Classification of Disaster Tweets

Authored By: Sharon Colson, Thomas Robertson, Caleb Smith, Tania Perdomo-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 -
Initially, we corrected corrupted characters, normalized text (e.g., removed accents), dropped unnecessary columns ("location" and "id"), removed rows containing NaN values after these columns were dropped, eliminated rows with identical "text" but conflicting "target" labels to reduce noise, and removed duplicate rows.

We then created three dataset variations to evaluate preprocessing approaches:

- Prepended "keyword" values to the "text" column.
- Dropped the "keyword" column entirely.
- Retained "keyword" as a separate feature.

Finally, ten preprocessing variants were generated for each of the three baseline datasets, resulting in a total of 3x10=30 processed datasets for model training. Each variant applied a distinct combination of text cleaning methods, as summarized in the following table:

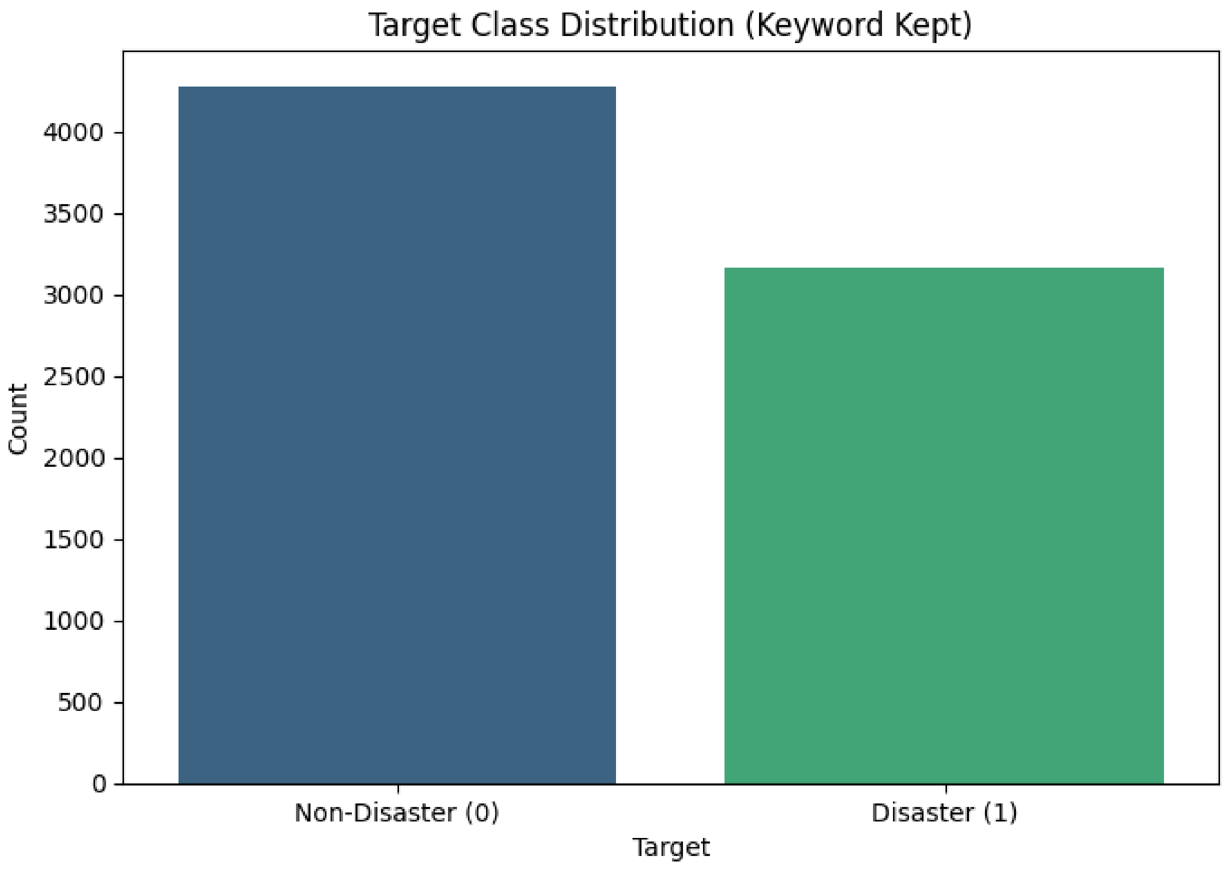
Variant	**Description**
v1_basic_clean	Lowercased, punctuation removed, stopwords removed. Emojis and mentions are **kept**.
v2_no_emojis_mentions	v1 + Emojis and @mentions are removed.
v3_lemmatized	v1 + Lemmatization applied.
v4_stemmed	v1 + Stemming applied.
v5_lemma_stem	v1 + Lemmatization and stemming applied sequentially.
v6_custom_stopwords	v1 + Lemmatization + expanded stopwords list (common Twitter words like ‘rt’, ‘im’, etc.).
v7_lowercase_words_only	v1 + Numbers removed + keep only lowercase alphabetic tokens.
v8_keep_hashtags	v1 + Lemmatization applied + hashtags preserved instead of stripped.
v9_minimal_processing	Minimal cleaning: No stopword removal, no emoji or mention removal, no URL stripping, no lemmatization or stemming (baseline variant).
v10_lemma_stem_custom_stopwords	Lemmatization + stemming + extended stopword list. Emojis and mentions are **kept**.

Following preprocessing, all dataset variants were vectorized using a Term Frequency–Inverse Document Frequency (TF-IDF) representation. The TF-IDF vectorizer was configured to remove English stopwords, ignore terms appearing in more than 80% of documents (`max_df=0.8`), and extract features across unigrams, bigrams, and trigrams (`ngram_range=(1,3)`). No additional manual tokenization or stopword filtering was performed beyond the transformations applied during initial preprocessing. For the purposes of our initial exploratory analysis, we restricted evaluation to the three baseline datasets.

Initial Exploratory Data Analysis

(Note: Most visualizations in this section come from the "Keyword Kept" variation.)

Now we will look at some of the initial findings of our exploratory data analysis. The distribution of our target variable is 57% 0's (non-disaster) and 43% 1's (disaster). According to our preliminary research, a ratio in the 60/40 to 40/60 range is generally considered balanced, so no steps were taken to provide class balancing.

