NLP Project for Disaster Tweet Classification

BY:-PRIYANSHU KUMAR

DATE:- JULY 2025

Agenda

- PROJECT OVERVIEW
- DATA ANALYSIS
- EDA

I. Disaster Non- Disaster Tweet
II. Characters used in disaster and non-disaster tweet
III. Pair Plot
IV. Correlation
V. Disaster Word Cloud
VI. 30 Most common words used in Non-Disaster Tweet
VII. 30 Most common words used in Disaster Tweet

- DEEP LEARNING
- APP.PY , HTML , Dashboard

Project Overview

In today's digital age, social media platforms like Twitter play a crucial role in disseminating information, especially during emergencies and disasters. However, distinguishing between tweets that genuinely indicate a disaster and those that do not can be challenging, even for humans. As a result, there is growing interest from various organizations, such as disaster relief agencies and news outlets, in developing automated methods to monitor Twitter for real-time disaster alerts.

Data Analysis

Datasets

- Historical store sales
- Promotions
- Holidays
- Assortment
- Competition.

Key Field

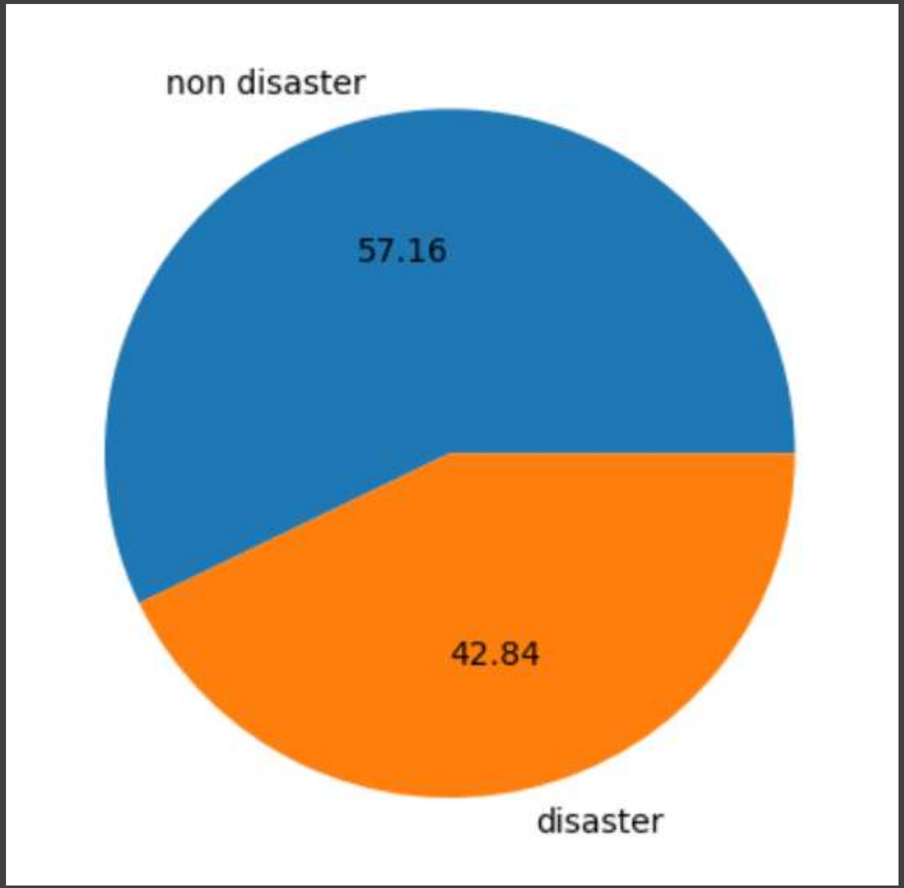
- TOTAL RECORD= 7613
- COLUMNS = 5
- MISSING VALUES :-

  KEYWORDS = 0.8% MISSING

  LOCATION = 33% MISSING

Disaster Non-Disaster Tweet

This Chart Shows that 57.16 % of the total tweet are non-disaster tweet whereas 42.84 % Of the total tweet are non- disaster tweet

