

Disaster Tweets Classifier

**From Chaos to Classification:
NLP for disaster detection in tweets**

Project Introduction

Goal

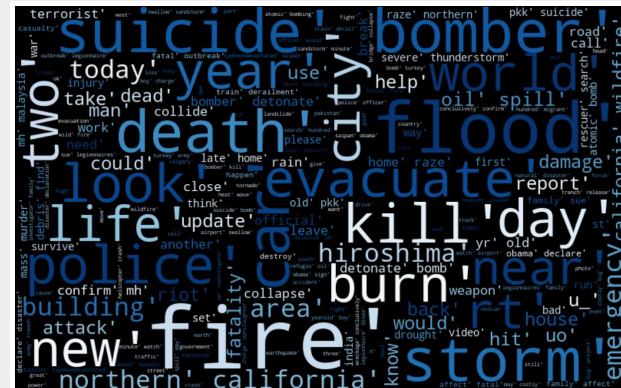
Predict whether a tweet refers to a real disaster (1) or not (0).

Impact

Rapid detection of disaster-related tweets can support emergency services & crisis management.

Data

7,613 labeled tweets
(text + target).



Pipeline

The Latest: More homes razed by Northern California wildfire - <http://t.co/BsGR67dyWY>
<http://t.co/nDGgr6Xyqg>

the latest more homes razed by northern california wildfire

late home raze northern california wildfire

Data Cleaning & Preprocessing

- Lowercasing, removing noise
- POS-tagged lemmatization
- Stopword removal

Feature Engineering

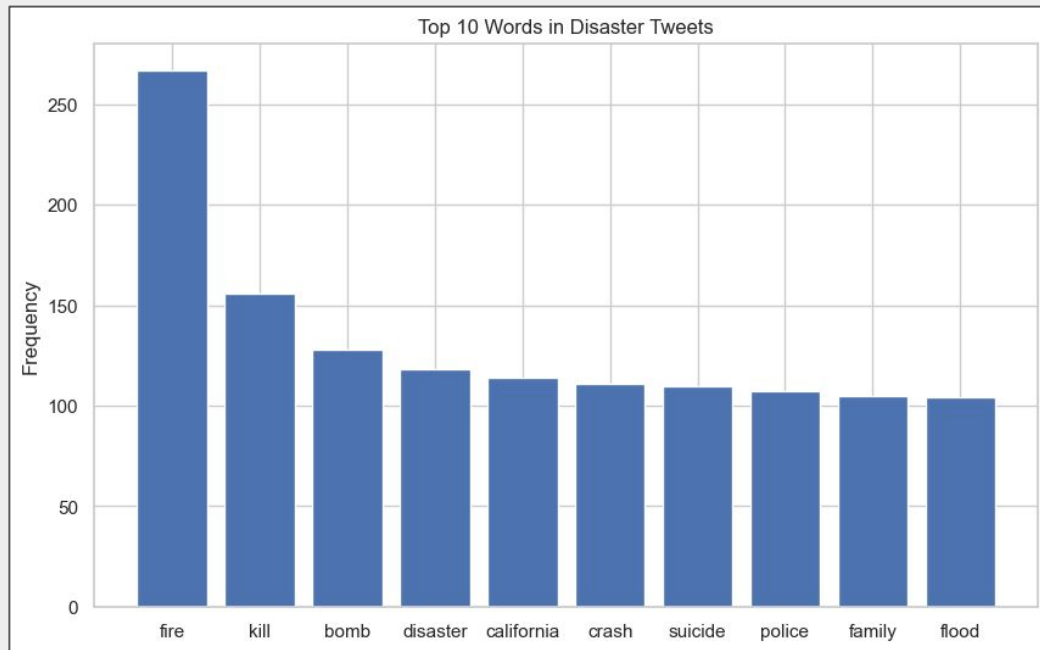
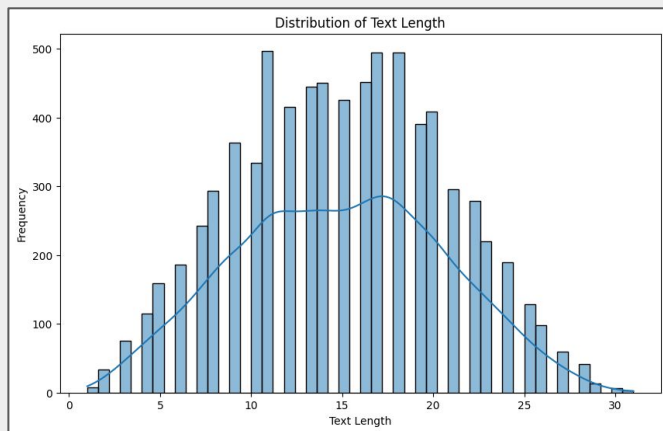
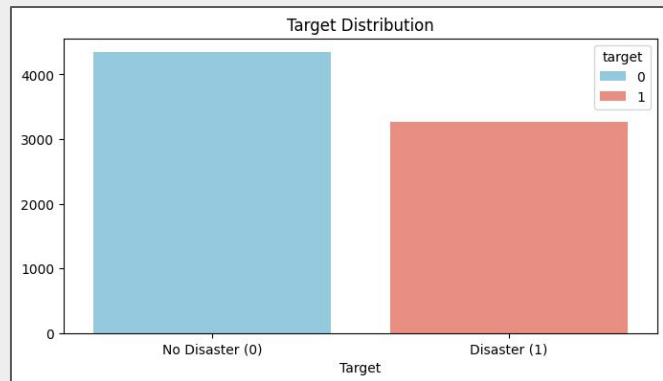
- CountVectorizer (n-grams)
- TF-IDF Vectorization
- Binary Count Vectorization for specific NB experiments

