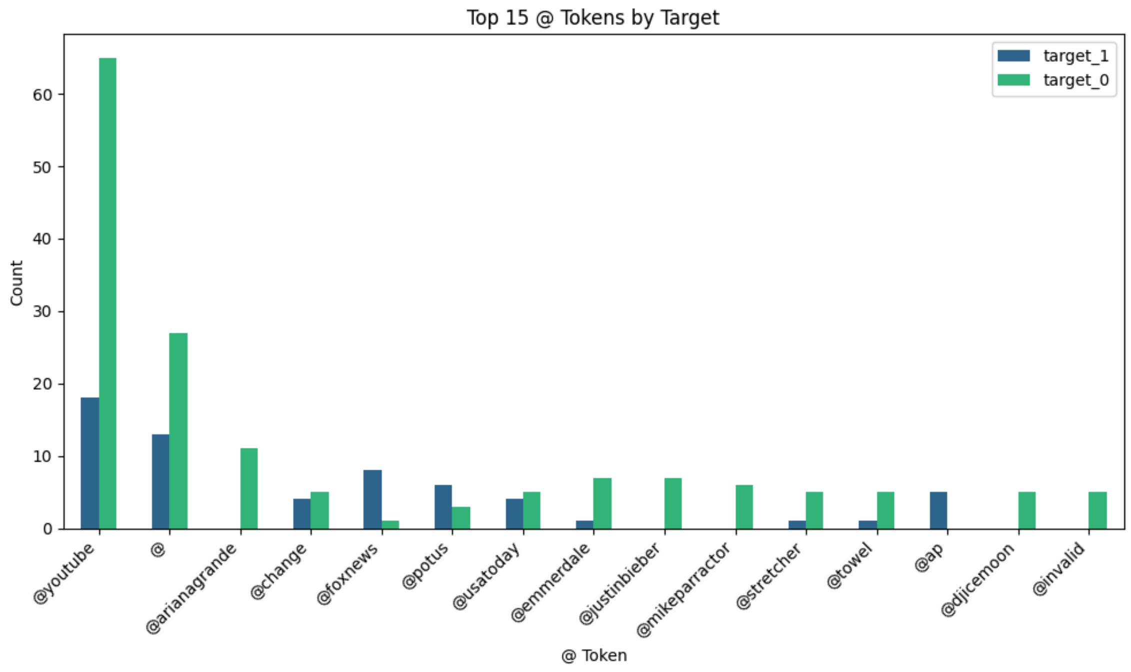


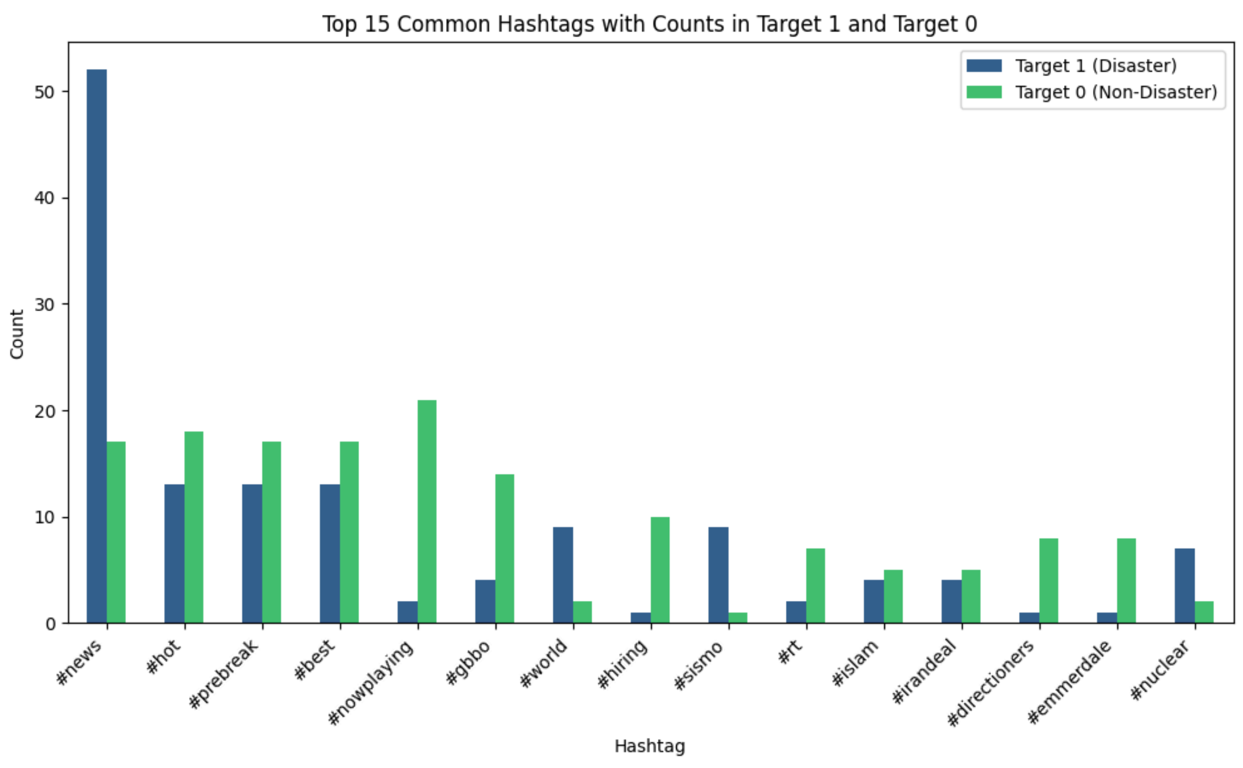
Data Analysis & Observations:
We continued analysis of tweets by examining the various statistics involving the text features in the training dataset. We started off by checking the statistics of all the tweets, and then the statistics of the individual classes. The goal was to find any patterns or trends in the tweets related to positive (disaster) or negative classes (non-disaster). From the data we found the following statistics:

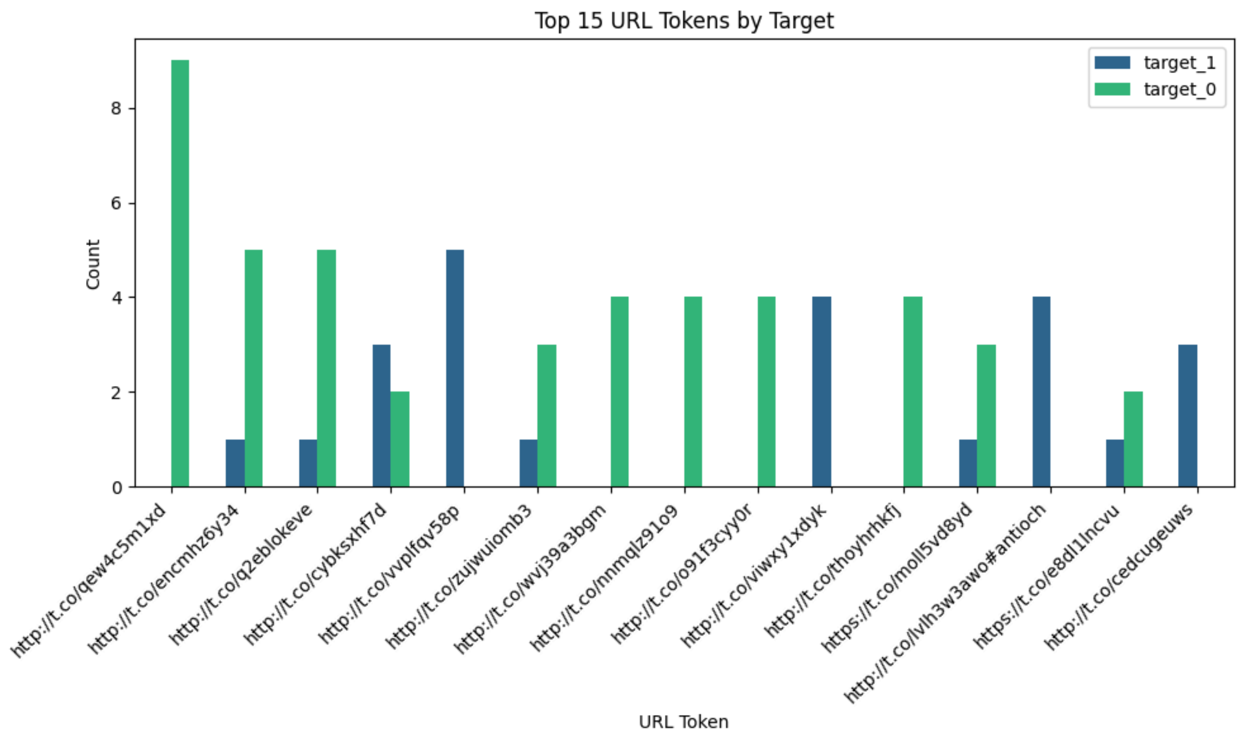
Feature	Value
Count	7613.0
Mean	101.0
Std	33.8
Min	7.0
25%	78.0
50%	107.0
75%	133.0
max	157.0

Looking at the dataset on a word by word basis, with minimal processing we were able to make a visual comparison of the most frequent words, hashtags, mentions, and URLs along with how they compared in the target values.