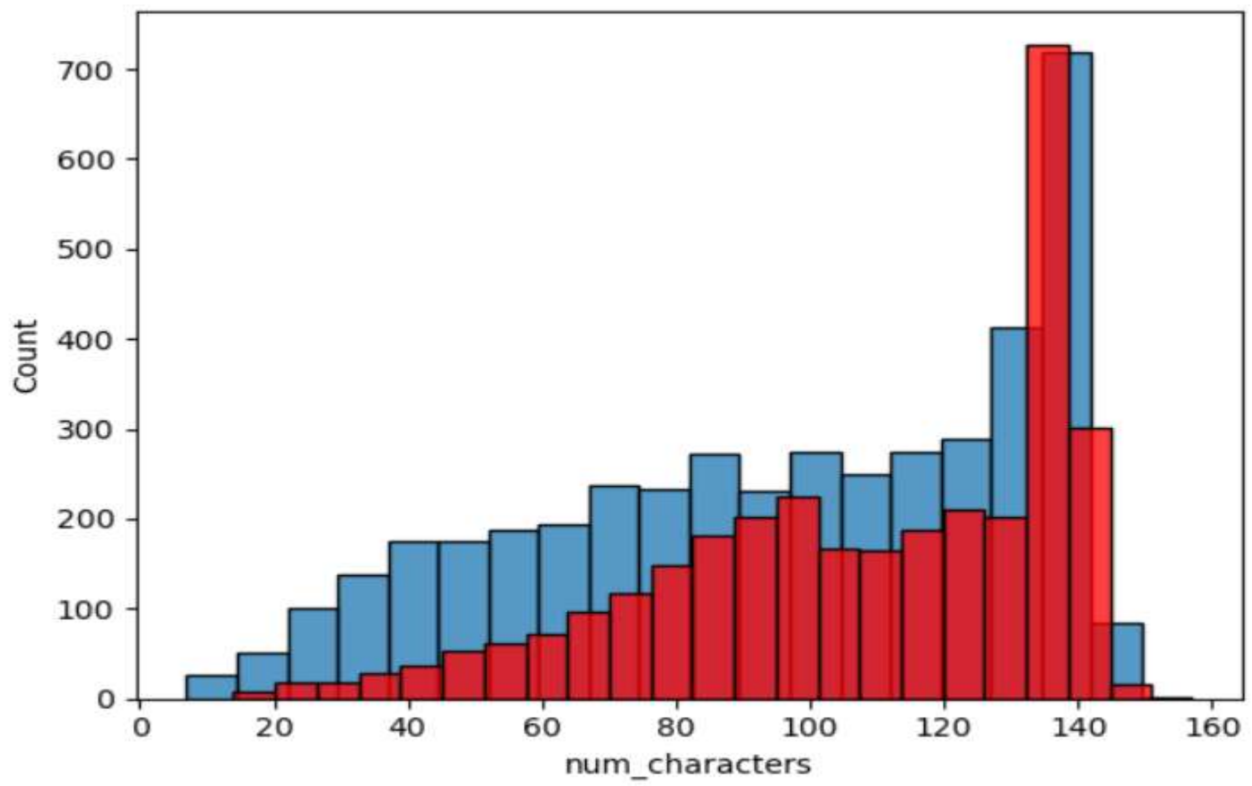


Characters used in disaster and non-disaster tweet

- Both types of tweets peak near **140 characters**, indicating users tend to use the full tweet length regardless of the topic.
- **Disaster tweets (Red)** are more concentrated in the **60–140 character range**, showing higher density in longer tweets.
- **Short tweets (< 60 characters)** are more common among **non-disaster tweets**.
- This suggests that **disaster-related tweets often contain more detailed content**, possibly including descriptions, locations, or emergency details.

```
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='Red')
```

```
<Axes: xlabel='num_characters', ylabel='Count'>
```



