1. Problem Statement and Background

Written by: Sharon Colson, Thomas Robertson, Caleb Smith, Tania Flores

Statement of the Problem:
Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

However, identifying whether a tweet is genuinely about a disaster or simply using disaster-related language in another context is not always straightforward. Our project aims to develop a method for distinguishing real disaster-related tweets from those that are unrelated, ensuring more accurate and reliable disaster detection.

Dataset Overview:
The dataset used in this project is hosted on Kaggle and originates from the company figure-eight, originally hosted on their 'Data for Everyone' website, and then repurposed for use in a kaggle competition. This dataset is meant to be used in tandem with custom built machine learning models designed to predict Tweet status. We are predicting whether a given tweet pertains to a real disaster or not. In this dataset, a 1 indicates a member of the positive class - a disaster - and a 0 indicates the negative class - no disaster.

Dataset Files:

train.csv
test.csv
sample_submission.csv

In the above files, train.csv contains the data used to train the custom built machine learning models. In particular it contains the "target" feature, which indicates whether a tweet was a disaster or not. The test.csv data contains the same overall information, except the 'target' feature has been removed. The train and test datasets, while sharing an id feature, have exclusive id data, as such there is no overlapping information between the datasets. Sample_submission is, as the name implies, a sample of what a a properly submitted predicted dataset would look like for competition scoring.

Dataset Variables:

- id - a unique identifier for each tweet
- text - the text of the tweet
- location - the location the tweet was sent from (user input, may be blank)
- keyword - a particular keyword from the tweet (may be blank)
- 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, but has been provided in **test.csv** for the purpose of calculating the final testing metrics (F1, accuracy, precision, recall).

Success Measures:
Our evaluation metrics include accuracy, precision, recall, F1 score, and AUC/ROC. An F1 score of 0.75 or higher is our target for the final tuned model, with visualizations (confusion matrices, ROC curves) used to assess the balance between precision and recall.

Background and Importance:
As emergency events unfold, timely and accurate detection of disaster-related information can save lives. Social media, and Twitter in particular, provides a unique window into real-time events. However, the informal language and noisy nature of tweets require specialized natural language processing (NLP) techniques to extract meaningful signals. Our work seeks not only to improve disaster detection accuracy but also to streamline data processing for faster, more effective decision making in crisis situations.

Related Work:
Numerous studies have applied NLP and machine learning to tweet classification. Prior research has explored techniques ranging from traditional bag-of-words models to advanced transformer-based approaches. By combining traditional text preprocessing with state-of-the-art vectorization and classification methods, our project builds on this foundation to address the unique challenges posed by disaster-related tweets.

Goal:
By developing a model that accurately distinguishes disaster-related tweets, this project contributes to the field of NLP and provides a tool for emergency management agencies to rapidly filter and respond to critical information.