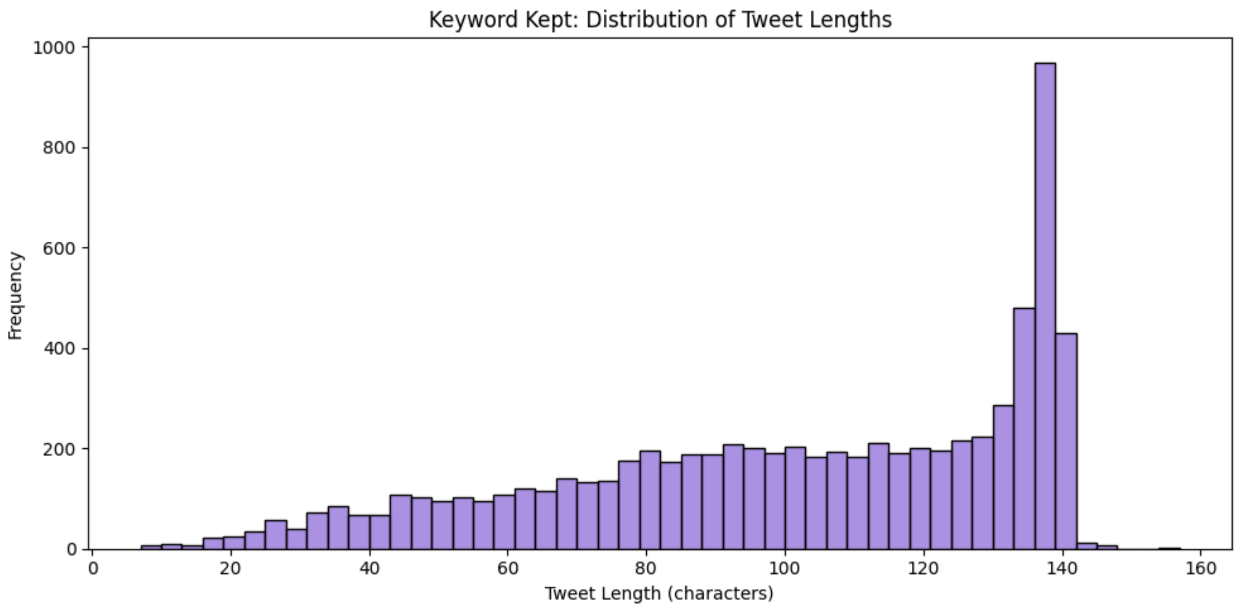


Data Analysis & Observations:
We continued analysis of tweets by examining the various statistics involving the text features in the training dataset. We started by checking the statistics of all the tweets, and then the statistics

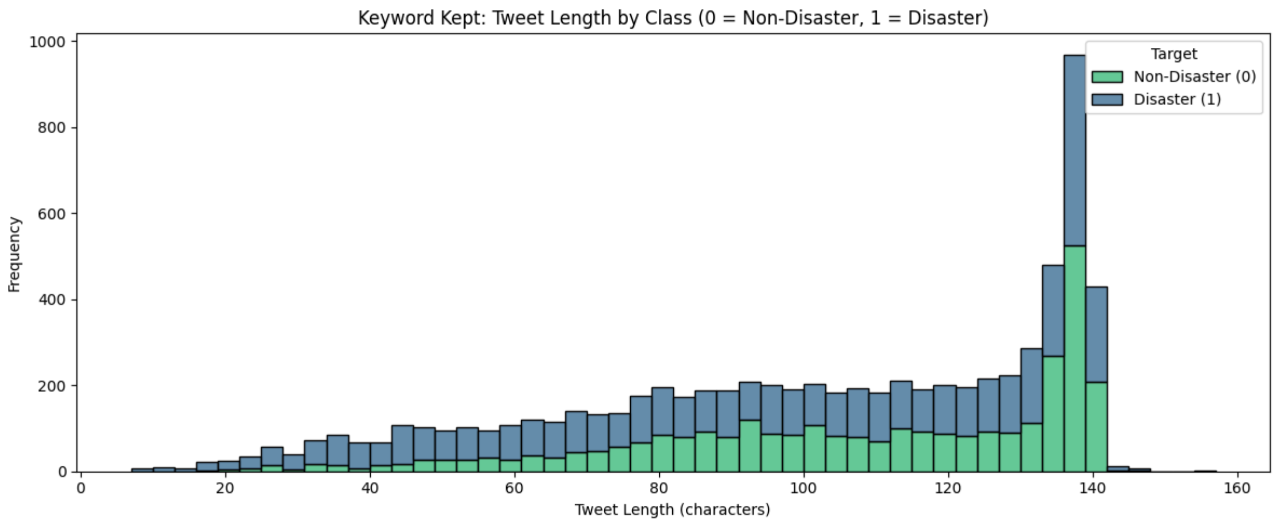
of the individual classes. The goal was to find any patterns or trends in the tweets related to positive (disaster) or negative classes (non-disaster). From the data we found the following statistics regarding the length of our tweets:

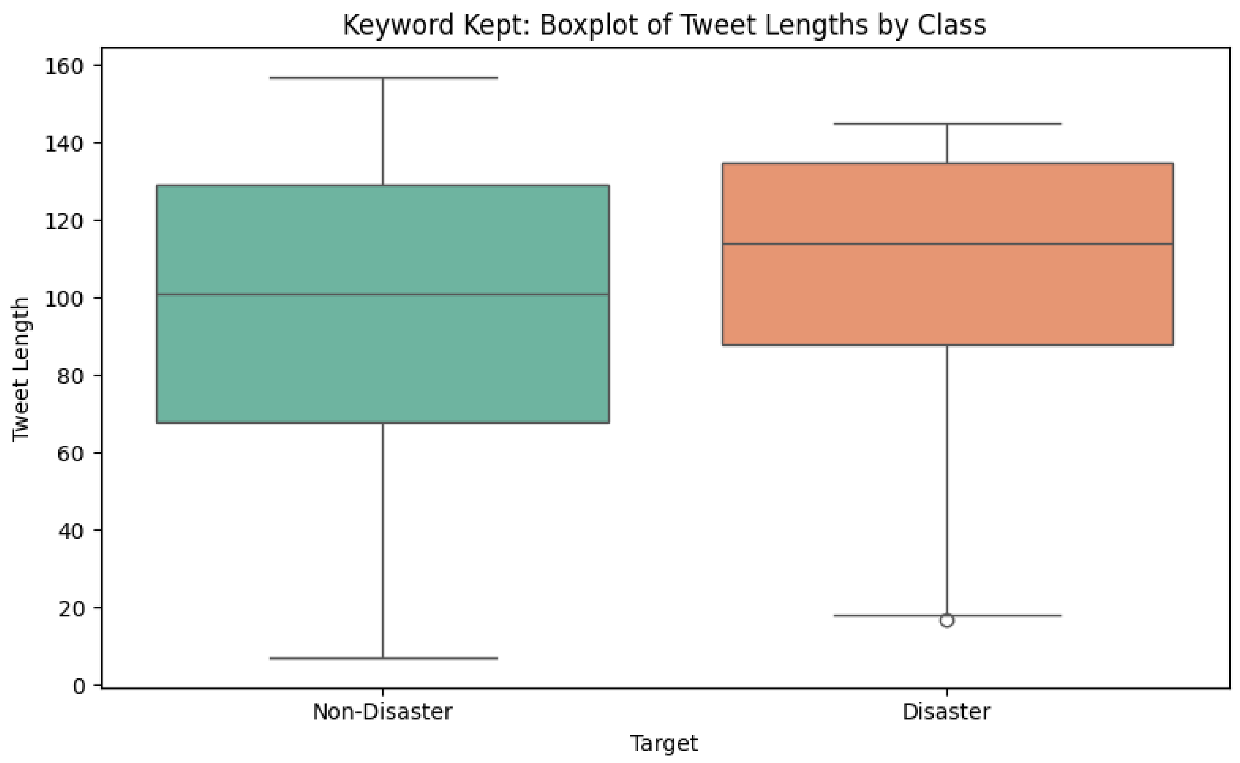
Feature	**Value**
Count	7613.0
Mean	101.0
Std	33.8
Min	7.0
25%	78.0
50%	107.0
75%	133.0
max	157.0

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.