Pair plot

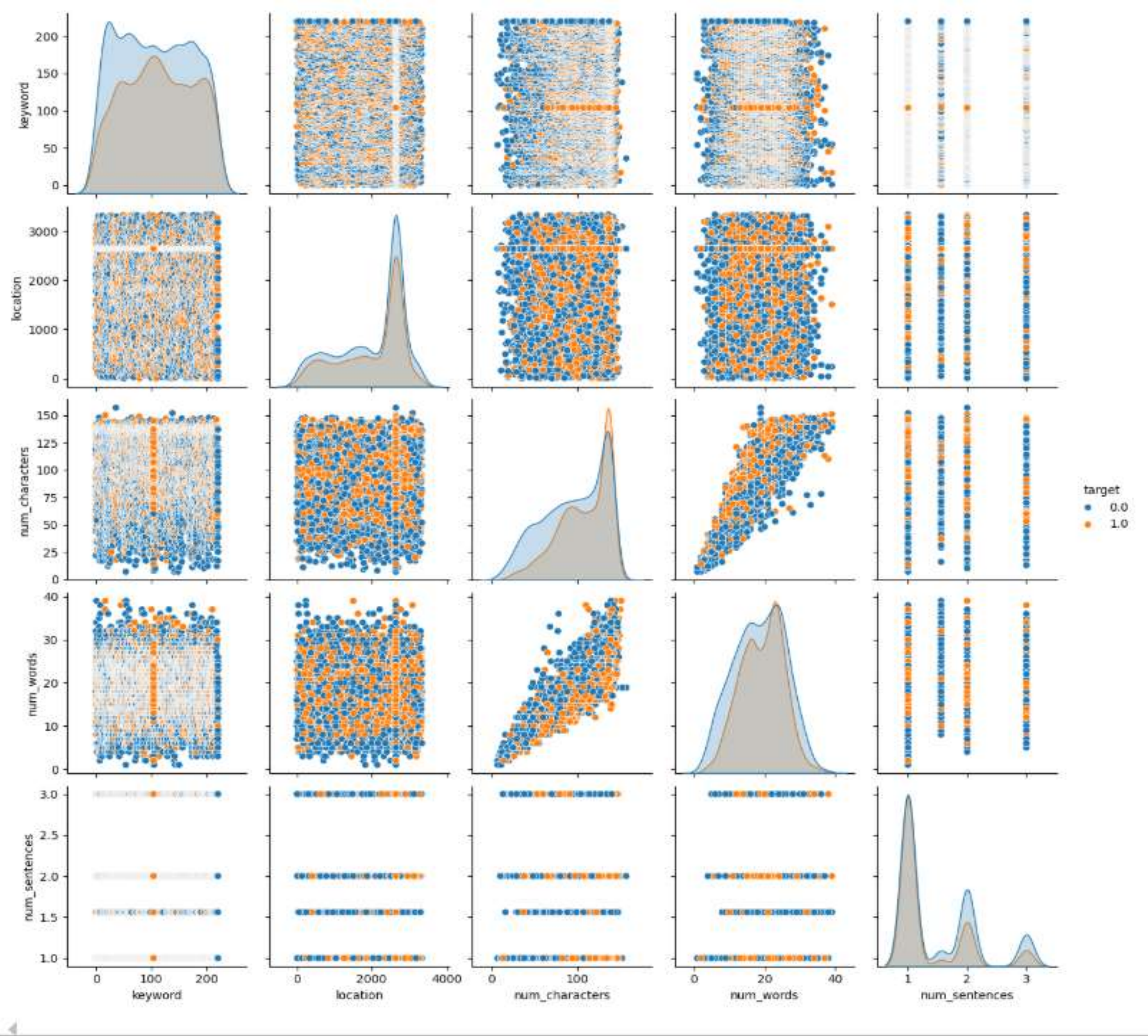
•**num_characters vs num_words**:
There is a strong **positive correlation**. As character count increases, word count also rises in both tweet types.

•**num_sentences**:
Most tweets contain **1–2 sentences**. No major difference in sentence count between disaster and non-disaster tweets.

•**keyword and location**:
These features are **categorical and sparse**. No clear distinction visible in this format — requires further preprocessing like encoding or grouping.

•**Disaster tweets** (orange) tend to have slightly **higher character and word counts**, suggesting more detailed content.

•**Non-disaster tweets** (blue) are more frequent with lower values in num_characters and num_words.



Correlation

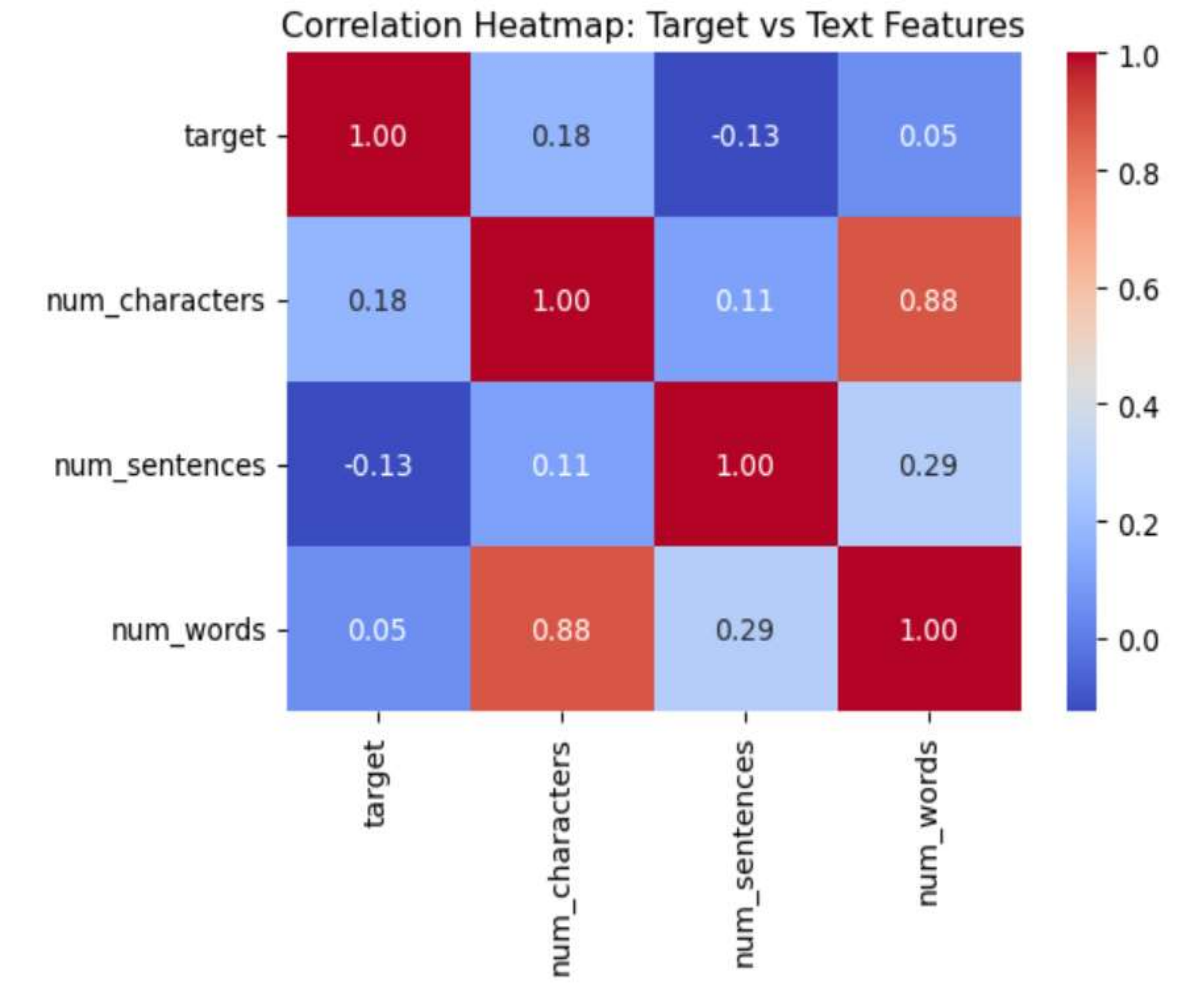
•**target** vs **num_characters**:
Correlation = **0.18** → Weak positive correlation. Disaster tweets tend to be slightly longer in character count.