Model Training & Evaluation

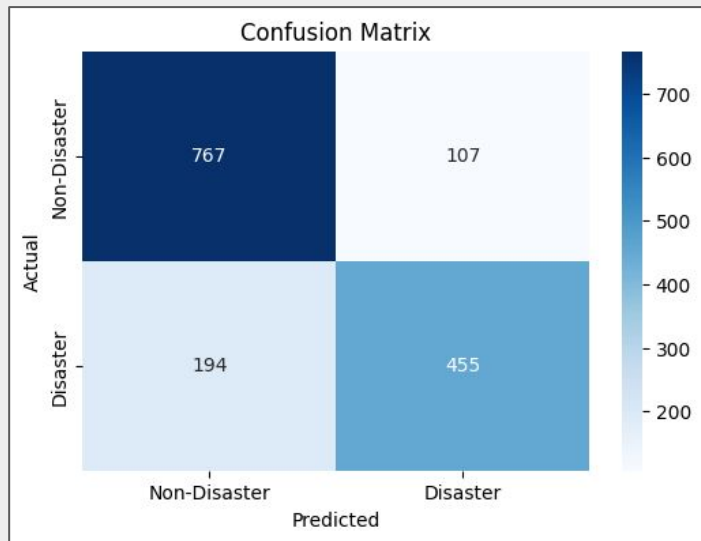
Models: Logistic Regression, Naive Bayes & CNN

Evaluation: Accuracy, Precision, Recall, F1-score, Confusion matrix & Classification report

Data Analysis



Logistic Regression (Baseline)



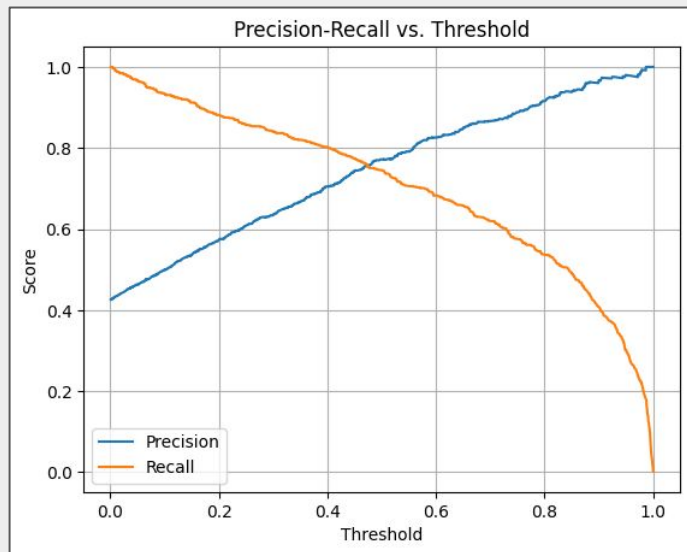
Accuracy	Precision	Recall	F1-Score
0.80	0.80	0.79	0.79

Why Logistic Regression?

Simple, fast, interpretable baseline for text classification.

- High precision, but recall lower → missed some disaster tweets.
- Useful as a performance benchmark for other models.

Logistic Regression (balanced model)



Threshold = 0.50	precision	recall	f1-score	support
Not Disaster	0.80	0.88	0.84	874
Disaster	0.81	0.70	0.75	649
accuracy			0.80	1523
macro avg	0.80	0.79	0.79	1523
weighted avg	0.80	0.80	0.80	1523

Goal: Improve recall to catch more disaster tweets.