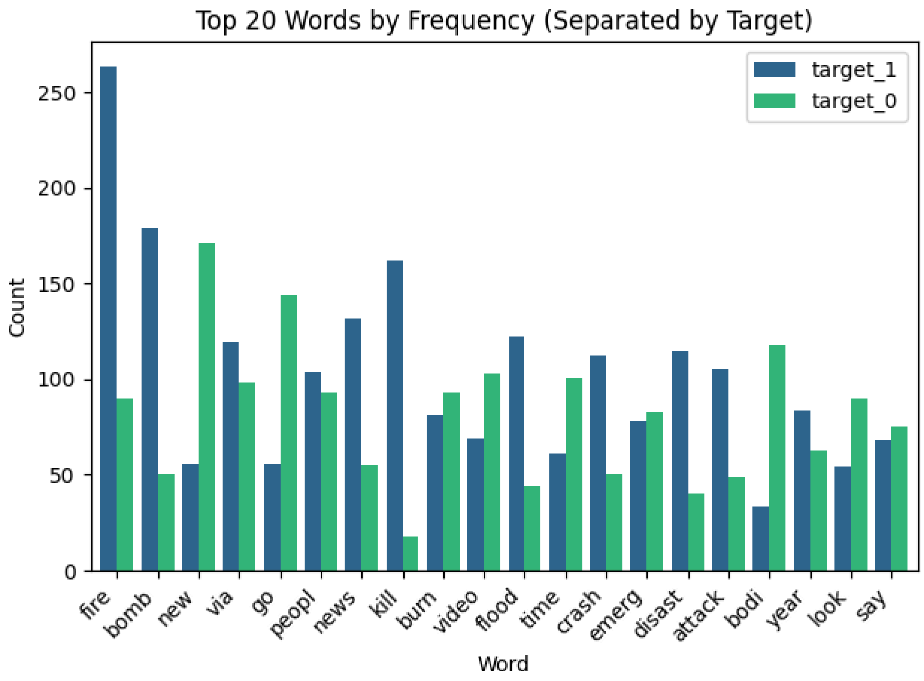
The length of most tweets falls between 70 and 125 words, with a few outliers. The distribution is slightly left-skewed, indicating a higher number of shorter tweets. Further analysis examined length distributions across both target classes to identify differences.