•**target** vs **num_words**:
Correlation = **0.05** → Very weak correlation. Word count alone does not strongly differentiate disaster from non-disaster tweets.

•**target** vs **num_sentences**:
Correlation = **-0.13** → Slight negative correlation. Disaster tweets may be more concise in terms of sentence count.

•**num_characters** vs **num_words**:
Correlation = **0.88** → Strong positive correlation, as expected. Longer tweets have more words.

•**num_words** vs **num_sentences**:
Correlation = **0.29** → Moderate relationship, suggesting sentence count grows modestly with word count.



Disaster Word Cloud

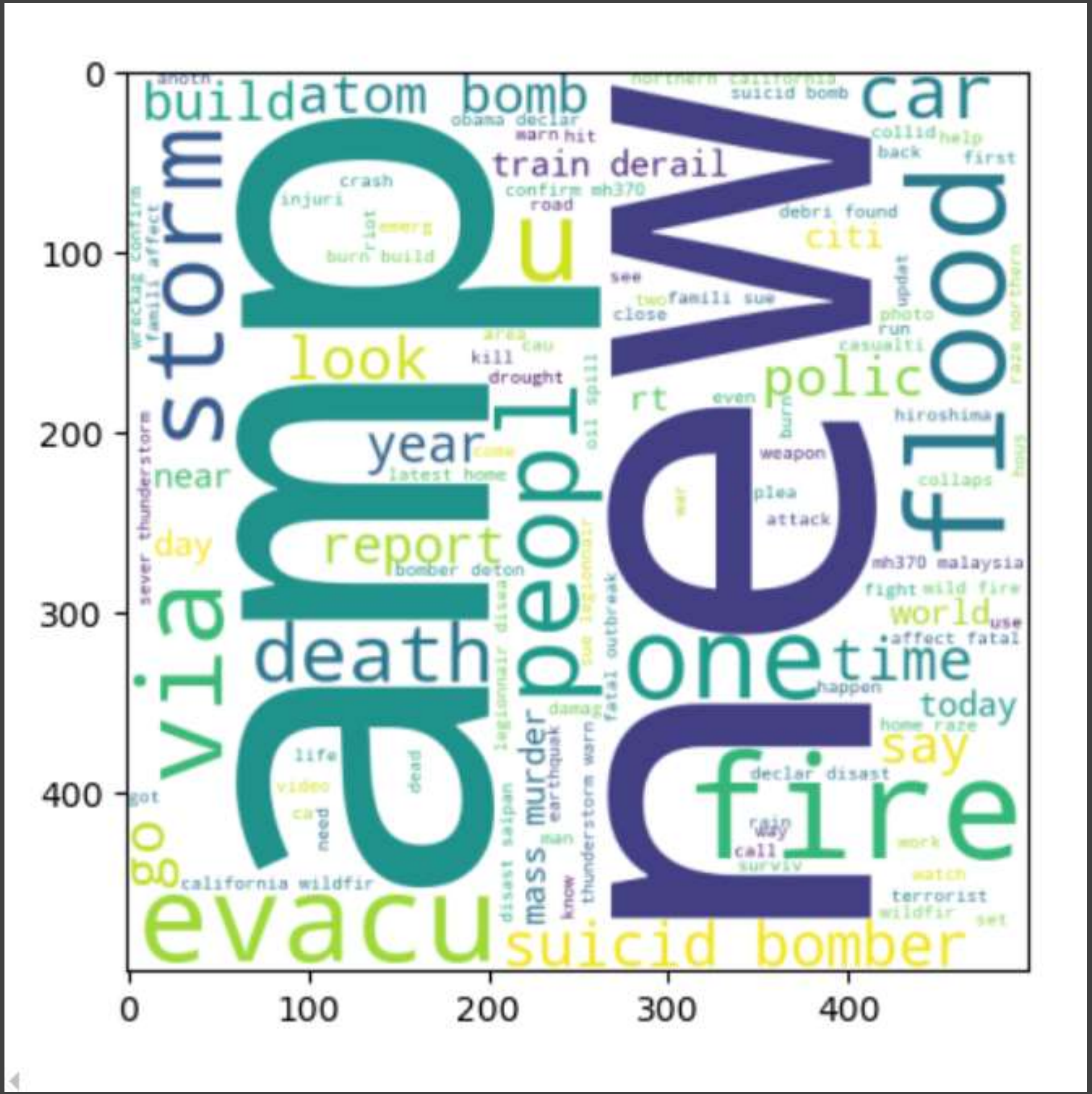
•The word cloud highlights the **most frequent words** in disaster-related tweets.