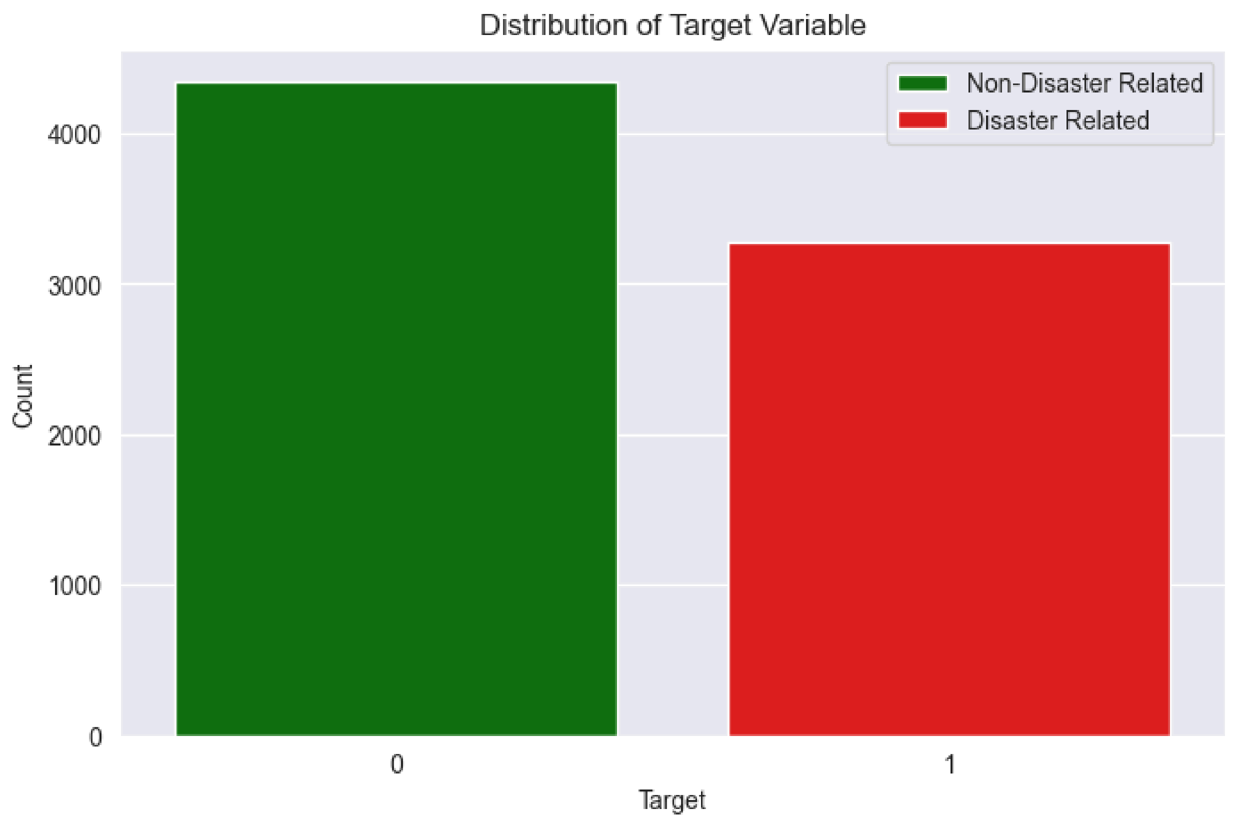
2. Data and Exploratory Analysis

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

The Tweets Dataset is hosted on Kaggle as a Prediction Competition. The dataset is …. There are several independent variables and one target outcome (dependent) variable, whether the Tweet posted is about a natural disaster or not.

Dataset Variables:
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, but has been provided in **test.csv** for the purpose of calculating the final testing metrics (F1, accuracy, precision, recall).

The dataset has a relatively even distribution of the target variable, with a slight imbalance towards the negative class at approximately 57% of the target share. This is important to note as it may affect the performance of our models, but with this rate distribution balancing should not be necessary.. General data cleaning included outlier removal, which in this case is repeated sentences. Sentences with mismatched keyword values will also be excluded from filtered models to avoid any bias or data skewing.