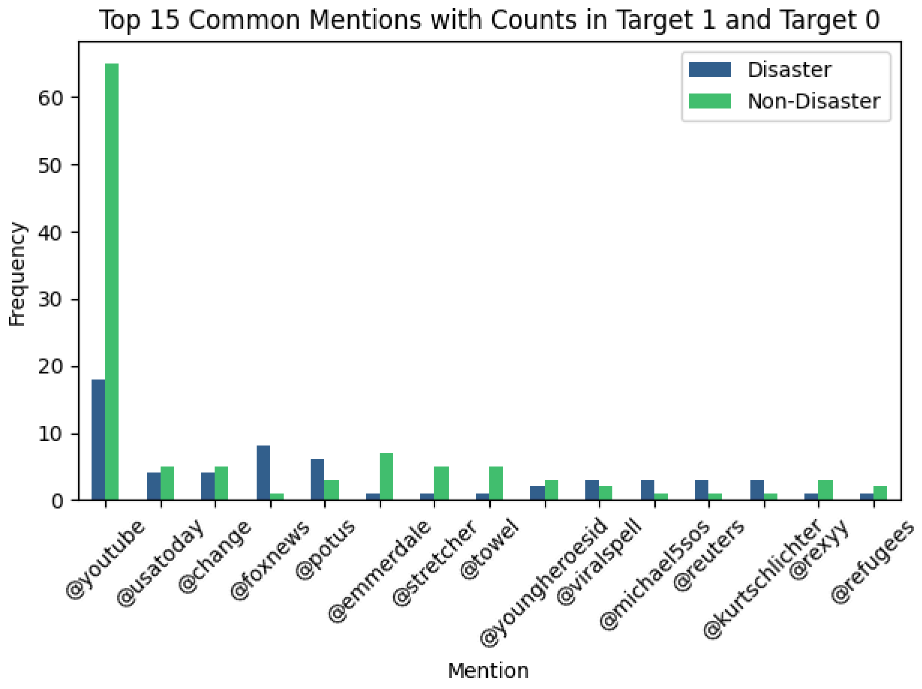


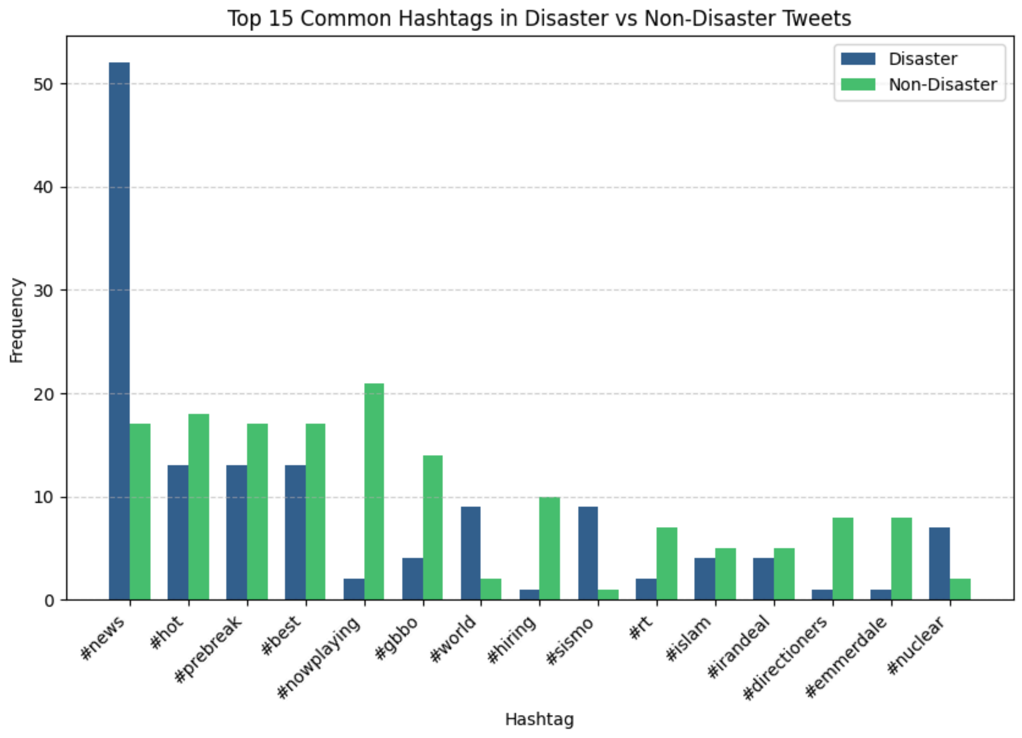


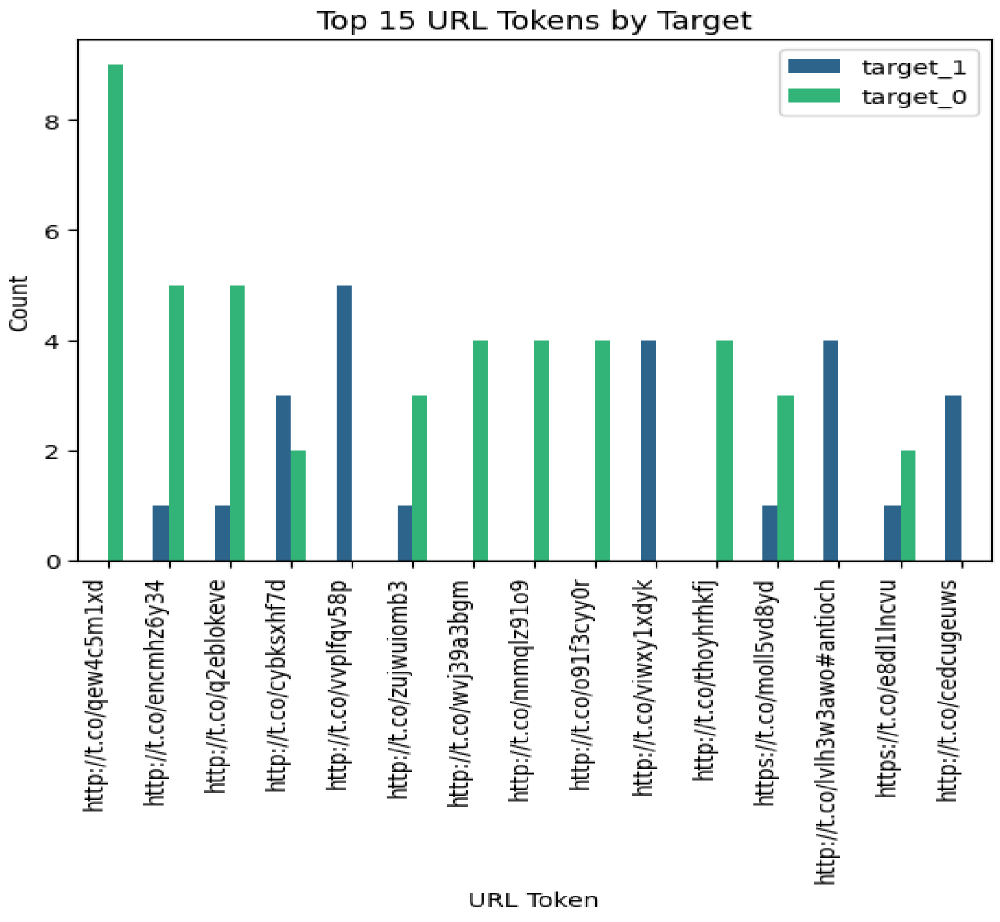
Medium-length tweets are more prevalent in the negative class, while disaster-related (positive class) tweets tend to be longer. This pattern likely arises because disaster tweets often convey critical details about the event, including its location, causes, and impact, making them more descriptive. 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.

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.

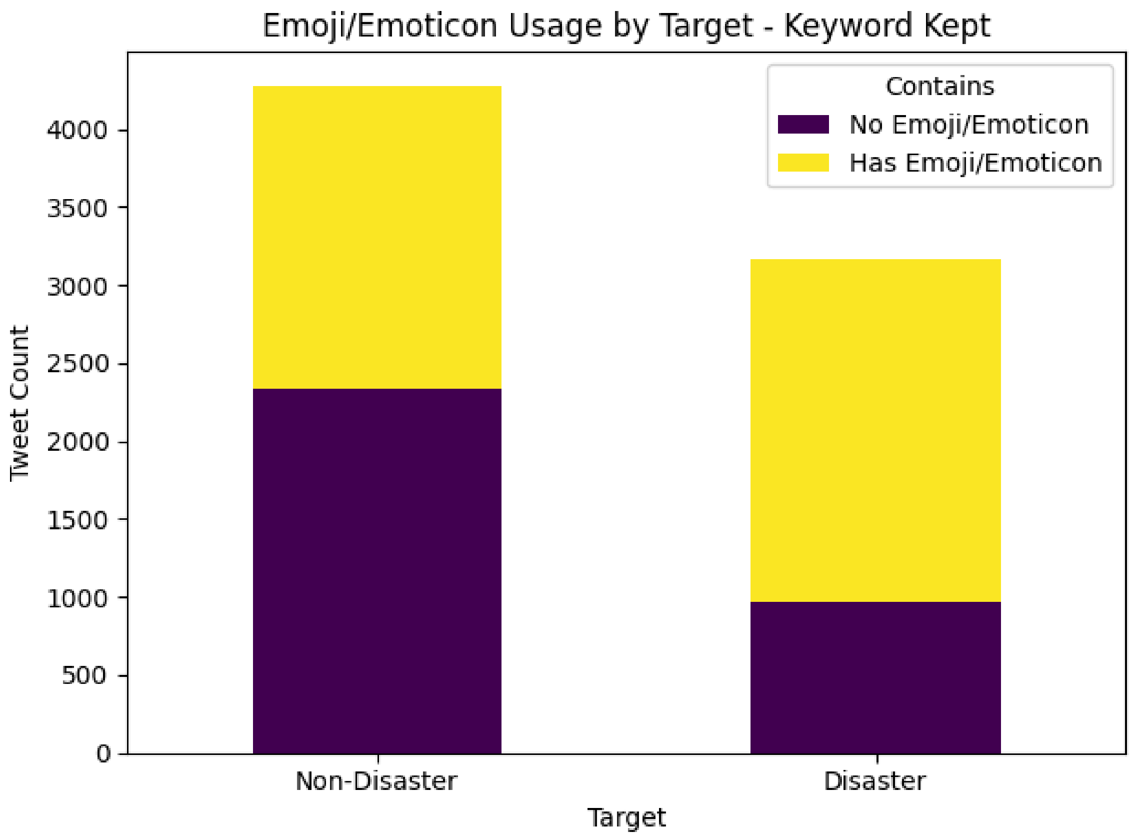





Top 15 Common Hashtags in Disaster vs Non-Disaster Tweets
Disaster
Non-Disaster
Frequency
Hashtag
#news
#hot
#prebreak
#best
#nowplaying
#gbbo
#world
#hiring
#sismo
#rt
#islam
#irandeal
#directioners
#emmerdale
#nuclear


Top 15 URL Tokens by Target
target_1
target_0
Count
URL Token
http://t.co/qew4c5m1xd
http://t.co/encmhz6y34
http://t.co/q2eblokeve
http://t.co/cybksxhf7d
http://t.co/vvplfqv58p
http://t.co/zujwuiomb3
http://t.co/wvj39a3bgm
http://t.co/nnmqlz91o9
http://t.co/o91f3cyy0r
http://t.co/viwxy1xdyk
http://t.co/thoyhrhkfj
https://t.co/moll5vd8yd
http://t.co/lvlh3w3awo#antioch
https://t.co/e8dl1lncvu
http://t.co/cedcugeuws

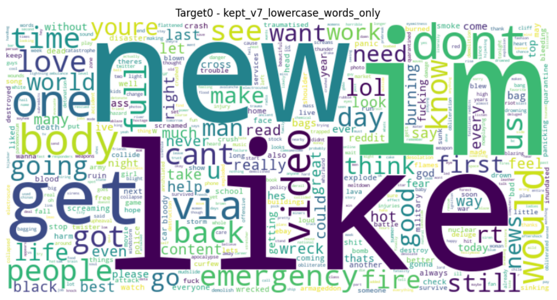
We additionally sought to identify emojis/ASCII emoticons in tweets and determine whether these were more commonly used in disaster related tweets or non-disaster related tweets. The following chart visualizes the ratio of the number of tweets with emojis/emoticons to the number of tweets without emojis/emoticons for both the disaster and non-disaster classes:

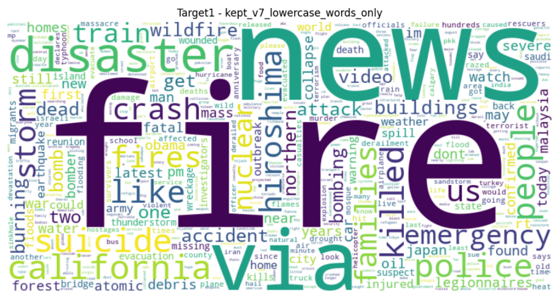


Interestingly, the ratio of disaster tweets with emojis/emoticons appears to be considerably higher than the ratio of non-disaster tweets with emojis/emoticons. It would seem logical that people in a state of panic are not going to take the time to type out emojis, but this data would seem to indicate otherwise. It is worth noting two things, however: 1. Emojis and emoticons can help someone convey their emotions in a message more clearly, and 2. Not everyone tweeting about a disaster is necessarily in a state of panic (they could simply be reporting the disaster or observing it from a distance).