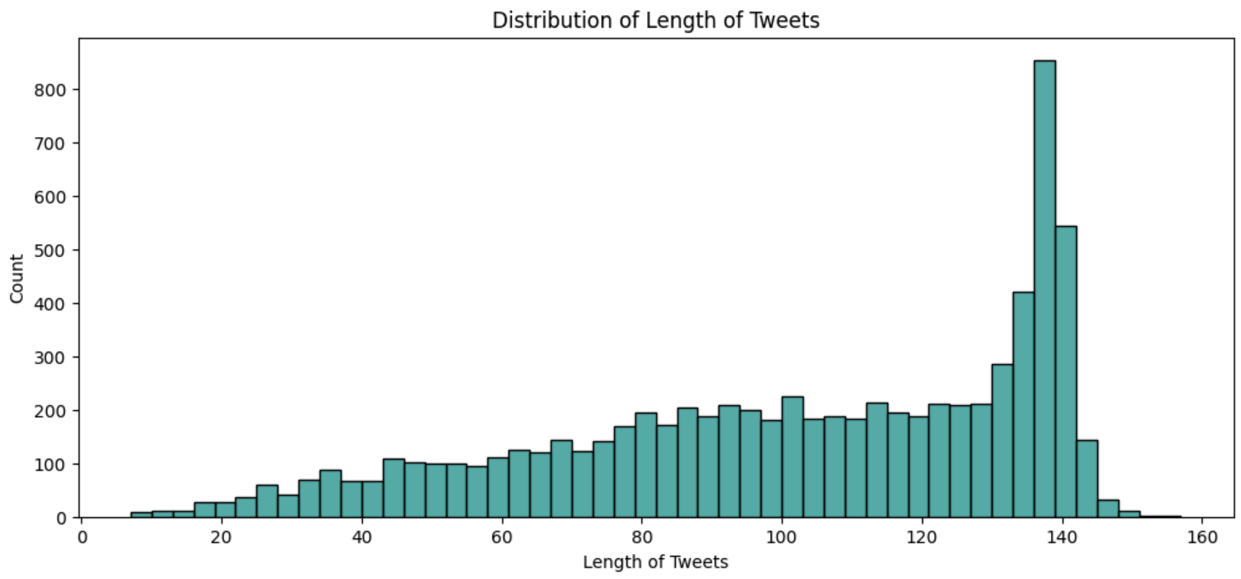




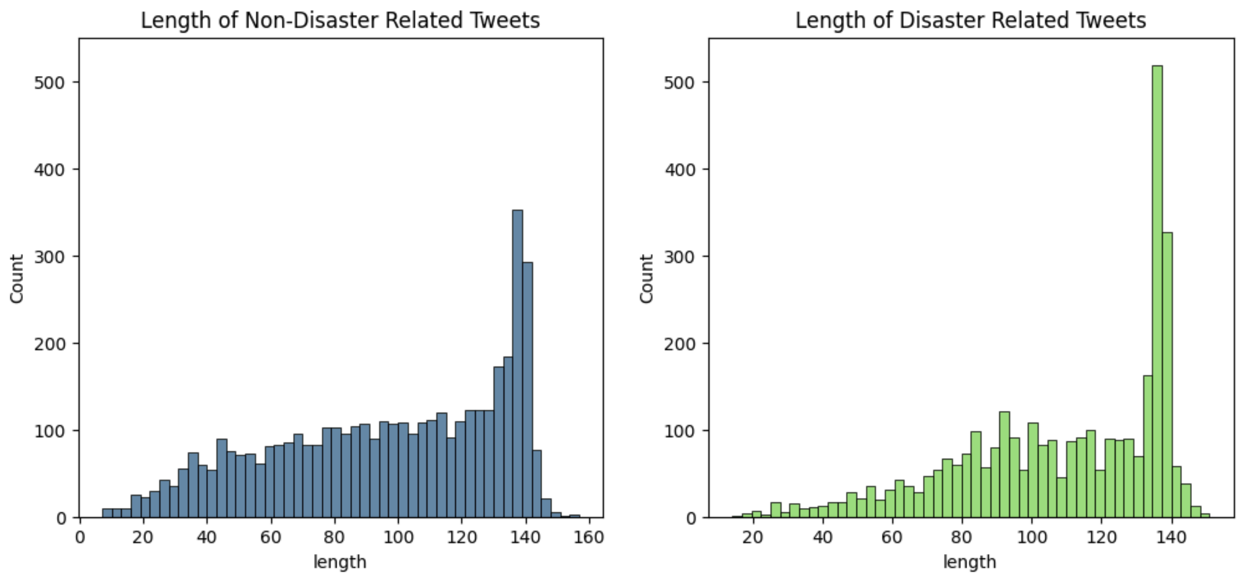


The dataset has an average tweet length of 101 words, with a range between 7 and 157 words. Disaster-related tweets tend to be longer and more descriptive, often including details about the event and its impact. Non-disaster tweets typically contain more conversational language.

Outliers, such as excessively long tweets with repeated punctuation (e.g., multiple exclamation marks), were identified and addressed.