•Prominent terms include:
•🔥 **fire**, 🌊 **flood**, **storm**, 🚨 **evacu**, ☠️ **death**, 💣 **bomb**, **suicid**, **bomber**

•📢 **report**, **police**, **people**, **news**, **emergency**

•Many words reflect **emergency events**, **natural disasters**, and **violent incidents**.

•Words like **“amp”**, **“via”**, and **“news”** are common due to **retweets and shared links**.



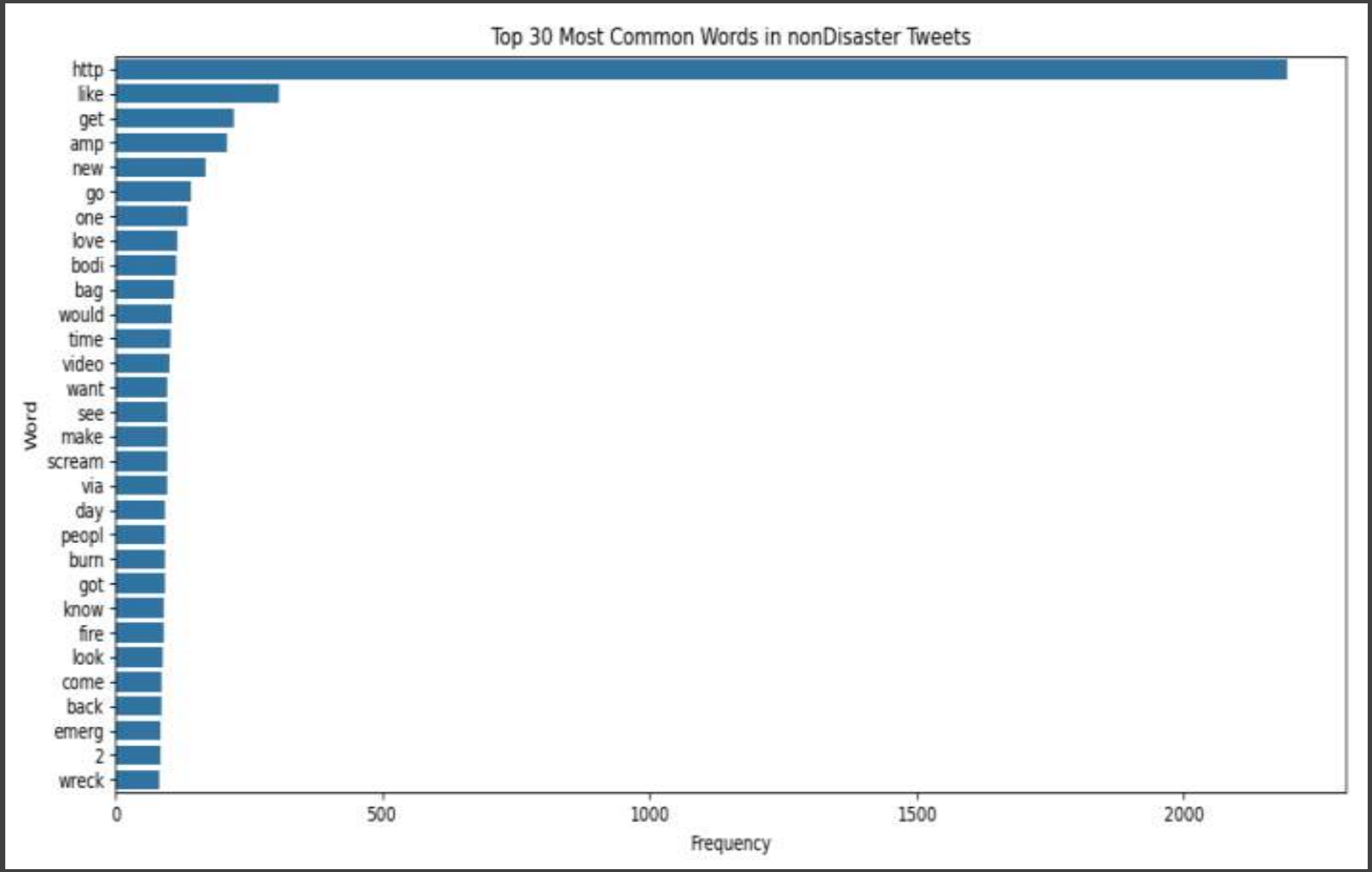
30 Most common words used in Non-Disaster Tweet