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. Three word clouds were formed: one for disaster related tweets, one for non-disaster related tweets, and one for all tweets. Below we show the target for non disaster(left) and disaster(right).





The disaster word cloud highlights the most frequently mentioned terms in disaster-related tweets, including "fire," "news," "disaster," "california," "family," "suicide," "police," "crash," "home," "train," and "storm." At first glance, the word cloud suggests that many of the disaster-related tweets focus specifically on wildfire outbreaks in California.

Non-disaster tweets contained words such as "new," "love," "day," "time," "back," and "video," with occasional overlap in words like "body," "fire," and "emergency," used in different contexts.

One common feature that had previously been present among both sets of tweets was 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

Following the exploratory phase, each of the 30 processed datasets underwent vectorization using a TF-IDF (Term Frequency-Inverse Document Frequency) approach. The TF-IDF vectorizer was optimized to exclude English stopwords, discard excessively common terms (max_df=0.8), and generate n-gram features (unigrams, bigrams, and trigrams) to capture contextual information.

Updated Modeling Pipeline

The 30 distinct datasets were each passed into a modeling pipeline designed for comprehensive analysis and comparison. The pipeline included the following classifiers, each selected for their suitability in text classification:

- **Multinomial Naive Bayes (MNB)**: A baseline probabilistic model testing for conditional independence.
- **Logistic Regression**: Linear model to estimate probability of tweets belonging to a set class.
- **Passive Aggressive Classifier**: Particularly effective for streaming data due to adaptive decision boundary updates as the model classifies the data.
- **Support Vector Machine (SVM)**: To find the optimal hyperplane separating the data clusters.
- **K-Nearest Neighbors (KNN)**: Non-parametric method to classify text based on its neighbors classification states.

- **Multi-Layer Perceptron (MLP)**: A neural network capable of capturing nonlinear patterns in text data.

Additionally, transformer-based models, **BERT Base Uncased and BERTweet-Base,** were tested separately on the top-performing dataset variants from the initial evaluations.
Each classifier's performance was evaluated using 5-fold cross-validation and the following metrics: Accuracy, Precision, Recall, F1 Score, and ROC AUC.

Integration of Multiple Datasets into Pipeline

The enhanced pipeline methodology involved:
- Initial baseline evaluation comparing Multinomial Naive Bayes and Passive Aggressive classifiers on the baseline datasets.
- Subsequent comparative analysis of the remaining models using the fully vectorized and preprocessed datasets.
- Detailed analysis to identify which preprocessing strategies produced optimal model performance.
- Consistent application of cross-validation for performance estimates across all dataset-model combinations.

This process enabled us to pinpoint the most effective data preprocessing and model selection strategies, significantly improving our classification accuracy and model performance.

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.

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.

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.

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.

BERT and BERTweet Models:

The BERT model leverages Bidirectional sentence representations to discern contextual meaning in natural language text, and can be further trained on downstream tasks like text classification, which is exactly what we are using it for. The first BERT model we are using is BERT-Base-Uncased, which is a general Transformer model for general English text. The second model in our tool belt is BERTweet. Having been specifically trained on raw Tweet data, we have the impression that BERTweet will perform better within the context of the problem we are trying to solve.

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.

Model Hypertuning with GridSearchCV

To further enhance our NLP pipeline, we implemented hyperparameter tuning using Scikit-learn's **GridSearchCV**. Hyperparameter tuning systematically searches through various combinations of hyperparameters to find the optimal set for each model, significantly boosting performance and ensuring our models generalize well to new, unseen data.

The key motivation behind this enhancement was to leverage data-driven optimization rather than relying on default or arbitrarily chosen hyperparameters. This automated approach utilizes cross-validation, which inherently provides robustness and reduces the risk of overfitting by validating hyperparameters across multiple subsets of data.

Below, we detail the rationale and selection criteria for each hyperparameter in our hypertuned pipeline:

- **Multinomial Naive Bayes:** Alpha (Laplace smoothing) prevents zero-probabilities for unseen features, with tested values [0.1, 0.5, 1.0, 1.5, 2.0], balancing noise sensitivity and generalization.
- **Logistic Regression:** Used inverse regularization strength (C: [0.01, 0.1, 1.0, 10.0]), Ridge penalty ('l2') for stability, generalization, and computational efficiency, and 'lbfgs' solver with max iterations [300, 500, 1000] to ensure convergence.
- **Passive Aggressive Classifier:** Regularization strength (C: [0.01, 0.1, 1.0, 10.0]), maximum iterations ([500, 1000, 2000]), and convergence tolerance ([1e-4, 1e-3, 1e-2]) controlled model adherence, accuracy, and training efficiency.
- **Support Vector Machine (SVM):** Tested C values [0.1, 1.0, 10.0], kernel types ('linear' vs. 'rbf'), and gamma ('scale', 'auto') to balance decision boundary complexity and generalization.
- **K-Nearest Neighbors (KNN):** Number of neighbors ([3, 5, 7, 9]), weighting methods ('uniform', 'distance'), and distance metrics (Euclidean vs. Manhattan) were evaluated for performance on normalized and sparse data.
- **Neural Network (MLPClassifier):** Explored hidden layer architectures ([(50,), (100,), (50,50), (100,50)]), activation functions ('relu', 'tanh'), solvers ('adam' efficient vs. 'sgd' tunable), alpha values ([0.0001, 0.001, 0.01]), learning rate strategies ('constant', 'adaptive'), early stopping with validation fractions ([0.1, 0.2]), and iterations without improvement ([5, 10]) to optimize generalization and prevent overfitting.

BERT Model (Pre-trained Transformer Approach)

As part of implementing our original future improvements, our project integrated a pre-trained BERT transformer model, sourced directly from Hugging Face's model repository. Due to the pre-training methods, where models have already learned language representations from large-scale text corpora, hyperparameter tuning was considered redundant for the purpose of this report. The BERT model was run separately from the rest of the pipeline.

Reasons BERT didn't require traditional hypertuning:

- **Pre-trained Weights**: BERT models are extensively pre-trained, capturing linguistic patterns, semantics, and context, thus limiting the need for intensive retraining from scratch.
- **Tuning Simplicity**: The primary focus with BERT typically involves minimal adjusting only the learning rate, epochs, and possibly batch sizes. Standard hyperparameters are documented and widely tested.
- **Generalization**: Due to its contextual understanding and extensive pre-training, BERT is inherently capable of producing strong baseline results without extensive hyperparameter searches.

Thus, our implementation strategy for BERT emphasizes minimal adjustments rather than extensive tuning..

Integration into Evaluation Pipeline

Hyperparameter tuning was conducted using GridSearchCV, performing cross-validation exclusively on the training data. Upon identifying the best hyperparameters, each model was then retrained on the full training dataset with these optimal settings. Finally, the tuned models were evaluated on a separate, previously unseen test set (20% of the data), ensuring an unbiased assessment of model performance and generalization capabilities.

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.

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.

5. GridSearchCV
Purpose: Automated hyperparameter optimization for models within the NLP classification pipeline.
Justification: GridSearchCV systematically explores multiple hyperparameter combinations using cross-validation, optimizing model parameters like regularization strengths, kernel choices, learning rates, and neural architectures. This approach significantly improves generalization, reduces overfitting, and ensures consistent, empirically driven model comparisons.
Evaluation: Integrating GridSearchCV substantially enhanced model performance, robustness, and reliability across all tested classifiers. The systematic hyperparameter optimization approach allowed for informed selection and clear understanding of how each parameter influenced the overall modeling outcomes.