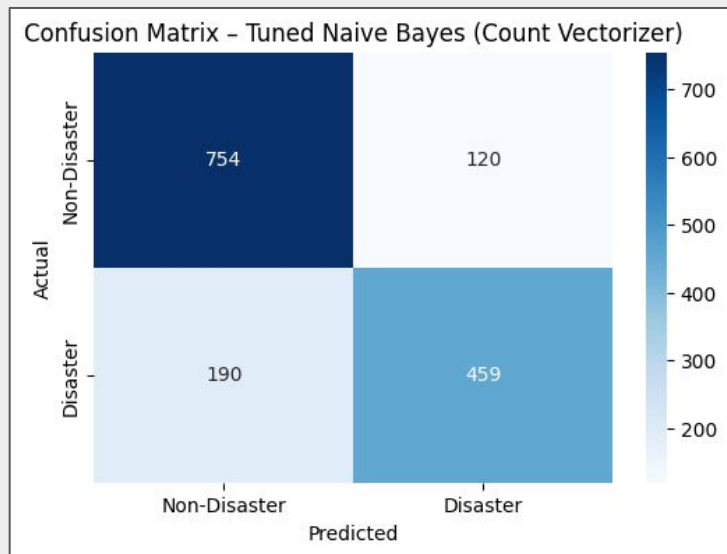
- Balanced class weights.
- Cross-validation
- Adjusted threshold

Outcome:

- Recall improved slightly
- but precision dropped noticeably.

Threshold = 0.45	precision	recall	f1-score	support
Not Disaster	0.83	0.80	0.81	874
Disaster	0.74	0.77	0.76	649
accuracy			0.79	1523
macro avg	0.79	0.79	0.79	1523
weighted avg	0.79	0.79	0.79	1523

Naive Bayes



Accuracy	Precision	Recall	F1-Score
0.80	0.80	0.78	0.79

Why Naive Bayes?

Efficient for sparse, high-dimensional text data.

Experiments:

- CountVectorizer (baseline NB)
- Tuned alpha smoothing values
- TF-IDF (regular & tuned alpha)

Key Insight:

- More balanced precision/recall than LR, but slightly lower accuracy.

Convolutional Neural Network (CNN)

Why CNN?

- Captures local word patterns automatically.
- Learns embeddings instead of relying solely on vectorizers.

Architecture Highlights:

- Tokenization → Padding → Embedding Layer
- Convolution + MaxPooling layers
- Dense + Dropout for regularization
- Sigmoid activation for binary classification

Tokenized Sequence (word tokens)

```
[
  0 : "new"
  1 : "summer"
  2 : "long"
  3 : "thin"
  4 : "body"
  5 : "bag"
  6 : "hip"
  7 : "word"
  8 : "skirt"
  9 : "blue"
]
```

Token-Index Mapping ↵

```
{
  "new" : 50
  "summer" : 484
  "long" : 524
  "thin" : 5276
  "body" : 80
  "bag" : 322
  "hip" : 3632
  "word" : 1390
  "skirt" : 5238
  "blue" : 922
}
```

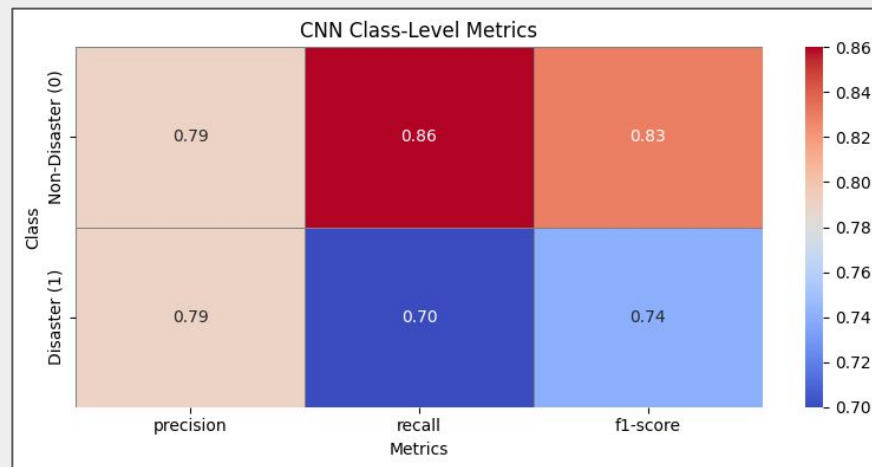
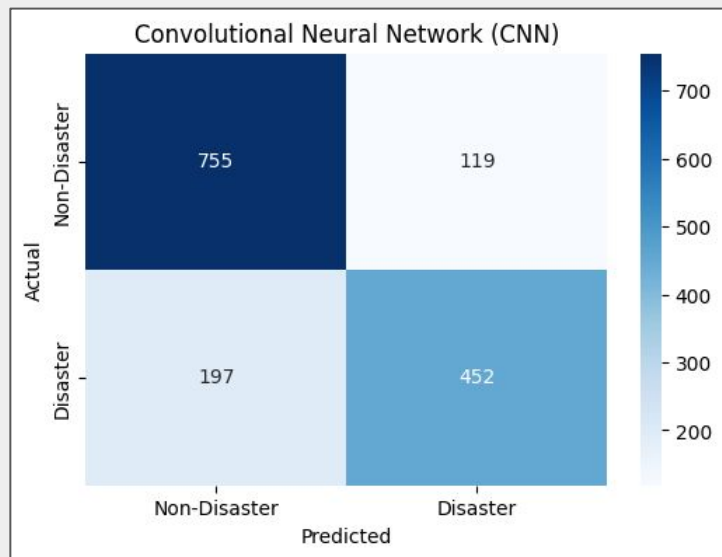
Vectorized Sequence (token IDs)

```
[
  0 : 50
  1 : 484
  2 : 524
  3 : 5276
  4 : 80
  5 : 322
  6 : 3632
  7 : 1390
  8 : 5238
  9 : 922
]
```