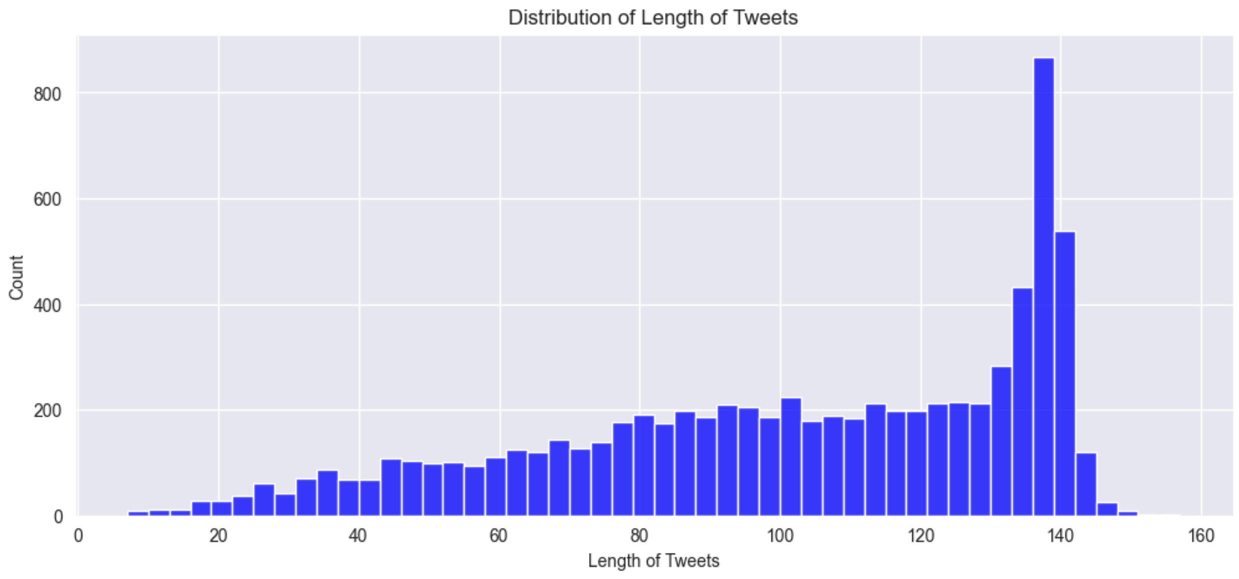


We continued analysis of tweets by examining the various statistics involving the text features in the training dataset. We started off by checking the statistics of all the tweets, and then the statistics of the individual classes. The goal was to find any patterns or trends in the tweets related to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

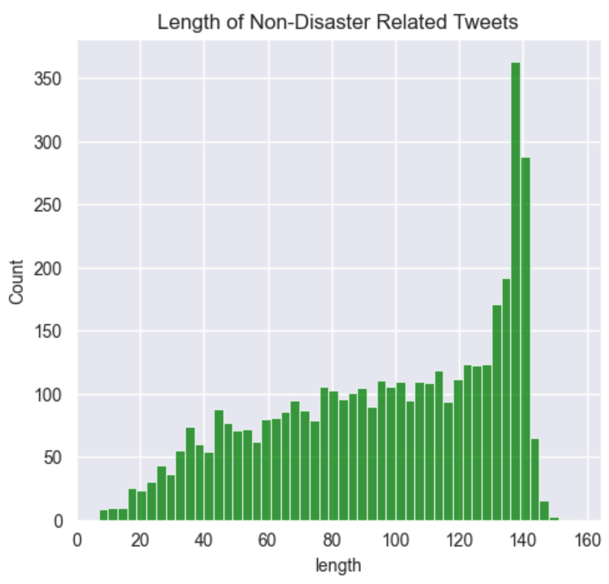
Feature	**Value**
Count	7613.0
Mean	101.0
Std	33.8
Min	7.0
25%	78.0
50%	107.0

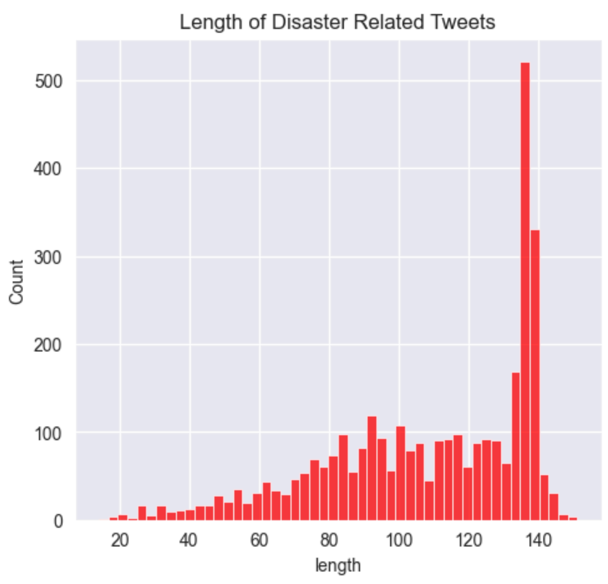
75%	133.0
max	157.0

From the table we can see that across all tweets in the dataset, there is an average length of 101 words, with a deviation of 33.8 words. The smallest tweet was 7 words, and the largest tweet was 157 words. A noticeable characteristic of the longest tweets is that they contained many repeating, and generally irrelevant, characters. Examples of this include repeating !, ?, etc.