6. Hugging Face Transformers - BERT Model
Purpose: Leveraging a pre-trained transformer model (BERT) for NLP classification tasks.
Justification: Transformer-based models, particularly BERT (Bidirectional Encoder Representations from Transformers), have become a common approach in NLP due to their deep contextual understanding and ability to capture language patterns. By using Hugging Face's Transformers library, we can easily access pre-trained BERT models, significantly reducing the computational resources and data requirements typically needed for large-scale training. Since BERT is pre-trained on extensive general-language corpora, fine-tuning typically involves minimal adjustments, primarily the learning rate, batch size, and number of training epochs,

making traditional exhaustive hyperparameter tuning less critical.
Evaluation: We tuned two pre-trained BERT models BERT-Base-Uncased and BERTweet-Base on our tweet dataset, achieving slightly better results than most of our self-trained models. Since BERTweet was specifically trained on tweet data (unlike the general English-trained BERT), we anticipated superior performance. Initial tests confirmed this expectation, though the improvement was modest.

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 for our Models (Pre-Hyper-Tuning)

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:** A visual and quantitative assessment of the model's ability to distinguish between the positive and negative classes.

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

Initial Model Analysis (Pre-Hyper Tuning)
The following models were evaluated using 5-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 due to the extreme overfitting, and will be reconfigured and explored again after hyper tuning.

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.

Pre-Trained Model Analysis:

The pre-trained BERT models we used were also validated with 5-fold cross validation, and then given a final scoring on a hold-out test set. Rather than being trained and tested on all variations of our dataset, these models were only run on the top three most promising variations from the results of our previous models.

BERT-Base-Uncased:

The BERT-Based-Uncased model from HuggingFace, which we fine-tuned for text classification on our training data, achieved a respectable accuracy of 0.8314, along with an F1 Score, Precision, and Recall of 0.7964, 0.8183, and 0.7757, respectively. This shows that the BERT model was not only highly accurate, but it was also proficient in detecting true positives without misassigning too many false positives.

BERTweet-Base:

The BERTweet-Base model managed to perform even better than the BERT-Base-Uncased model, although the difference was not as significant as one might expect. Its best performance was on kept_v2_no_emojis_mentions and kept_v7_lowercase_words_only, with Accuracy of 0.8408, F1 Score of 0.8081, Precision of 0.8289, and Recall of 0.7883.

Key Insights and Model Considerations:

- BERTweet-Base outperformed BERT-Base-Uncased on all scoring metrics that were measured. In addition, both models tended to perform better than the models we trained from scratch, but the small magnitude of the difference raised an interesting consideration about whether or not more complex and computationally expensive models like BERT are truly worth the slight increase in performance.

6. Results After Hyper-Tuning Parameters

After integrating hyperparameter tuning through GridSearchCV, we systematically evaluated several text preprocessing variations across multiple classification models. Below we summarize the results for the best-performing dataset for each model based on the highest achieved F1 Score.

Model Results after Hypertuning

Metric	Multinomial Naive Bayes	Passive Aggressive Classifier	Logistic Regression	Support-Vector Machine	K-Nearest Neighbors	MLP Classifier (NN)
Accuracy	0.7883	0.7926	0.7936	0.7965	0.7676	0.7893
Precision	0.7964	0.7650	0.7816	0.7913	0.7997	0.7714
Recall	0.6732	0.7392	0.7139	0.7082	0.6057	0.7168
F1 Score	0.7295	0.7518	0.7462	0.7474	0.6884	0.7428
ROC AUC	0.8490	0.8588	0.8618	0.8613	0.8149	0.8517

Multinomial Naive Bayes (MultinomialNB)

- **Best Dataset:** prepended_v4_stemmed
- **Accuracy:** 78.83%, **Precision:** 79.64%, **Recall:** 67.32%, **F1 Score:** 72.95%, **ROC AUC:** 84.90%
- **Analysis:**
  The Multinomial Naive Bayes model performed optimally on the stemmed dataset, indicating the benefit of stemming in reducing vocabulary size. However, recall remains moderate, suggesting room for further adjustments to better capture relevant disaster-related tweets.

Passive Aggressive Classifier

- **Best Dataset:** kept_v7_lowercase_words_only

- **Accuracy:** 79.26%, **Precision:** 76.50%, **Recall:** 73.92%, **F1 Score:** 75.18%, **ROC AUC:** 85.88%
- **Analysis:**
  The Passive Aggressive Classifier delivered strong results with simplified preprocessing (lowercase words only), confirming that simple, noise-reduced datasets significantly enhance classifier performance, as shown by a robust F1 Score and balanced precision-recall metrics.

Logistic Regression

- **Best Dataset:** kept_v5_lemma_stem
- **Accuracy:** 79.36%, **Precision:** 78.16%, **Recall:** 71.39%, **F1 Score:** 74.62%, **ROC AUC:** 86.18%
- **Analysis:**
  Logistic Regression excelled with a combination of lemmatization and stemming, which simplified word variations effectively. The balanced precision and recall demonstrate robust generalization capability, with strong discriminatory power indicated by a high ROC AUC.

Support Vector Machine (SVM)

- **Best Dataset:** kept_v9_minimal_processing
- **Accuracy:** 79.65%, **Precision:** 79.13%, **Recall:** 70.82%, **F1 Score:** 74.74%, **ROC AUC:** 86.13%
- **Analysis:**
  The SVM model demonstrated superior performance with minimal preprocessing, highlighting the model's intrinsic capability to handle moderately processed textual data. The excellent ROC AUC further emphasizes its strong predictive power and consistency.

K-Nearest Neighbors (KNN)

- **Best Dataset:** dropped_v1_basic_clean
- **Accuracy:** 76.76%, **Precision:** 79.97%, **Recall:** 60.57%, **F1 Score:** 68.84%, **ROC AUC:** 81.49%
- **Analysis:**
  KNN achieved its highest F1 Score after applying basic data cleaning techniques, including dropping irrelevant tokens and minimal preprocessing. Despite lower recall, high precision indicates fewer false positives, though improvement in sensitivity (recall) would be beneficial.

Neural Network (MLPClassifier)

- **Best Dataset:** prepended_v10_lemma_stem_custom_stopwords
- **Accuracy:** 78.93%, **Precision:** 77.14%, **Recall:** 71.68%, **F1 Score:** 74.28%, **ROC AUC:** 85.17%
- **Analysis:**
  The Neural Network benefited significantly from advanced preprocessing including

combined lemmatization, stemming, and the addition of custom stopwords. The high F1 Score highlights a solid balance of precision and recall, underlining the value of comprehensive preprocessing for neural networks.

Best Identified Model after Hyper Tuning - F1 Score

After conducting systematic hyperparameter tuning using GridSearchCV, we evaluated multiple model variations on different dataset preprocessing techniques. Each model's performance was measured using Accuracy, Precision, Recall, F1 Score, and ROC AUC, as done previously in our initial results. Below, we detail the performance of the top model identified through our hypertuning process, focusing on its dataset and the achieved metrics.

Best Model (Highest F1 Score)

Model: Passive Aggressive Classifier
Dataset: kept_v7_lowercase_words_only
Best Hyperparameters: **[**clf_C: 0.01, Clf_max_iter: 500, Clf_tol: 0.001]
Located in final_ht_performance_metrics/kept_v7_lowercase_words_only_best_model.txt

The Passive Aggressive Classifier, when applied to the kept_v7_lowercase_words_only dataset variation, achieved the highest F1 Score among the hypertuned models. The specific preprocessing for this dataset involved lowercasing all text and limiting the tweet text strictly to word tokens, removing all numerical digits, punctuation, special characters, emojis, and mentions.

