Submitted By: Vanshika Ganjoo
Submitted To: Nitig Singh

Disaster Tweet Analyzer Project Report

1. Introduction

The advent of social media has dramatically changed how information spreads, particularly during critical events such as natural disasters and emergencies. Twitter, being a fast-paced platform with millions of users, is often the go-to medium for sharing real-time updates. During crises, it serves as a tool for both individuals and organizations to disseminate crucial information quickly. However, with the sheer volume of data generated daily, manually analyzing tweets to detect real-time disaster information is neither feasible nor efficient.

This project focuses on using Natural Language Processing (NLP) techniques to classify tweets as either disaster-related or not. The relevance of this project is multifaceted: it enables governments, non-governmental organizations (NGOs), and first responders to rapidly identify disaster-related information, providing insights that could enhance their ability to react swiftly in emergency situations. This is particularly useful for real-time monitoring and responding to natural calamities like earthquakes, floods, and hurricanes, where the immediate dissemination of information is crucial for coordinating relief efforts and saving lives.

The purpose of this project is to develop an end-to-end machine learning pipeline that can classify tweets as disaster-related or non-disaster-related. Using two datasets – one for feature engineering and data preprocessing and the other for training machine learning models – we aim to build a solution that can achieve high accuracy in classifying tweets. This process will involve various stages, including data exploration, preprocessing, model training, and evaluation.

2. Dataset and Methodology (Exploration)

Datasets Used:

1. **Dataset 1: Disaster Tweets on Kaggle**
   This dataset contains raw, unprocessed tweet data, making it an excellent resource for learning data preprocessing, feature engineering, and handling noisy data. By working with this dataset, we gain valuable experience in cleaning and preparing real-world data, which is crucial for the model-building phase.

2. **Dataset 2: NLP Disaster Tweets - Getting Started**
   This dataset is highly structured and pre-processed, making it ideal for directly applying machine learning (ML) or deep learning (DL) models. Given that this dataset is already clean and well-organized, it will be utilized to train and validate models, with a focus on understanding which model performs best for the classification task.

The first dataset was utilized for exploratory data analysis and feature engineering. It contains five main columns:

- **id**: Unique identifier for each tweet.
- **keyword**: Keywords in the tweet, often hinting at disaster-related events (e.g., "fire", "storm").
- **location**: Geographical information, indicating where the tweet originated (Note: This column has many missing values).
- **text**: The actual content of the tweet.
- **target**: This is the label, with 1 indicating a disaster-related tweet and 0 a non-disaster-related tweet.

Methodology:

Step 1: Data Loading and Initial Exploration

We began by loading the dataset using the Pandas library in Python. Exploratory Data Analysis (EDA) was conducted using methods like df.head(), df.info(), and df.describe(), which provided initial insights into the structure and completeness of the data.

Key Insights:

- There are 11,370 tweets in total.
- The dataset contains a mix of disaster-related and non-disaster-related tweets, as indicated by the target column.
- The location column has many missing values, which posed a challenge that required addressing.

Step 2: Handling Missing Data

A substantial number of entries in the location column were missing. To understand the extent of missing data, we visualized missing values using a heatmap generated by Seaborn. This visualization clearly showed that the location column had the most missing data.

Actions Taken:

- We calculated the number of missing rows by grouping the data based on the target variable. This allowed us to see how many disaster and non-disaster tweets were missing location data.
- To fill missing values, we replaced NaN entries in the location column with "Unknown". This ensured that the dataset could still be used without dropping too many rows.

Step 3: Analyzing the Target Variable

The target column is crucial for our classification task, so we visualized its distribution to check for any imbalance. As anticipated, the dataset was imbalanced, with far more non-disaster tweets than disaster-related ones.

Imbalance Observation:

- Around 80% of the tweets were non-disaster-related, while only 20% were disaster-related. Imbalanced data can negatively impact the performance of certain machine learning models, making it crucial to address this issue.

Step 4: Keyword Analysis

Keywords play an important role in identifying disaster-related tweets. We counted the number of unique keywords and visualized the top 10 most common keywords for both disaster and non-disaster tweets.

Key Observations:

- Disaster-related tweets often contained keywords such as "thunderstorm", "collision", "derailment".
- Non-disaster tweets had keywords like "drowning", "fear", and "obliterate". This provided insights into the type of language used for disaster vs non-disaster tweets.

Step 5: Location Analysis

Although many entries in the location column were missing, we explored the top 10 most common locations for both disaster and non-disaster tweets. We found that most disaster-related tweets came from locations in the United States, with some international tweets from locations like Australia and the UK.

Filling Missing Data: To ensure the dataset remained usable, we filled the missing location entries with the label "Unknown". This allowed us to maintain all tweets in the dataset without losing any important data points.