Padded Sequence (length = 100)

```
[
  0 : 50
  1 : 484
  2 : 524
  3 : 5276
  4 : 80
  5 : 322
  6 : 3632
  7 : 1390
  8 : 5238
  9 : 922
  10 : 0
]
```


Convolutional Neural Network (CNN)






Accuracy	Precision	Recall	F1-Score
0.79	0.79	0.78	0.78

🏆 **Best overall balance between precision and recall.**

Outperformed LR and NB in recall while keeping precision high.

Model Comparison & Final Selection

Model	Accuracy	Precision	Recall	F1-score
 CNN	0.79	0.79	0.78	0.78
 Logistic Regression	0.80	0.80	0.79	0.79
 Naive Bayes (TF-IDF)	0.80	0.80	0.78	0.79

Final Model: CNN

- Balanced performance across precision and recall.
- Slightly lower accuracy than LR and NB, but strong in disaster detection without over-flagging.
- LR and NB, remain strong, efficient alternatives for production

Model Demo

Features:

- User Tweet input or random Tweet from test set
- Preprocessing steps
- Info page

Technical Setup:

- Saved CNN model + Tokenizer
- Same preprocessing pipeline as training

The screenshot shows the 'Random Tweet Classifier' interface. A sidebar on the left contains a 'Disaster Tweet Classifier' header, a description, a 'How to Use' button, and a 'Navigate' section with four radio buttons: 'Classify a Tweet', 'Random Tweet Test' (selected), 'EDA & Word Analysis', and 'Model Summary'. The main content area is titled 'Random Tweet Classifier' and includes a description, a 'Generate and Classify Random Tweet' button, and a section for the 'Original Tweet' (a tweet about Boise forest fires). Below this is a 'Preprocessing Steps' button. The 'Prediction' is 'Disaster' with a confidence of 85.35% and a 'Correct' status. Annotations with blue boxes and arrows point to the 'User Input' (the 'Generate and Classify Random Tweet' button), 'Generate Test Tweet' (the same button), and 'Preprocessing Steps' (the 'Preprocessing Steps' button).

User Input

Disaster Tweet Classifier
This app classifies tweets as related to real disasters or not.
Use the sidebar to navigate between different functionalities.
> How to Use

Navigate
☐ Classify a Tweet
☒ Random Tweet Test
☐ EDA & Word Analysis
☐ Model Summary

Random Tweet Classifier
This page generates a random tweet from the test dataset and classifies it, showing the preprocessing steps and prediction.
Check out the preprocessing steps to understand how the tweet is processed before classification.

Generate and Classify Random Tweet

Original Tweet
8-5-2015 - 4:30 P.M. - Progress Being Made on Boise Forest Fires <http://t.co/6o1mgMGHgt>
<http://t.co/wTPO6elRZd>

> 🔍 Preprocessing Steps

Prediction: 🌪️ Disaster
Confidence: 85.35%
✅ Correct

Generate Test Tweet

Preprocessing Steps

Conclusion

- ✓ Created NLP pipeline
- ✓ CNN achieved the best balanced performance.
- ✓ Streamlit app provides an interactive, transparent model demo.

Key Learnings:

- 🎯 Preprocessing decisions improved generalization.
- 🎯 Boosting recall often lowers precision — finding balance is key.
- 🎯 A well-documented, reproducible pipeline is essential for deployment readiness.