- The most frequent word is **“http”**, indicating a high number of **links and shared content** in non-disaster tweets.

- Common words like **“like”**, **“get”**, **“love”**, **“video”**, and **“time”** reflect a **casual or social tone**, unrelated to emergencies.

- Words such as **“amp”**, **“via”**, and **“new”** also appear frequently due to **retweets and media sharing**.

- A few disaster-related words like **“fire”**, **“burn”**, and **“wreck”** exist but are used in a **non-emergency context** (e.g., metaphors, sarcasm, or casual conversation).



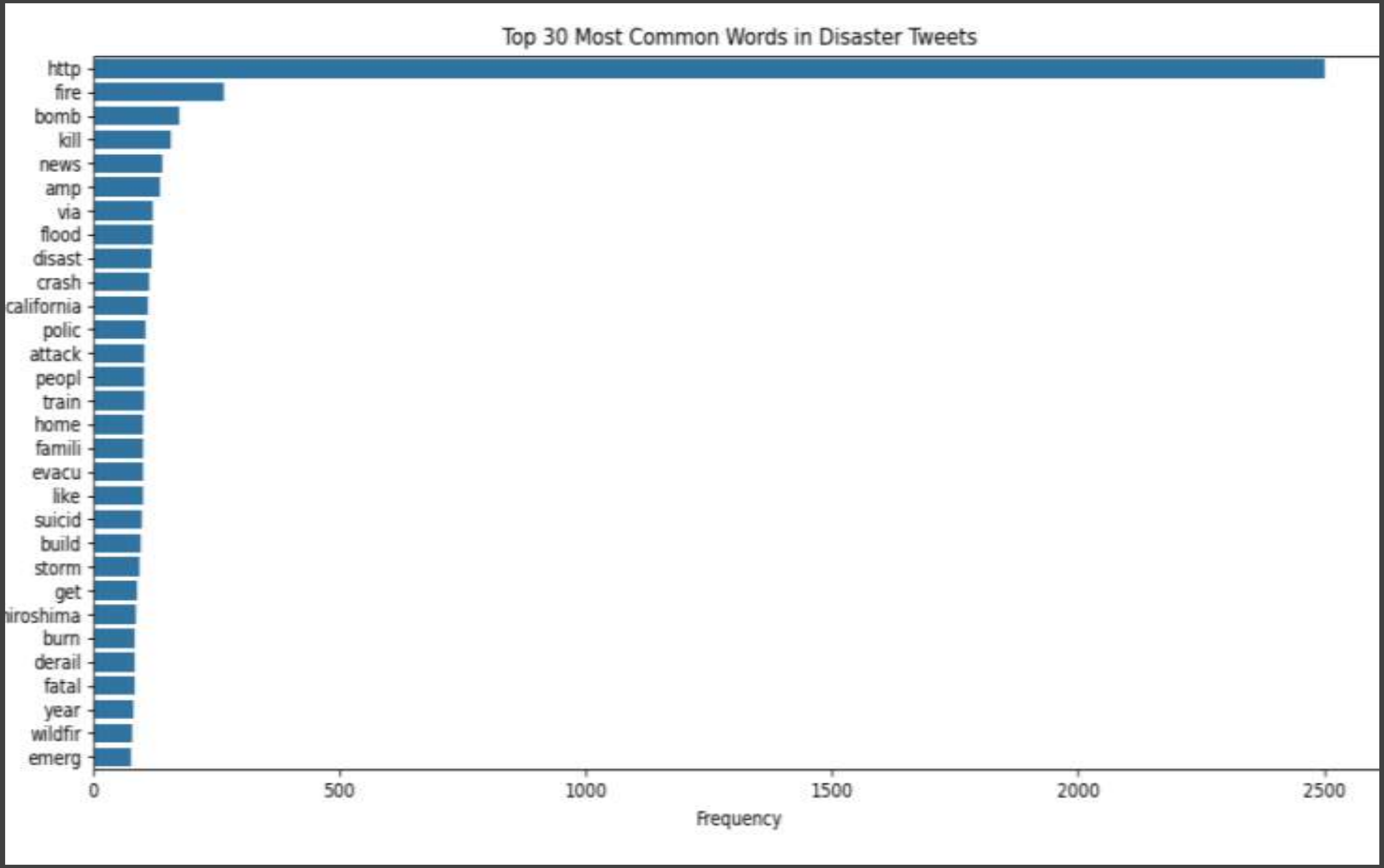
30 Most common words used in Disaster Tweet

•The most frequent term is **"http"**, showing that many disaster tweets include **links to news or emergency updates**.