The length of most tweets falls between 70 and 125 words, with a few outliers. The distribution is slightly left-skewed, indicating a higher number of shorter tweets. Further analysis examined length distributions across both target classes to identify differences.

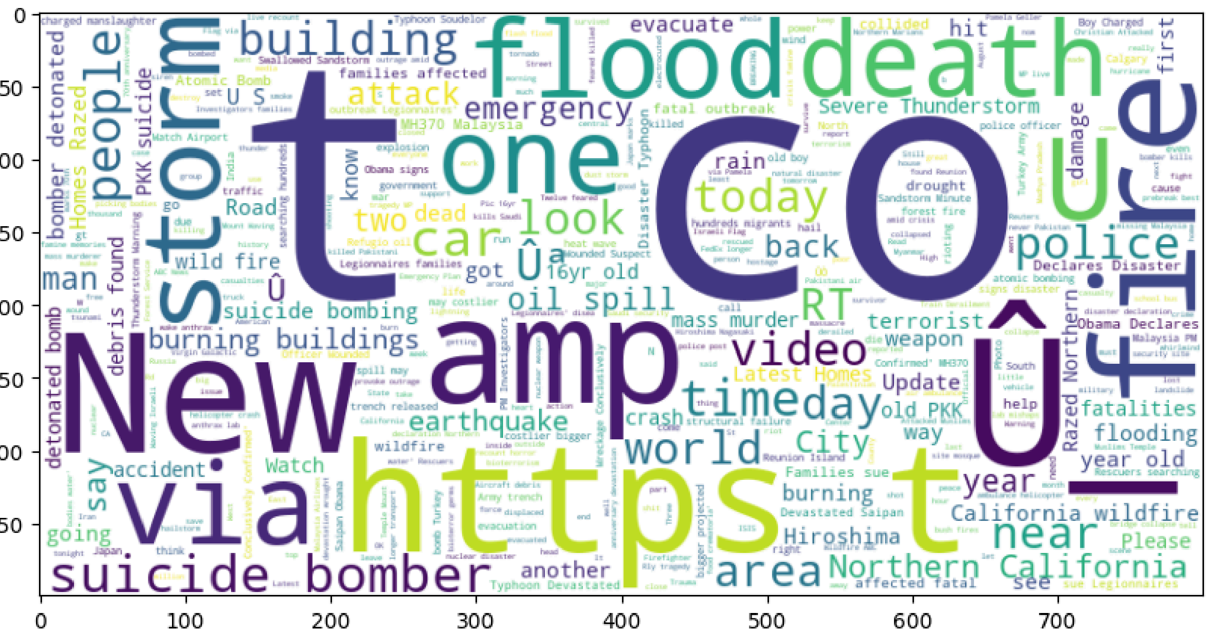


Medium-length tweets are more prevalent in the negative class, while disaster-related (positive class) tweets tend to be longer. This pattern likely arises because disaster tweets often convey critical details about the event, including its location, causes, and impact, making them more descriptive. In contrast, non-disaster tweets are generally shorter and less detailed. Recognizing these trends helped inform our data cleaning process, ensuring that text length variations were properly accounted for before modeling.

Data Visualization

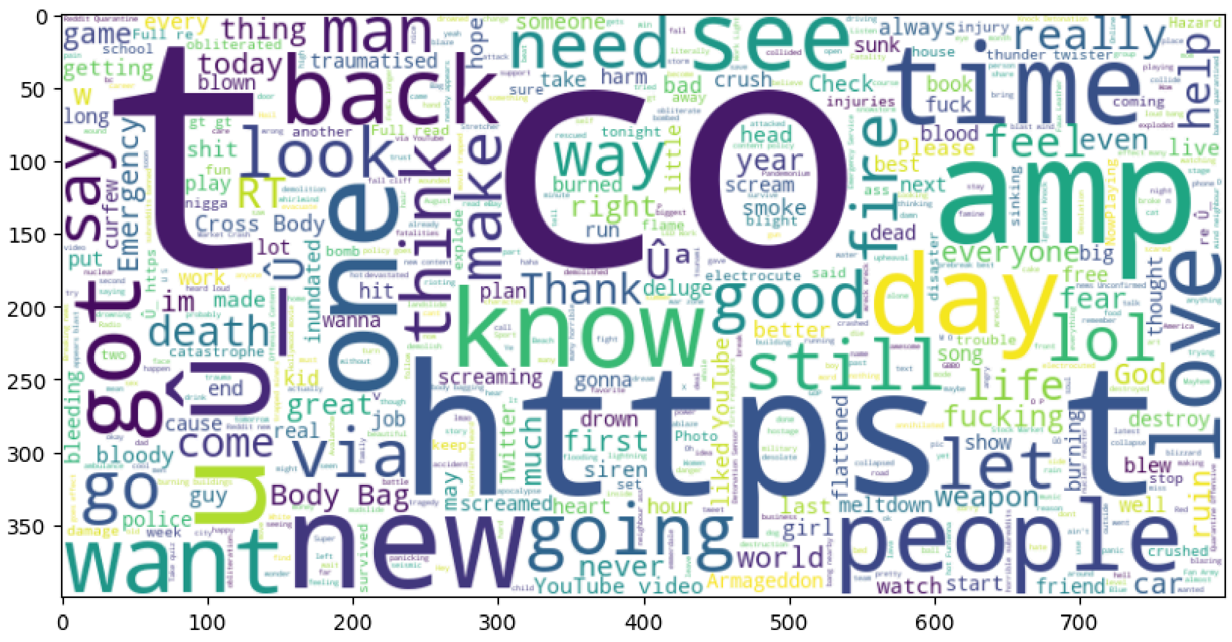
After initial data cleaning, a word cloud was generated to form a visual representation of the most common words used in the dataset, with the size of each word representing its frequency. This allowed for visual identification of patterns and trends in the data as well as giving a general sense of word distribution. Two word clouds were formed, one for disaster related tweets, and one for non-disaster related tweets.