We can see that the distribution of the length of tweets generally lies in the 70-125 range, with a few outliers on either end. The distribution is slightly skewed to the right, indicating that there are more tweets with shorter lengths than longer ones. We then performed further analysis by checking the distribution of the length of tweets for both targets to see if there are any differences between the two classes.





We can see a significantly higher distribution of medium length tweets in the negative class, while positive class tweets are generally higher length tweets. This is likely due to the nature of disaster related tweets, which are often more detailed and descriptive than normal tweets as they are attempting to give out detailed information regarding the disaster, its location, causes, etc.After finding these trends, we began cleaning the data to prepare it for modeling.

Data Cleaning and Preparation:

The data preparation involved these important steps:

- Removing duplicates
  - Bias and outlier correction
- Removing punctuations, stopwords, and special characters.
  - Noise reduction
- Removing numerical content from the data.
  - Numerical content doesn't apply to contextual processing.
- Converting all text to lowercase.
  - Ensures uniformity
- Removing URLs and links
  - Ensures that the model focuses on the text content.
- Removing emojis and other non-textual elements.
  - Disaster tweets generally do not contain emojis.

Outlier Detection and Missing values:

The data being textual based allowed for easy removal of duplicated tweets to aid in skew removal. Missing values (keyword, location) were not used as part of the model training or testing process. As such, these features could be ignored.

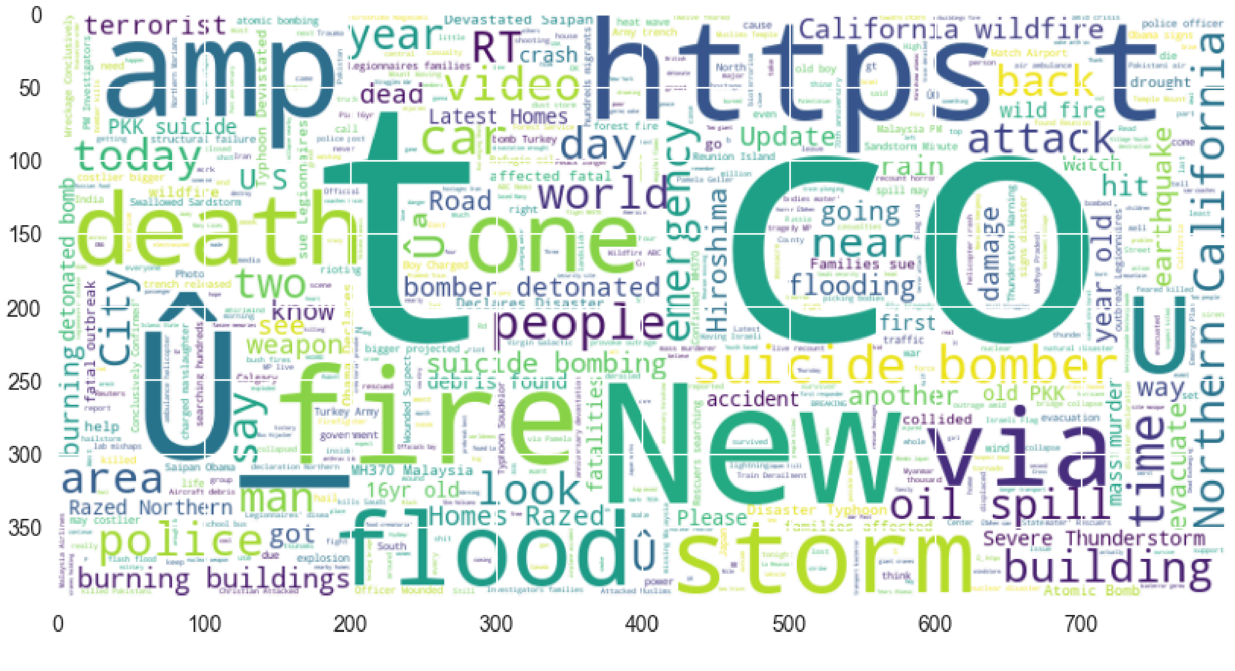
Overview:

The cleaning process involved removing duplicates, bias/noise, punctuations, stopwords, special characters, numerical content, URLs and links, emojis, and text conversion. Missing variables in the keyword and location feature were not relevant to our model testing, so they were ignored. Further analysis where they would need to be addressed could include keyword to text contextual matching, which might be carried out in further iterations. The cleaned dataset was then used to build predictive models and assess their accuracy. Further evaluation will focus on comparing the performance of models built with different analytical methods (e.g transformer based contextual analysis) to determine the best approach for this specific dataset.