•High-frequency disaster-related words include:
•🔥 **fire**, 💣 **bomb**, ☠️ **kill**, 🚨 **disast**, 🚓 **polic**, 🌊 **flood**, 🚆 **crash**, **storm**

•Other emotionally charged or action-related terms like **"attack"**, **"evacu"**, **"suicid"**, **"fatal"**, and **"burn"** are common.

•Mentions of **specific events/places** (e.g., "hiroshima", "california", "wildfir") suggest real incidents being discussed.



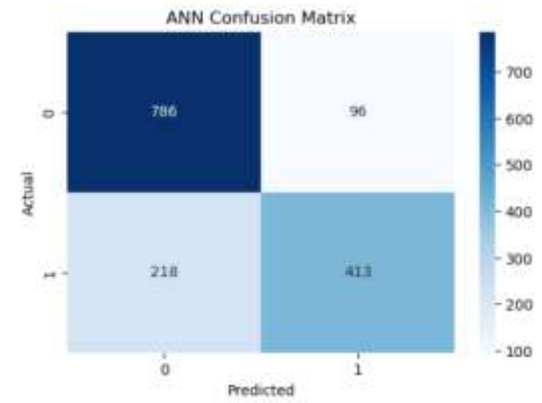
Deep Learning

ANN

```
# X_train, X_val, y_train, y_val = train_test_split(X_pad, y, test_size=0.2, random_state=42)
max_len = 50

model_ann = Sequential()
model_ann.add(Embedding(input_dim=5000, output_dim=64, input_length=max_len))
model_ann.add(GlobalAveragePooling1D())
model_ann.add(Dense(64, activation='relu'))
model_ann.add(Dropout(0.5))
model_ann.add(Dense(1, activation='sigmoid'))
from tensorflow.keras.callbacks import EarlyStopping

model_ann.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

early_stop = EarlyStopping(patience=2, restore_best_weights=True)

model_ann.fit(X_train, y_train, validation_data=(X_val, y_val),
              epochs=10, batch_size=32, callbacks=[early_stop])
```

```
Epoch 1/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 6s 10ms/step - accuracy: 0.5318 - loss: 0.6887 - val_accuracy: 0.5960 - val_loss: 0.6687
Epoch 2/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.5521 - loss: 0.6770 - val_accuracy: 0.7509 - val_loss: 0.6401
Epoch 3/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.6765 - loss: 0.6161 - val_accuracy: 0.7198 - val_loss: 0.5567
Epoch 4/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8068 - loss: 0.4630 - val_accuracy: 0.7824 - val_loss: 0.4795
Epoch 5/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8324 - loss: 0.3965 - val_accuracy: 0.7811 - val_loss: 0.4783
Epoch 6/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.8536 - loss: 0.3471 - val_accuracy: 0.8135 - val_loss: 0.4310
Epoch 7/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8824 - loss: 0.2966 - val_accuracy: 0.8036 - val_loss: 0.4660
Epoch 8/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8827 - loss: 0.2754 - val_accuracy: 0.8089 - val_loss: 0.4392

<keras.src.callbacks.history.History at 0x185ec32c1c0>
```

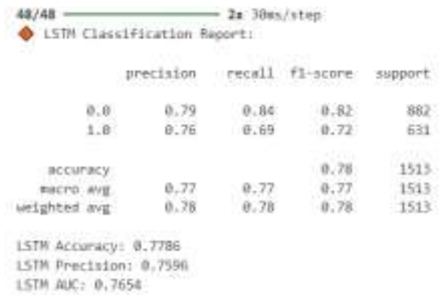
```
48/48 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
◆ ANN Classification Report:

              precision    recall  f1-score   support

         0.0       0.78      0.89      0.83       882
         1.0       0.81      0.65      0.72       631

    accuracy                           0.79      1513
   macro avg       0.80      0.77      0.78      1513
weighted avg       0.79      0.79      0.79      1513

ANN Accuracy: 0.7925
ANN Precision: 0.8114
ANN AUC: 0.7728
```