Disaster Word Cloud:



The disaster word cloud highlights the most frequently mentioned terms in disaster-related tweets, including "fire," "flood," "storm," "suicide bomber," "people," "police," "oil spill," "burning buildings," "California wildfire," and "Northern California." At first glance, the word cloud suggests that many of the disaster-related tweets focus specifically on wildfire outbreaks in California.

Non-Disaster Word Cloud



Non-disaster tweets contained words such as “love,” “help,” and “thank,” with occasional overlap in words like “fire” and “emergency,” used in different contexts. Noise words like “https” and “CO” were removed using custom stopwords. Porter stemming was applied to standardize variations in spelling.

One common feature among both sets of tweets is the abundance of words such as "https", "CO", and "t". We needed to deal with these common noisy words before doing further analysis, so as to keep them from throwing off our models. So the addition of custom stop words were used to account for the common noise components. Stemming (with the Porter stemmer) was carried out on the tweets to account for any regional spelling differences.

Feature Engineering:
The cleaned dataset was split into training (80%) and testing (20%) sets. Key features included:

- Text features (X) – Processed tweet content.
- Target variable (y) – Disaster classification label.

Bi-Grams & Tri-Grams:
Although tweets are typically short, a TF-IDF Vectorizer was applied to detect meaningful two- and three-word phrases. Common words appearing in over 80% of tweets were filtered out to avoid redundancy. Converting text into numerical form using vectorization optimized processing efficiency for large-scale text-based data.

This cleaned dataset was then used for model development and evaluation, with further analysis planned to explore transformer-based contextual models for improved classification accuracy. It was fed into an initial fitted and cross validated model comparing baseline Multinomial Naive Bayes with a Passive Aggressive Classifier to compare initial results. Then the same process was applied to our other selected models using pipeline features to feed the fitted data into each model.

3. Methods for Model Evaluation

The following models were run after implementing 5-fold cross-validation on the cleaned dataset, which was processed through a pipeline into each model:

Multinomial Naive Bayes (MultinomialNB):

This model is a probabilistic classifier based on Bayes' theorem, commonly used for text classification tasks. It assumes that word occurrences are conditionally independent given the class label. The model is well-suited for high-dimensional sparse data, such as TF-IDF vectorized text, and serves as a strong baseline for comparison against more complex models.

Logistic Regression:

Logistic Regression is a linear model that estimates the probability of a tweet belonging to a particular class by applying the sigmoid function to a weighted sum of input features. It is particularly useful for its interpretability, as it provides insight into the influence of individual words on classification outcomes.

Support Vector Machine (SVM):

SVM is a classification algorithm that finds the optimal hyperplane to separate data points in a high-dimensional space. In this implementation, a linear kernel is used to classify tweets by maximizing the margin between classes. SVM is effective in handling high-dimensional text data and is particularly useful when the dataset is linearly separable.

Passive Aggressive Classifier:

The Passive Aggressive Classifier is an online learning algorithm designed for large-scale classification problems. Unlike traditional models that converge to a fixed solution, this model updates its decision boundary dynamically as new training examples are introduced. It is particularly well-suited for streaming data or real-time classification tasks, making it a strong choice for scenarios where tweets arrive continuously over time.

K-Nearest Neighbors (KNN):

KNN is a non-parametric model that classifies new data points based on their proximity to labeled examples in the training set. The number of neighbors (k) determines how many closest points are considered when assigning a class label. Although KNN can perform well in certain text classification tasks, it is computationally expensive for large datasets, as it requires storing and comparing all training instances at prediction time.