Data Visualization

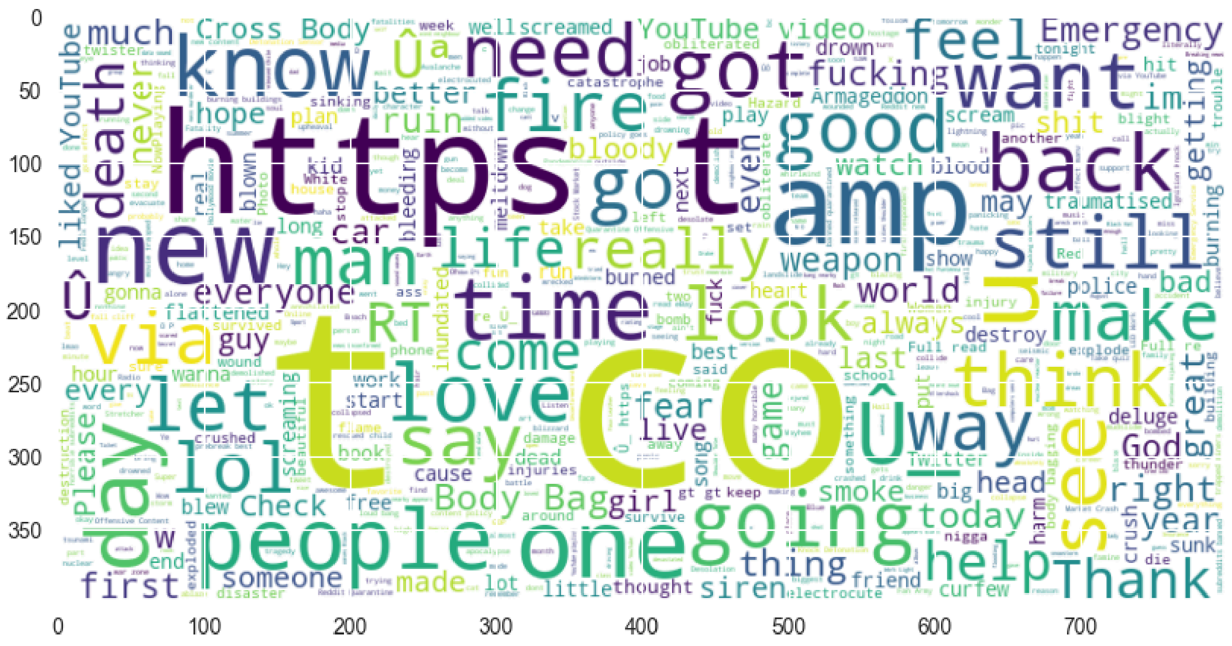
After initial data cleaning, a word cloud was generated to form a visual representation of the most common words used in the dataset, with the size of each word representing its frequency. This allowed for visual identification of patterns and trends in the data as well as giving a general sense of word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

Disaster Word Cloud:



We can see in this disaster word cloud that the most common words used in disaster tweets are "fire", "flood", "storm", "suicide bomber", "people", "police", "oil spill", "burning buildings", "California wildfire" and "Northern California". From a first glance at this word cloud, initial assumptions would suggest many of the disaster related tweets are specifically referring to wildfire outbreaks in California.

Non-Disaster Word Cloud



From the non-disaster related tweets we can see thematic patterns such as "love", "people", "help", "Thank", "good", "feel" and "know". We also see some words similar to the disaster cloud such as "fire", and "Emergency". These words are probably being used as general conversation modifiers.

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. Porter Stemmanization was carried out on the tweets to account for any regional spelling differences.