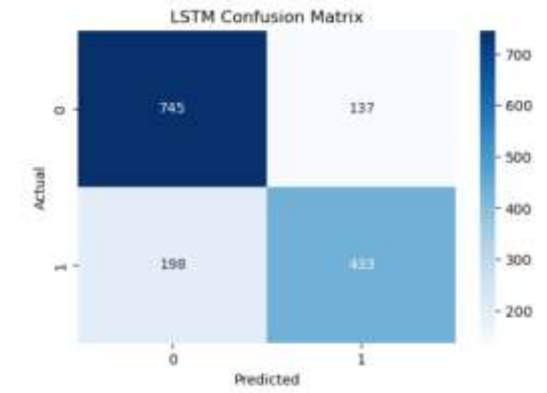
Deep Learning

```
# Define the LSTM model
model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=5000, output_dim=64, input_length=50))
model_lstm.add(LSTM(64, return_sequences=False))
model_lstm.add(Dense(64, activation='relu'))
model_lstm.add(Dropout(0.5))
model_lstm.add(Dense(1, activation='sigmoid'))

# Compile the model
model_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(patience=2, restore_best_weights=True)

model_lstm.fit(X_train_pad, y_train, validation_data=(X_val_pad, y_val),
               epochs=10, batch_size=32, callbacks=[early_stop])
```

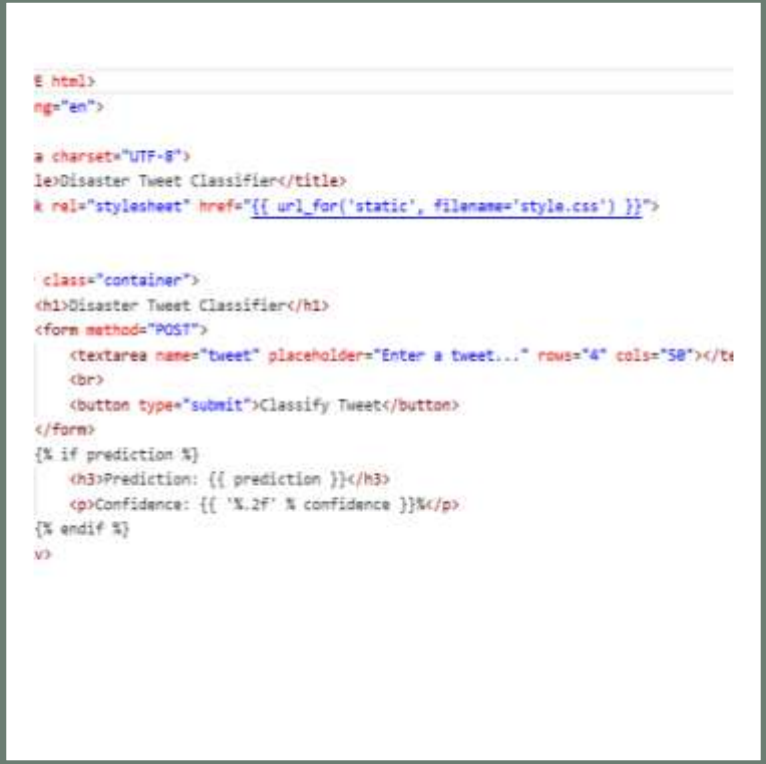
```
Epoch 1/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 15s 57ms/step - accuracy: 0.6250 - loss: 0.6535 - val_accuracy: 0.7884 - val_loss: 0.4598
Epoch 2/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 9s 58ms/step - accuracy: 0.8451 - loss: 0.3628 - val_accuracy: 0.7903 - val_loss: 0.4545
Epoch 3/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 8s 54ms/step - accuracy: 0.9124 - loss: 0.2442 - val_accuracy: 0.7632 - val_loss: 0.5586
Epoch 4/10
142/142 ━━━━━━━━━━━━━━━━━━━━ 8s 56ms/step - accuracy: 0.9384 - loss: 0.1738 - val_accuracy: 0.7579 - val_loss: 0.6531

<keras.src.callbacks.history.History at 0x185f0ec6b00>
```



```
48/48 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step
🔶 LSTM Classification Report:

              precision    recall  f1-score   support

         0.0       0.79      0.84      0.82       882
         1.0       0.76      0.69      0.72       631

    accuracy                           0.78      1513
   macro avg       0.77      0.77      0.77      1513
weighted avg       0.78      0.78      0.78      1513

LSTM Accuracy: 0.7786
LSTM Precision: 0.7596
LSTM AUC: 0.7654
```



HTML , app.py


E html>
ng="en">
a charset="UTF-8">
le>Disaster Tweet Classifier</title>
k rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
class="container">
<h1>Disaster Tweet Classifier</h1>
<form method="POST">
<textarea name="tweet" placeholder="Enter a tweet..." rows="4" cols="50"></te

<button type="submit">Classify Tweet</button>
</form>
{% if prediction %}
<h3>Prediction: {{ prediction }}</h3>
<p>Confidence: {{ '%.2f' % confidence }}%</p>
{% endif %}
v>


Disaster Tweet Classifier
Enter a tweet...
Classify Tweet
Prediction: Not a Disaster
Confidence: 72.83%


Disaster Tweet Classifier
Enter a tweet...
Classify Tweet
Prediction: Disaster
Confidence: 95.12%

Thank You