Multi-Layer Perceptron (MLP) Classifier:

The MLP classifier is a feedforward neural network that learns complex patterns in text data through multiple layers of interconnected neurons. In this implementation, it consists of a single hidden layer with 50 neurons and uses the ReLU activation function for non-linearity. Optimized with the Adam optimizer, MLP processes TF-IDF vectorized input to capture relationships between words. While not as advanced as transformer-based models, it offers a balance between performance and efficiency, making it a strong choice for capturing non-linear patterns in tweet classification.

Models Under Consideration (Future Exploration):

Due to time constraints, the following models have not yet been implemented, but they are planned for future testing to enhance model performance.

Transformer-Based Models:

Transformer-based models, such as BERT or GPT, will be tested for fine-tuning to directly leverage contextual embeddings. These models have shown promise in understanding nuanced meanings within text, especially in tasks involving informal language like tweets. Their ability to capture long-range dependencies and context makes them highly effective for natural language processing tasks.

Additional Models for Future Consideration:

Beyond the current models in our pipeline, additional techniques could be explored to further improve performance:

- **Random Forest Classifier:** An ensemble method that uses multiple decision trees to improve robustness and reduce overfitting.
- **XGBoost:** A gradient boosting framework known for its efficiency and high performance in structured data classification tasks.
- **LSTM (Long Short-Term Memory Networks):** A type of recurrent neural network (RNN) that is effective at capturing sequential dependencies in text data.

4. Tools Used for Model Evaluation

To efficiently clean, visualize, and evaluate the models, we employed a suite of tools and Python packages, each selected for their relevance to the problem at hand and their ability to support various stages of data preparation, analysis, and model evaluation. Below is a breakdown of the tools used, how they were applied, and the reasons for their selection:

Tools Used and Their Justification -

1. Pandas
Purpose: Data manipulation, cleaning, and initial exploratory analysis.
Justification: Pandas is a powerful and flexible library for working with structured data. It was essential for handling and cleaning our tweet dataset, such as removing duplicates, filtering out irrelevant content, and transforming the data into a more usable format. The ease of use, especially for manipulating data frames, made it an ideal choice for the preprocessing phase.
Evaluation: Pandas performed well for data manipulation and cleaning. It allowed for quick transformations, but some challenges arose with handling missing data in certain features, which required further preprocessing steps.

2. NLTK (Natural Language Toolkit)
Purpose: Text tokenization, stemming, and part-of-speech tagging.
Justification: NLTK was chosen for its strengths in processing natural language text. NLTK's tokenizer and Porter Stemmer were used for simple tokenization and stemming
Evaluation: NLTKs built in methods were efficient for tokenization using its tweet tokenizer function for use in preparing the data for analysis.

3. Scikit-learn
Purpose: Model training and evaluation, including classification algorithms and utility functions for cross-validation and performance metrics.
Justification: Scikit-learn is a comprehensive machine learning library that provides implementations of essential algorithms such as Multinomial Naive Bayes, Logistic Regression, and Support Vector Machines (SVM). It also offers tools for splitting data, cross-validation, and model evaluation (e.g., accuracy, precision, recall, ROC curves).
Evaluation: Scikit-learn was crucial in our model-building process, offering a user-friendly interface for testing multiple algorithms and evaluating their performance. However, we encountered some limitations when trying to scale up models for more complex tasks. Future improvements could include better hyperparameter tuning and the integration of pipeline procedures for more seamless model comparison.

4. Matplotlib and Seaborn
Purpose: Data visualization for distributions, word clouds, and model performance metrics.
Justification: Matplotlib and Seaborn were selected for their ability to create a variety of visualizations, including distributions, word clouds, and model performance metrics (e.g., ROC curves). These tools allowed us to better understand the data and model behavior visually.
Evaluation: Both libraries worked well for our visualization needs. Matplotlib offered flexibility, while Seaborn provided more polished and easily interpretable plots. However, further exploration of more interactive visualization tools could provide deeper insights, especially when analyzing model performance across different subsets of data.

5. PyLDAVis ((Not yet implemented)
Purpose: Interactive topic modeling and visualization.
Justification: PyLDAVis was intended to help visualize the topics generated by Latent Dirichlet Allocation (LDA) for topic modeling. This tool is particularly useful when exploring tweet data

to identify dominant themes in the corpus.
Evaluation: While not yet implemented, PyLDAVis could prove valuable for exploring tweet content and generating insights about the broader context of the disaster-related tweets. This tool could enhance our understanding of the data before classification, and we plan to integrate it in future iterations.

6. SpaCY (Not yet implemented)
Purpose: Text tokenization, stemming, and part-of-speech tagging.
Justification: SpaCy will be used as an additional tokenization method for comparison against NLTK for the models. SpaCy's advanced tokenization pipeline could potentially handle complex cases, such as slang and informal text. SpaCy also provided part-of-speech tagging and dependency parsing, which helped with deeper semantic understanding.
Evaluation: While not yet implemented, SpaCY should be able to handle the short and informal nature of tweets better more efficiently than NLTK.