Feature Engineering

After cleaning the data, the creation of a new features (x) and target variables (y) was performed for model analysis. The cleaned tweets as our feature x, feature ‘target’ served as our target variable. The data was split into training and testing sets with the traditional 80:20 split to evaluate the performance of our models.

Bi-Grams and Tri-Grams

While Tweets are oftentimes shorter form and usually don’t have too many chained word groupings, TF-IDF Vectorizer was utilized to identify two to three word pattern trends in the data. Additionally, the vectorizers string to numerical information conversion allows for more efficient model processing results, which can be crucial for large textual based datasets. In the data we ignored any word chains that occur in more than 80% of tweets within the dataset, as these words are likely to be stop words and would not hold any value to the analysis.

Model Analysis

For our initial model, we utilized a Multinomial Naive Bayes model to serve as our baseline for comparison on future models. Our vectorized training data was passed to the model, fitted, and then a 10 Fold Cross Validation Test was performed on the model to test our current models efficiency.

ADD TEST DATA RESULTS HERE, I'LL GET TO IT LATER PROBABLY

In the future we plan on testing similar model analysis results on Logistic Regression based models, Support Vector Machines, and Transformer based models. We will use model specific hyper-tuning for each case, and then compare the results to find the best model fit for Natural Language Processing on our specific dataset.

3. Methods

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

Data Cleaning and Preprocessing Methods:
Our cleaning process was designed to address both the unique challenges of tweet text and the general need for consistency across datasets. Two primary strategies were applied:

A. Text Cleaning and Normalization

- **Method:**
  - Remove duplicate tweets
  - Strip URLs, mentions, hashtags (unless relevant for context), and special characters
  - Convert all text to lowercase for uniformity
  - Remove numbers and extraneous punctuation
  - Apply custom stopword filtering to eliminate common noise terms (e.g., "https", "rt")
- **Justification:**
  Tweets often contain non-standard language, and eliminating noise improves the signal for NLP models. Lowercasing and stopword removal ensure that models focus on meaningful content.
- **Evaluation:**
  Visual inspections (e.g., word clouds before and after cleaning) demonstrated improved clarity in the vocabulary distribution. Reduced noise levels were confirmed by increased coherence in tokenized outputs.

B. Data Preprocessing Tools and Python Code

- **Tokenization and Stemming:**
  - **Tools:** Natural Language Toolkit (NLTK) and SpaCy
  - **Method:** Use NLTK's tokenizer and Porter Stemmer, combined with SpaCy's advanced tokenization to handle informal text patterns.

    - **Justification:** Provides a balance between rule-based tokenization and contextual understanding, ensuring that slang and abbreviations are processed appropriately.
    - **Evaluation:** Improved token consistency and reduced vocabulary size led to more robust feature extraction.

C. Data Cleaning Methods

- **Method Types:**
    - **Rule-Based Filtering:** Removal of duplicates, non-alphabetical characters, and non-essential tokens.
    - **Contextual Tokenization:** Leveraging SpaCy's pipeline to capture dependencies in the informal tweet text.
- **Justifications and Evaluations:**
  The rule-based filtering was crucial for rapid noise reduction, while the contextual tokenization provided enhanced understanding of tweet semantics. Evaluation metrics (e.g., model performance improvements after cleaning) confirmed that these combined approaches resulted in clearer class separations and better downstream model accuracy.

Explored Algorithms for Classification:

We considered a range of machine learning models to address the binary classification problem on predicting tweet class status. The models were chosen based on their suitability for handling binary classification tasks and their ability to manage complexities of the datasets' contextual nature.

- **Multinomial Naive Bayes:**
  A baseline classifier leveraging the frequency distribution of tokens.

Models under consideration:

The following models are planned to be utilized in the future, they have not been implemented yet due to time constraints or other factors.

- **Logistic Regression:**
  Utilized on vectorized data to provide interpretable probability outputs.
- **Support Vector Machine (SVM):**
  Testing for its ability to capture non-linear decision boundaries in the embedding space.
- **Transformer-Based Models:**
  Explored for fine-tuning on the task to leverage contextual embeddings directly.