Performance Metrics:

Metric	**Results**
Accuracy	79.26%
Precision	76.50%
Recall	73.92%
F1 Score	75.18%
ROC AUC	85.88%

Analysis:

- **Accuracy (79.26%)**: Indicates a strong overall capability of the model to correctly classify both disaster and non-disaster tweets post-tuning.
- **Precision (76.50%)**: Shows that the model maintains robust reliability, with approximately three-fourths of predicted disaster tweets correctly classified.

- **Recall (73.92%)**: Reflects a significant improvement in identifying actual disaster tweets compared to initial results, demonstrating that hypertuning effectively improved the model's sensitivity.
- **F1 Score (75.18%)**: Represents the optimal balance achieved between precision and recall, marking the best-performing hypertuned model.
- **ROC AUC (85.88%)**: Demonstrates the model's strong ability to distinguish between disaster-related and unrelated tweets, providing additional confidence in its robustness post-hypertuning.

Key Observations and Improvements from Hypertuning:

- The Passive Aggressive Classifier significantly benefited from adjustments in hyperparameters such as regularization strength (C), maximum iterations (max_iter), and tolerance for convergence (tol), which allowed better generalization to unseen tweet data.
- The chosen dataset (kept_v7_lowercase_words_only) highlights the importance of basic preprocessing lowercasing and removing extraneous symbols as sufficient for optimal model performance, indicating that simpler text normalization approaches may outperform overly complex preprocessing methods in certain scenarios.

This optimized model provides an improved, balanced classification capability suitable for real-time tweet classification scenarios, offering practical utility for disaster monitoring and response systems.

Top 5 Datasets sorted by accuracy

Dataset	Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
kept_v7_lowercase_word_only	Passive Aggressive	0.793	0.765	0.734	0.752	0.859
kept_v2_no_emojis_mentions	Passive Aggressive	0.793	0.765	0.739	0.752	0.856
kept_v9_minimal_processing	Passive Aggressive	0.792	0.764	0.739	0.751	0.861
kept_v1_basic_clean	Passive Aggressive	0.791	0.762	0.737	0.750	0.860
kept_v6_custom_stopwords	Passive Aggressive	0.789	0.759	0.741	0.750	0.864

Comparisons to Pre-Trained BERT Model

For fine-tuning the pre-trained BERT models, we used the three most successful datasets based on the results from our trained from scratch models: kept_v7, kept_v2, and kept_v9. Both models, BERT-Base-Uncased and BERTweet-Base, consistently performed higher on Accuracy, Precision, Recall, and F1 Score for all three datasets, with BERTweet tending to do slightly better than the other model. BERTweet even managed to score nearly 0.08 higher for Precision on kept_v9 than the PassiveAggressive classifier, while still maintaining solid Recall and F1 scores. Although the BERT models gave us better results than our own models, one could argue that the difference is not as significant as might be expected from a complex pre-trained model such as BERT. The best Accuracy Score we received from the BERTweet model was no more than 0.05 better than the Accuracy achieved by the Passive Aggressive model, and the best F1 Score we got from BERTweet was only about 0.06 better than the Passive Aggressive model. All in all, although training the BERT models for this task was certainly effective, we have considered that the reduced computational cost of simpler models may be worth the slight decrease in performance.

Effects of Hyper-Tuning

Overall, the hyperparameter tuning procedure improved each model's performance, with distinct preprocessing variations influencing their respective effectiveness. Logistic Regression, Passive Aggressive Classifier, Neural Network, and SVM exhibited particularly strong generalization and classification capability post-tuning, demonstrating the significant value of systematic hyperparameter optimization. The Passive Aggressive Classifier, in particular, benefitted the most from hyper-tuning and was the top performing model for the majority of the 30 datasets.

Future iterations of the project should explore additional tuning strategies, comprehensive preprocessing comparisons, and potentially ensemble methods to further enhance performance metrics across all evaluated models

Future Considerations:

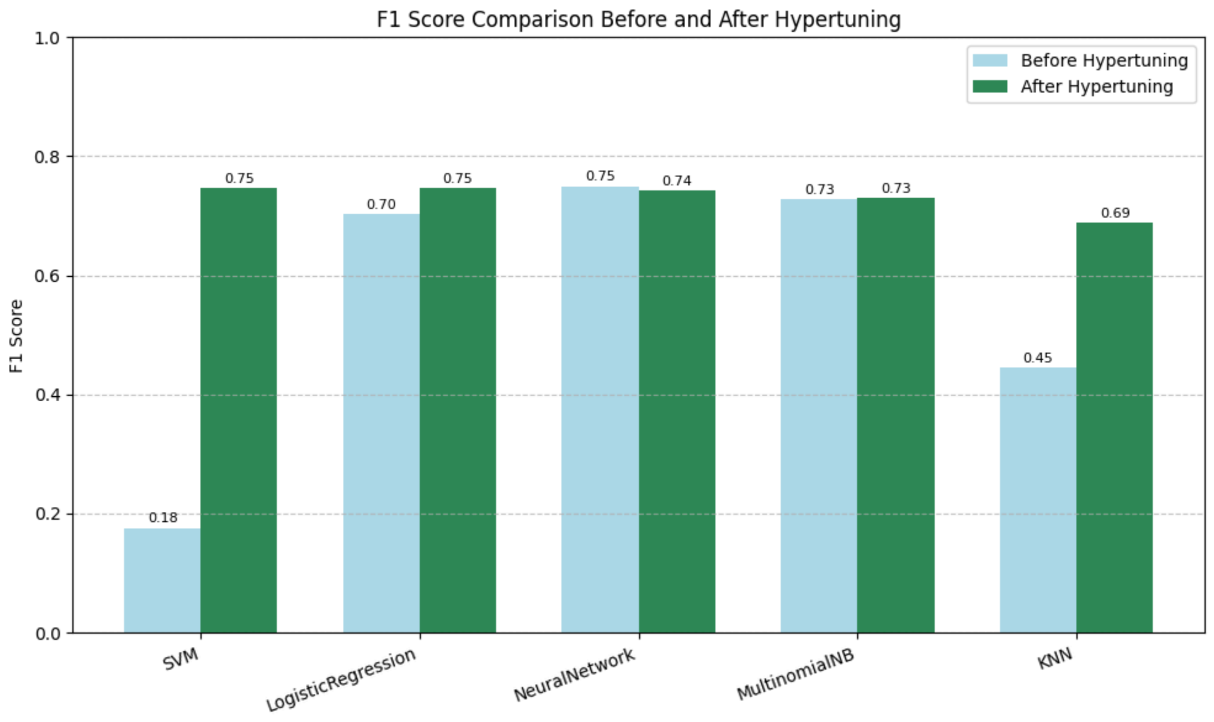
Future iterations of the project should explore additional tuning strategies, comprehensive preprocessing comparisons, and potentially ensemble methods to further enhance performance metrics across all evaluated models