Future Improvements and Tools Not Used
While the above tools provided a solid foundation for data cleaning, modeling, and evaluation, there are areas where we could make improvements or explore additional tools:

- **Hyperparameter Tuning:** We plan to implement more advanced hyperparameter tuning techniques (e.g., GridSearchCV or RandomizedSearchCV) within Scikit-learn to better optimize model performance.
- **Hugging Face Transformers:** Although not yet explored, Transformer-based models from Hugging Face could provide more advanced approaches to tweet classification by leveraging pre-trained models for fine-tuning tasks. This approach could enhance the model's contextual understanding even further, particularly for informal and domain-specific language used in tweets.

In summary, the selected tools helped us create a robust workflow for data cleaning, modeling, and evaluation. While there were some challenges in terms of computational cost and optimization, the chosen libraries performed well for their intended tasks, and we are excited to explore additional tools and improvements in future iterations of the project.

5. Initial Results

Evaluation Metrics:
We assessed model performance using the following key metrics:

- **Accuracy:** The overall percentage of correct classifications across all predictions.
- **Precision:** The proportion of predicted disaster tweets that were correctly classified as disaster-related.
- **Sensitivity (Recall):** The ability of the model to correctly identify all disaster-related tweets.

- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance, particularly in cases where there is an uneven class distribution.
- **AUC/ROC (Not yet implemented):** A visual and quantitative assessment of the model's ability to distinguish between the positive and negative classes, which will be integrated in future iterations.

For our initial implementation, we used a Multinomial Naive Bayes classifier as a baseline model for classifying disaster-related tweets. The performance metrics obtained from cross-validation tests for this model are summarized as follows:

Metric	Multinomial Naive Bayes	Passive Aggressive Classifier	Logistic Regression	Support-Vector Machine	K-Nearest Neighbors	MLP Classifier (NN)
Accuracy	0.7969	0.7846	0.7864	0.6110	0.6849	0.7951
Precision	0.8607	0.7609	0.7609	1.000(?)	0.8775	0.7903
Recall	0.6304	0.7288	0.5885	0.0964	0.3340	0.7136
F1 Score	0.7276	0.7443	0.7034	0.1757	0.4458	0.7499
ROC AUC	0.8481	0.8399	0.8465	0.8524	0.7383	0.8463

Detailed Model Analysis (Not yet finished) -
The following models were evaluated using 10-fold cross-validation, and their performance was assessed based on accuracy, precision, recall, F1 score, and ROC AUC. The results provide insights into how well each model balances classification performance for disaster-related tweets.

Multinomial Naive Bayes (MultinomialNB):

The Multinomial Naive Bayes classifier performed the best in terms of overall accuracy (**0.7969**) and achieved a strong F1 score (**0.7276**). Its high precision (**0.8607**) indicates that it effectively avoids false positives, meaning it rarely misclassifies non-disaster tweets as disaster-related. However, its recall (**0.6304**) is comparatively lower, meaning it may miss some actual disaster tweets. This suggests that while the model is reliable for precision, improvements could be made to capture more true positive disaster-related tweets.

Passive Aggressive Classifier:

This model achieved competitive results, with an accuracy of **0.7846** and an F1 score of **0.7443**, slightly outperforming Naïve Bayes in terms of recall (**0.7288**). The Passive Aggressive Classifier adapts well to new examples, making it a strong candidate for real-time disaster detection. However, its precision (**0.7609**) is lower than that of Naïve Bayes, indicating a slightly higher tendency to misclassify non-disaster tweets as disasters.

Logistic Regression:

Logistic Regression produced a balanced performance with an accuracy of **0.7864** and an F1 score of **0.7034**. While its precision (**0.8743**) was relatively high, its recall (**0.5885**) was lower, meaning it excels at correctly identifying non-disaster tweets but may miss disaster-related ones. The model's AUC score (**0.8466**) suggests that it distinguishes well between classes, though recall could be improved to capture more disaster-related tweets.

Support Vector Machine (SVM):

SVM had the highest precision (**1.0000**) but suffered from extremely poor recall (**0.0965**) and an F1 score of **0.1758**. This indicates that while it classifies non-disaster tweets with near-perfect accuracy, it struggles to correctly identify disaster-related tweets, leading to a high number of false negatives. Despite its high ROC AUC score (**0.8524**), the imbalance between precision and recall suggests that SVM, in its current configuration, may not be suitable for this classification task and I do not think the model is correctly implemented, as such it will be ignored for now.

K-Nearest Neighbors (KNN):

The KNN model exhibited lower performance overall, with an accuracy of **0.6849** and an F1 score of **0.4458**. Its precision (**0.8775**) was high, but recall (**0.3340**) was low, meaning it frequently misclassified disaster-related tweets. This is likely due to KNN’s sensitivity to high-dimensional data and its reliance on local patterns, which may not generalize well to complex textual data.