4. Tools Used for Data Cleaning, Visualization, and Model Evaluation

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

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 evaluation. Below is a breakdown of the tools used, how they were applied, and the reasons for their selection:

The following tools and Python libraries were either used, or we plan to use, throughout the project for data cleaning, visualization, and model evaluation:

- **Pandas:**
  For data manipulation, cleaning, and initial exploratory analysis.
- **NLTK and SpaCy:**
  For text tokenization, stemming, and part-of-speech tagging. Their complementary strengths helped balance rule-based and context-aware text processing.
- **Sentence-Transformers (HuggingFace):**
  To convert text into dense embeddings, capturing semantic similarities that are critical for nuanced tweet classification.
- **Scikit-learn:**
  Provids implementations of Multinomial Naive Bayes, Logistic Regression, and SVM classifiers, along with utility functions for cross-validation and performance metrics.
- **Matplotlib and Seaborn:**
  For visualizing distributions, word clouds, and model performance metrics (e.g., ROC curves).
- **PyLDAVIS (not yet implemented)**: Provides interactive topic highlighting visualization.

Overview:

The combination of tools we used provided for a robust workflow for data cleaning, visualization, model training, and evaluation. Pandas facilitated crucial data cleaning steps, while Matplotlib and Seaborn provided valuable insight through visualization outputs. Scikit-learn enables thorough model training and evaluation. Further improvements could include exploring additional hyper-tuning optimization techniques for our models. Implementation of pipeline procedures to enable multiple model analysis. And considering the use of more interactive visualization tools, and revisiting unused models to expand out analysis such as hugging face transformers

5. Initial Results

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

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

- **Accuracy:** Overall percentage of correct classifications.
- **Precision:** Proportion of predicted disaster tweets that were correct.
- **Recall (Sensitivity):** Ability of the model to correctly identify disaster tweets.
- **F1 Score:** Harmonic mean of precision and recall, emphasizing balance between the two.

- **AUC/ROC (not yet implemented):** Visual and quantitative assessment of the classifier's ability to distinguish between classes.

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

Metrics	Multinomial Naive Bayes	Logistic Regression	Support-Vector Machine	Transformer-Models
Accuracy	0.7939	Not yet implemented	Not yet implemented	Not yet implemented
Sensitivity (Recall)	0.6037	Not yet implemented	Not yet implemented	Not yet implemented
Specificity	0.8805	Not yet implemented	Not yet implemented	Not yet implemented
F1 Score	0.7159	Not yet implemented	Not yet implemented	Not yet implemented

Detailed Model Analysis:
These results indicate that the classifier is performing reasonably well, achieving an average F1 score of approximately 71.6%. Notably, the model demonstrates high precision—suggesting that it makes few false-positive errors. However, the relatively lower recall indicates that a significant portion of positive cases (i.e., genuine disaster-related tweets) may be missed. This trade-off between precision and recall is a key insight from our current evaluation.

Conclusions and Recommendations (not yet implemented)

- Best Model for Balance (F1 score)
- Best Model for Sensitivity
- Best model Specificity
- Considerations

6. Summary and Conclusions for Initial Stages

Summary of Results:
Our initial experiment focused solely on the Multinomial Naive Bayes classifier. The model achieved a mean accuracy of 79.4%, with a particularly high precision (88.1%) and an F1 score of 71.6%. While these results are promising, the model's lower recall of 60.4% suggests that it may not be capturing all disaster-related tweets effectively.

Key Findings:

- **Strengths:**
  - The classifier is highly precise, which means that when it predicts a tweet is disaster-related, it is very likely to be correct.
- **Limitations:**
  - The relatively lower recall indicates that the model is missing a significant number of positive instances, which is a critical shortcoming for real-world disaster detection where missing a true disaster signal could have serious consequences.

Conclusion and Future Work:
The current Multinomial Naive Bayes implementation provides a solid baseline for disaster tweet classification. Its high precision is encouraging; however, the lower recall underscores the need for further refinement. Moving forward, we plan to implement and compare additional models—including logistic regression, support vector machines, and transformer-based approaches—to explore strategies that could improve recall while maintaining overall model performance. These future enhancements will be critical to developing a more balanced and robust system for accurately detecting disaster-related tweets.

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