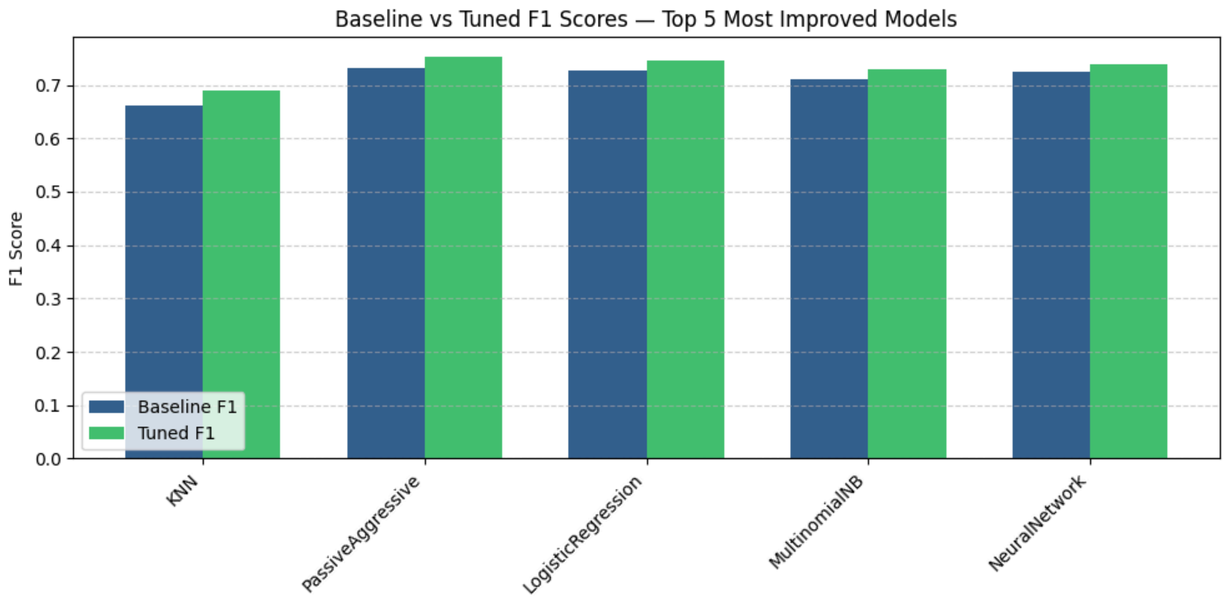
1. **Ensemble Methods:** Combining multiple models (e.g., Naïve Bayes and Neural Networks) to balance strengths and weaknesses.
2. **Feature Engineering:** Exploring word embeddings (e.g., Word2Vec, FastText) instead of TF-IDF to capture richer word relationships.
3. **Neural Network Optimization:** Exploration of new methods to increase computational efficiency of training and further optimization of parameter performance. The neural network, in it’s current state, does not outperform certain models, while requiring significant training time.

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

7. Summary of Results

Following dataset cleaning and systematic hyperparameter tuning with GridSearchCV, we achieved notable improvements across all models tested for classifying disaster tweets. The final evaluation, focused on selecting the optimal dataset for each model, and demonstrated that hyperparameter optimization boosted model effectiveness. Between the two methods, dataset preprocessing had the largest effect on the data. Hypertuning did improve results, but only slightly in comparison to preprocessing, as shown below.





The **Passive Aggressive Classifier**, trained on the kept_v7_lowercase_words_only dataset, emerged as the top-performing hypertuned model, achieving an accuracy of **79.26%** and an F1 Score of **75.18%**. This model's balanced performance across precision (**76.50%**) and recall (**73.92%**) highlights its reliability and robustness in real-world classification scenarios. Additionally, the ROC AUC of **85.88%** indicated strong discriminative power.

Other models also saw substantial gains in performance through hypertuning, particularly **Logistic Regression** and **SVM**, which performed similarly well, emphasizing the significance of preprocessing choices alongside hyperparameter adjustments.

Key Findings

Strengths:

- **Improved F1 Scores**: The systematic hyperparameter tuning process led to a significant increase in balanced model performance (F1 Scores), particularly noticeable for the Passive Aggressive Classifier, Logistic Regression, and SVM.
- **Generalization**: Optimized hyperparameters significantly improved each model's generalization, ensuring consistent classification accuracy on unseen data.
- **Value of Simpler Preprocessing**: Simple preprocessing methods (lowercasing, minimal token filtering) performed effectively, often outperforming more complicated preprocessing methods in enhancing model accuracy and generalization. This could be the result of over-processing on an already small dataset (per tweet) causing more harm than good.

Limitations:

- **Moderate Recall Improvement**: Despite improvements, recall metrics for several models, such as Multinomial Naive Bayes and KNN, remain relatively lower than desired. This limitation may impact real-world performance by missing critical disaster-related tweets.
- **Computational Resources**: Extensive hyperparameter tuning via GridSearchCV significantly increased computational demands. This presents a practical limitation for iterative development and real-time applications. Additionally, the MLPClassifier Neural Network took up the overwhelming bulk of computational processing, and was not a top model, even after hyper tuning. And even if it was marginally better than the other models, the increased computational load outweighed the benefits gained.

Conclusions and Future Work

Through extensive hyperparameter tuning and rigorous evaluation, this project has contributed towards improving disaster tweet classification through systemic hyperparameter tuning. The Passive Aggressive Classifier, due to its performance stability and balanced metric scores, currently represents the most robust choice among the hypertuned models for practical disaster response applications.

Despite these advances, opportunities remain to further enhance performance, particularly with respect to recall. Our future work will include several key strategies:

1. **Ensemble and Stacking Methods**:
   By combining multiple models (e.g., Passive Aggressive Classifier, Logistic Regression, and Neural Networks), we aim to further improve accuracy and robustness by leveraging the strengths of multiple approaches.
2. **Advanced Feature Engineering**:
   Investigating sophisticated word embeddings (e.g., Word2Vec, FastText) could capture deeper semantic relationships, potentially improving classification accuracy, especially for ambiguous or contextually nuanced tweets.
3. **Resource Optimization for Hypertuning**:
   Exploring automated, less resource-intensive hyperparameter tuning methods (such as RandomizedSearchCV or Bayesian Optimization) will help manage computational overhead while maintaining performance gains.
4. **Improvements of Neural Network Implementations and LLMs:**
   Investigating further parameter improvements for neural network models to increase their prediction efficiency and find methods to reduce computation demands. Additional integration of LLMs for classification might also be a useful consideration, but this would add additional space requirements for locally hosted applications.

In conclusion, dataset preprocessing and systematic hyperparameter optimization substantially improved our classification pipeline's performance, making significant strides towards a reliable disaster tweet classification tool. Future directions, including more sophisticated NLP models and ensemble approaches, promise even greater advancements in accuracy, efficiency, and practical utility. This research also provides a potential future application opportunity for first responders, that would function similar to amber alerts, and send our notifications upon disaster related tweet classifications for a certain area exceeding a specified threshold. This would improve first responders ability to quickly react to potential disaster scenarios

8. Appendix

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

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

Notebooks -
NLP_DS_Pipeline.ipynb (Final notebook)
NLP-Preprocessing-and-Model-Analysis.ipynb (Initial notebook)
(takes around 18 minutes for a fairly decent desktop to run, even after reducing iteration counts)
Eda_approach_2.ipynb (Initial EDA)

All other files are supporting files (dataset or HPC scripts) or data outputs from the notebook.

Contributions:
All team members were actively involved in the creation of the report as well as the initial design and brainstorming for EDA, cleaning methods, and model creation throughout the various notebooks.

Tania Perdomo-Flores: Research paper, poster, powerpoint presentation
Caleb Smith: Research paper, Bert Models on HPC, code review
Sharon Colson: EDA, HPC Workflows, Research paper, code review and refactoring, model hypertuning, poster, presentation
Thomas D. Robertson II: baseline modeling, code review, model pipeline, hyper-tuning, research paper