Multi-Layer Perceptron (MLP) Classifier:

The MLP model performed well, with an accuracy of **0.7951**, an F1 score of **0.7499**, and a recall of **0.7136**, making it one of the most balanced models in the evaluation. It captures complex relationships in the data better than traditional linear models, resulting in a strong trade-off between precision (**0.7909**) and recall. Its ROC AUC score (**0.8464**) further indicates that it effectively differentiates between disaster and non-disaster tweets.

Key Insights and Model Considerations:

- **Best Model for F1 Score:** The **Neural Network (MLPClassifier)** achieved the highest F1 score (**0.7499**), making it the most balanced model for disaster tweet classification.
- **Best Model for Recall:** The **Passive Aggressive Classifier** had the highest recall (**0.7288**), meaning it was the best at identifying disaster-related tweets.

- **Best Model for Precision:** The **Multinomial Naive Bayes** model had a strong precision (**0.8607**) while maintaining a good balance with recall, making it a reliable classifier for distinguishing between disaster and non-disaster tweets.
- **Best Overall Model: Multinomial Naive Bayes** had the highest accuracy (**0.7969**) and strong precision-recall balance, making it a strong candidate for deployment.

Future Considerations:
To improve recall while maintaining high precision, the following strategies should be considered:

1. **Hyperparameter Tuning:** Adjusting model parameters for better recall without sacrificing too much precision.
2. **Ensemble Methods:** Combining multiple models (e.g., Naïve Bayes and Neural Networks) to balance strengths and weaknesses.
3. **Transformer-Based Models:** BERT and GPT-based models could enhance contextual understanding and improve recall for disaster tweets.
4. **Feature Engineering:** Exploring word embeddings (e.g., Word2Vec, FastText) instead of TF-IDF to capture richer word relationships.

By refining model selection and tuning parameters, the classification performance can be improved, leading to more reliable disaster tweet detection.

6. Summary and Conclusions for Initial Stages

Summary of Results:
Our initial experiment evaluated multiple models for disaster tweet classification, with **Multinomial Naive Bayes** emerging as the most well-balanced baseline model. It achieved a **mean accuracy of 79.7%**, demonstrating strong overall performance. The model also maintained **high precision at 86.1%** and an **F1 score of 72.8%**, making it a reliable option for classification. However, its **recall of 63.0%** suggests that while it correctly identifies most disaster tweets it classifies, it may fail to detect a significant portion of actual disaster-related tweets.

Key Findings -

Strengths:

- **High Precision:** The model effectively avoids false positives, ensuring that when it predicts a disaster tweet, it is likely to be correct. This is crucial in reducing unnecessary alerts in disaster response systems.
- **Strong Overall Accuracy:** Multinomial Naive Bayes performed well in distinguishing disaster-related tweets from non-disaster tweets, making it a reliable starting point for further model improvements.

Limitations:

- **Lower Recall:** The recall score of **63.0%** indicates that a substantial number of actual disaster tweets are being missed, which could lead to delayed emergency responses in real-world applications.
- **Room for Optimization:** While the model provides a strong foundation, improvements in recall are necessary to create a more robust disaster detection system.

Conclusion and Future Work:

The **Multinomial Naive Bayes classifier** has proven to be an effective baseline, offering high precision while maintaining a reasonable balance between recall and F1 score. However, the model's **recall limitations suggest that further refinement is necessary** to ensure that critical disaster-related tweets are not overlooked.

To enhance performance, we will explore **alternative models and techniques** to improve recall while maintaining or enhancing overall classification accuracy. Future work will include:

1. **Exploring Transformer-Based Models:** Leveraging models like BERT or GPT to improve contextual understanding.
2. **Hyperparameter Tuning:** Optimizing parameters across multiple models to enhance recall without sacrificing precision.
3. **Feature Engineering:** Investigating word embeddings such as Word2Vec or FastText for richer text representations.
4. **Ensemble Learning:** Combining the strengths of multiple models (e.g., Naive Bayes, Neural Networks, and Passive Aggressive Classifier) to create a more balanced and robust classification system.

These future enhancements are critical to **building a more comprehensive disaster tweet classification system**, ensuring timely and accurate disaster detection with minimal false negatives.

7. Appendix

GitHub Repository -
https://github.com/CSC-4260-Advanced-Data-Science-Project/NLP_Disaster_Tweets

Natural Language Processing with Disaster Tweets -
https://www.kaggle.com/competitions/nlp-getting-started/data

Notebooks -
NLP-Preprocessing-and-Model-Analysis.ipynb
(takes around 18 minutes for a fairly decent desktop to run, even after reducing iteration counts)

eda_approach_2.ipynb