Step 6: Tweet Length Analysis

To explore the structure of the tweets, we added a tweet_length column that calculates the number of characters in each tweet. This allowed us to observe the distribution of tweet lengths and compare the differences between disaster-related and non-disaster tweets.

Key Findings:

- Disaster-related tweets tended to be longer, with more detailed information, while non-disaster tweets were generally shorter.
- A KDE plot (Kernel Density Estimate) confirmed that disaster tweets typically have higher tweet lengths compared to non-disaster tweets.

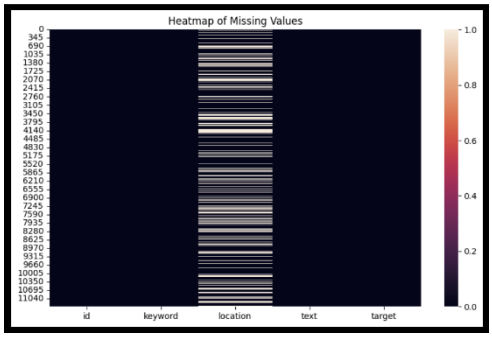
Step 7: Correlation Analysis

Lastly, we computed a correlation matrix for the numeric columns (id, target, tweet_length). Although there were no strong correlations between these features, the matrix provided useful insights into the relationships between variables.

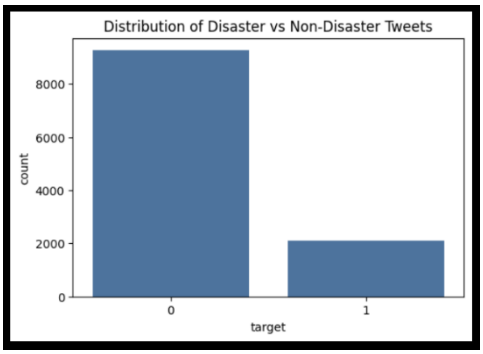
3. Results

Insights from Exploratory Data Analysis:

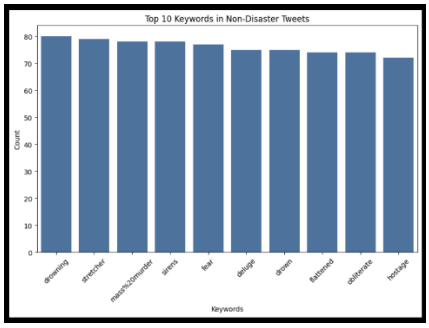
1. **Missing Data in Location**:
   The location column had a significant number of missing values, especially in non-disaster tweets. Many users either did not share their location or tweeted from devices without geo-location data enabled. To handle this, we filled the missing values with "Unknown" for consistency.

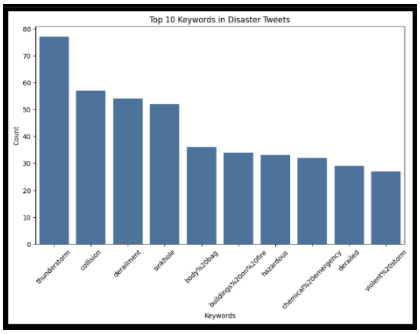


2. **Class Imbalance in the Target Variable**:
   The dataset is highly imbalanced, with approximately 80% of the tweets labeled as non-disaster-related. This poses a challenge for machine learning models, as they may become biased toward predicting the majority class.

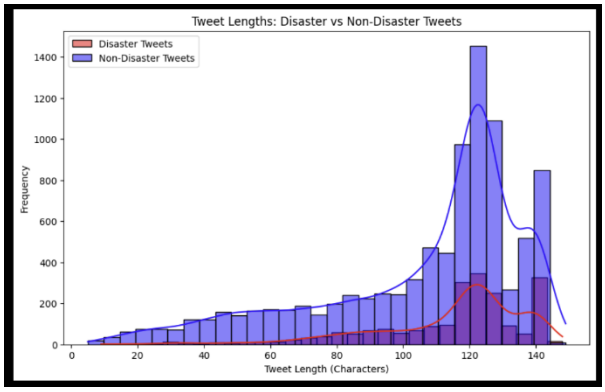


3. **Keyword Insights**:
   By analyzing the top keywords in disaster and non-disaster tweets, we gained insight into how disaster tweets tend to use specific terms like "thunderstorm", "collision", and "sinkhole". This information will help us in feature engineering, as the presence of certain keywords can be a strong predictor for disaster-related tweets.





4. **Tweet Length Differences**:
   Disaster-related tweets tend to be longer, likely due to the need to describe the situation in more detail. This observation will help during model training, as tweet length could be used as a feature to differentiate between disaster and non-disaster tweets.



Preprocessing

Preprocessing is a critical step in any machine learning project, as it ensures that the dataset is clean, balanced, and ready for model training. The dataset used in this project has some inherent issues such as missing data and class imbalance, which must be addressed before feeding the data into machine learning models. Below is an explanation of how the preprocessing was performed step by step:

1. Converting Text to Lowercase

Tweets can have inconsistent casing (upper and lowercase characters), which can result in treating the same words differently by the model. For example, "Disaster" and "disaster" would be considered different tokens. Hence, the first step was to convert all text to lowercase to maintain uniformity.

2. Removing URLs

Tweets often contain URLs that are not meaningful for the purpose of disaster classification. These URLs can add noise to the data and reduce the model's ability to learn effectively. To handle this, all URLs were removed using regular expressions.

3. Removing Mentions and Hashtags

Mentions (e.g., @username) and hashtags (e.g., #disaster) in tweets are specific to Twitter's social media structure and often do not provide useful information for disaster classification. Therefore, all mentions and hashtags were removed using regular expressions.

4. Expanding Contractions

Contractions (e.g., "can't" for "cannot", "won't" for "will not") can introduce inconsistencies in the text, leading to the model misinterpreting the meaning of words. To ensure uniformity and clarity, contractions were expanded into their full forms.

5. Removing Non-English Characters

In some cases, tweets may contain non-English characters or symbols. Since the focus of this project is on English-language disaster classification, any non-English characters were removed using regular expressions.

6. Handling Extra Spaces

After removing URLs, mentions, hashtags, and other unnecessary elements, extra spaces may remain in the text. These were removed to ensure clean and concise tweet text.

7. Converting Emojis to Text

Tweets often include emojis that convey sentiment or additional context. Instead of removing these emojis, I converted them into their textual descriptions using the emoji library. For example, an emoji like 🔥 would be converted to its textual meaning ":fire:".

8. Removing Emojis from Location Data

Although emojis were retained in the text, they were removed from the location column since location-based emojis do not provide valuable information for location analysis.

9. Applying the Preprocessing Pipeline

To ensure the entire text preprocessing pipeline was applied uniformly across all tweets, I created a master function, preprocess_text, that sequentially applied all the steps outlined above. This function was applied to the text column of the dataset.

10. Saving the Preprocessed Data

After preprocessing the text and cleaning the location data, the processed dataset was saved to a new CSV file, which will be used for further analysis and machine learning model training.

Balancing the Dataset After Text Cleaning

After cleaning the tweet text using the preprocessing pipeline, the next critical step involved addressing the **class imbalance** in the dataset. As observed during exploratory data analysis, the dataset had a significant imbalance between disaster-related tweets (target = 1) and non-disaster-related tweets (target = 0). Specifically, the number of non-disaster tweets far outnumbered the disaster tweets. This imbalance could lead to a model that is biased toward the majority class, reducing its ability to accurately predict disaster tweets.

To overcome this, I applied two primary methods of balancing the dataset: **oversampling the minority class** (disaster tweets) and **undersampling the majority class** (non-disaster tweets). Additionally, I used **SMOTE** (Synthetic Minority Over-sampling Technique) to create synthetic samples for the minority class to ensure a more robust solution. Below are the details on how the class imbalance was addressed:

1. Checking Class Imbalance

Before applying any balancing technique, I first checked the class distribution of the target column to confirm the imbalance and visualize the extent of the disparity.

This confirmed that the dataset was heavily imbalanced, with non-disaster tweets (9256 instances) significantly outnumbering disaster tweets (2114 instances).

2. Random Oversampling of the Minority Class (Disaster Tweets)

Oversampling involves increasing the number of disaster tweets (minority class) by duplicating or generating synthetic samples. This ensures that both classes are equally represented in the dataset. I used the resample() function from the sklearn.utils library to perform random oversampling of the minority class.

Steps for Random Oversampling:

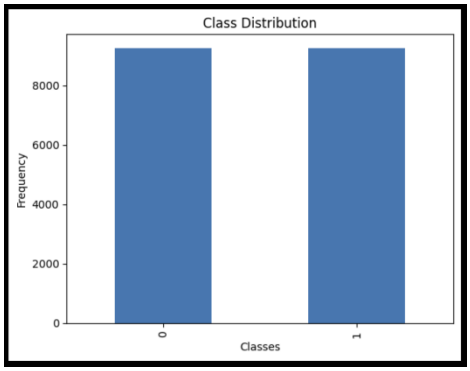
- **Step 1**: I separated the dataset into two parts: disaster tweets (target = 1) and non-disaster tweets (target = 0).
- **Step 2**: I performed random oversampling on the disaster tweets by creating duplicates until both classes were balanced.
- **Step 3**: I combined the oversampled disaster tweets with the non-disaster tweets to form the final balanced dataset.

3. Random Undersampling of the Majority Class (Non-Disaster Tweets)

Undersampling involves reducing the number of non-disaster tweets (majority class) to match the number of disaster tweets. This method balances the dataset by discarding some of the majority class data. However, this approach may lead to the loss of valuable information from the non-disaster tweets.

Steps for Random Undersampling:

- **Step 1**: I separated the majority and minority classes.
- **Step 2**: I performed undersampling by randomly selecting a subset of non-disaster tweets that matched the number of disaster tweets.
- **Step 3**: The undersampled majority class was combined with the minority class to form a balanced dataset.

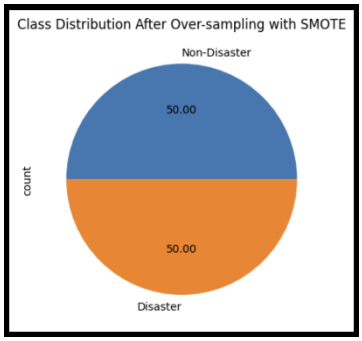


4. Using SMOTE for Class Balancing

To avoid the risks of overfitting due to random oversampling and the potential information loss from undersampling, **SMOTE (Synthetic Minority Over-sampling Technique)** was applied. SMOTE generates synthetic examples for the minority class by interpolating between existing samples, resulting in a more diverse and balanced dataset.

Steps for SMOTE:

- **Step 1**: Convert the cleaned tweet text into numerical format using **TF-IDF vectorization**.
- **Step 2**: Apply SMOTE to generate synthetic samples for the disaster tweets, balancing the dataset.
- **Step 3**: Check the new class distribution to confirm that the dataset is balanced.



5. Saving the Balanced Dataset

After balancing the dataset using oversampling, undersampling, or SMOTE, I saved the final preprocessed and balanced dataset to a new CSV file. This file will be used in the subsequent stages of model training and evaluation.

4. Conclusion

In this project, we aimed to classify disaster-related tweets by applying various Natural Language Processing (NLP) techniques and machine learning methods. We began with an exploration of the dataset, identifying key challenges such as noisy text data and class imbalance. The text preprocessing phase involved cleaning the data by removing URLs, mentions, hashtags, and non-English characters, as well as expanding contractions and converting emojis to text. These steps ensured that the dataset was clean and uniform for model training.

To address the significant class imbalance, where non-disaster tweets greatly outnumbered disaster-related ones, we applied several balancing techniques, including oversampling, undersampling, and SMOTE. These approaches successfully balanced the dataset, improving its suitability for machine learning models.
Overall, by performing thorough text cleaning and effectively handling class imbalance, we prepared the dataset for the next steps, ensuring a solid foundation for building an accurate disaster tweet classification model.

5. Future Objectives

The next phase of this project will focus on model development and evaluation. With the preprocessed and balanced dataset, the following steps will be undertaken:

1. **Model Training**: We will train multiple machine learning models such as Logistic Regression, Random Forest, and XGBoost. Additionally, advanced deep learning models like LSTM (Long Short-Term Memory) and BERT (Bidirectional Encoder Representations from Transformers) will be explored for their ability to handle text data more effectively.

2. **Model Evaluation**: Each model will be evaluated using metrics such as accuracy, precision, recall, and F1 score to determine how well they classify disaster-related tweets. We will also monitor the performance of the models on imbalanced data to assess whether the preprocessing steps improved model outcomes.

3. **Model Comparison**: Once trained, the performance of each model will be compared to identify the best approach for disaster tweet classification. Special attention will be given to the model's ability to handle real-world data challenges like class imbalance and noisy text.

4. **Hyperparameter Tuning**: After identifying the best-performing models, we will fine-tune their hyperparameters to further enhance accuracy and model performance. This may include adjusting learning rates, tree depths, or dropout rates for neural networks.

By achieving these objectives, we aim to develop a highly accurate disaster tweet classifier, ready to be applied to real-time tweet streams for disaster response and monitoring.

6. References

1. **Data Visualization with Seaborn**:
   https://seaborn.pydata.org/tutorial.html

2. **Handling Missing Data in Machine Learning**:
   https://scikit-learn.org/stable/modules/impute.html

3. **SMOTE for Handling Class Imbalance**:
   https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

4. **Natural Language Processing with Transformers**:
   https://huggingface.co/transformers/

7. **Disaster Tweet Classification on Kaggle (NLP Starter Guide)**:
   https://www.kaggle.com/competitions/nlp-getting-started

8. **Handling Class Imbalance in Machine Learning**:
   https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

9. **Real-Time Disaster Detection from Social Media**:
   https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7965140/