Disaster Tweet Classification

An End-to-End NLP Pipeline for Binary Tweet Classification

Disaster vs. Non-Disaster Tweet Detection using Classical Machine Learning and Text Vectorisation

Author:	Guy M. Kaptue T.
GitHub:	github.com/guykaptue

Project Type:	Natural Language Processing / Binary Classification
Dataset:	Kaggle — Natural Language Processing with Disaster Tweets
Framework:	Python 3.12 · scikit-learn · NLTK · Gensim
Primary Metric:	F1-Score
Final Result:	F1 = 0.7715 · ROC-AUC = 0.8625

Report generated from the reproducible Jupyter Notebook
`katastrophen_Tweets_final.ipynb`

github.com/guykaptue

Contents

1 Executive Summary

This report documents a complete, reproducible Natural Language Processing (NLP) pipeline developed to address the Kaggle challenge *Natural Language Processing with Disaster Tweets*. The core objective is to automatically distinguish tweets describing real disaster events from those that use disaster-related language in a figurative, humorous, or metaphorical context.

The project follows a systematic, data-driven framework encompassing six major phases: exploratory data analysis (EDA), text preprocessing, text vectorisation, model training via stratified cross-validation, hyperparameter tuning, and final model selection with deployment recommendations.

The final model—**Logistic Regression with Count Bigram vectorisation**—achieves an F1-score of **0.7715** and a ROC-AUC of **0.8625** on the validation set, meeting and exceeding the project's primary target of F1 $\geq$ 0.75. A central finding of this study is that *simple, sparse token-based features combined with linear classifiers outperform complex models and dense embeddings* on this short-text, high-signal corpus.

Key Findings at a Glance

- Final model: Count Bigram + Logistic Regression (F1 = 0.7715, AUC = 0.8625)
- 7,503 training samples after deduplication; 3,243 test samples
- 7 vectorisation strategies $\times$ 7 classifiers = 49 experimental combinations evaluated
- Sparse token features consistently outperform dense word embeddings (Word2Vec, FastText)
- Linear models outperform non-linear models (XGBoost, MLP, Random Forest) throughout

2 Problem Formulation and Business Context

2.1 The Core Challenge

Social media platforms—in particular Twitter—have become a primary communication channel during disaster events. Emergency services, government agencies, media organisations, and NGOs monitor these platforms in real time to coordinate response efforts. The fundamental challenge arises from the pervasive use of disaster-related language in non-disaster contexts.

Consider the tweet: *"On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE."* The word *ablaze*, which constitutes a strong lexical signal for fire or catastrophe, here describes a vivid sunset. To a human reader the context is transparent; to a machine the distinction is non-trivial. This ambiguity—between literal and figurative usage of disaster vocabulary—constitutes the core challenge addressed in this project.

2.2 Stakeholder Groups and Business Value

The ability to automatically classify disaster-related tweets delivers concrete economic and societal value across multiple stakeholder groups:

Table 1: Stakeholder Groups and Operational Use Cases

Stakeholder	Use Case	Business Value
Emergency Services	Real-time situational awareness	Faster resource allocation; reduced response times
Media Organisations	Breaking-news monitoring	Early access to unfolding events
Insurance Companies	Damage potential early detection	Real-time risk scoring
NGOs & Aid Organisations	Disaster early warning	Optimised personnel and resource deployment
Social Media Platforms	Content moderation	Regulatory compliance, reputation protection

A 5–10% improvement in tweet classification F1-score can, in practice, mean that a rescue operation is initiated **10–30 minutes earlier**—a difference that may save lives.

2.3 Project Objectives

The project pursues three explicitly defined objectives:

Objective 1. **Technical:** Develop a robust NLP classification model that reliably classifies tweets as *Disaster* (1) or *Non-Disaster* (0), with a primary optimisation target of F1-Score ≥ 0.75 on the validation set.

Objective 2. **Analytical:** Ensure full pipeline transparency—every design decision, from tokenisation to model selection, is explicitly justified and grounded in data.

Objective 3. **Business:** Derive concrete, actionable recommendations for deploying the model in a production monitoring environment.

3 Dataset and Exploratory Data Analysis

3.1 Dataset Overview

The dataset originates from the Kaggle competition *Natural Language Processing with Disaster Tweets*. It consists of manually labelled tweets with the following schema:

Table 2: Dataset Schema and Dimensions

Split	Samples	Columns
Training set	7,613	id, keyword, location, text, target
Test set (holdout)	3,263	id, keyword, location, text

The binary target variable encodes: 1 = Disaster Tweet, 0 = Non-Disaster Tweet.

3.2 Class Distribution

After deduplication, the training set contains 7,503 samples with the following class distribution:

- Non-Disaster (0): 4,305 samples — 57.4%
- Disaster (1): 3,198 samples — 42.6%
- Class ratio (0/1): 1.35

This *moderate imbalance* (ratio < 2.0) does not warrant aggressive resampling. Instead, the strategy of setting `class_weight='balanced'` (sklearn) and `scale_pos_weight=1.35` (XGBoost) is applied throughout.

3.3 Duplicate and Label Noise Analysis

A systematic duplicate analysis revealed 110 exact text duplicates in the training set and 20 in the test set. More critically, some identical tweet texts carry conflicting `target` labels, constituting clear label noise. Post-cleaning, the training set is reduced to 7,503 unique records. No conflicting labels were detected after deduplication.

3.4 Linguistic Patterns: Disaster vs. Non-Disaster

Token-frequency analysis on the raw texts reveals structurally distinct linguistic profiles:

Disaster tweets are characterised by:

- Reporting, event-oriented vocabulary (*fire, after, police, california, suicide, disaster*)
- High URL prevalence (66.2% of disaster tweets contain URLs vs. 41.7% of non-disaster tweets)
- Higher hashtag density (0.498 vs. 0.383 per tweet)
- Longer texts (avg. 108 characters vs. 96)
- More punctuation (3.40 vs. 2.86 per tweet)

Non-Disaster tweets exhibit:

- Personal, dialogic vocabulary (*like, get, new, one, day, time, video*)
- Higher mention density (0.422 vs. 0.275 per tweet)
- Greater lexical diversity (9,591 unique tokens vs. 7,486)

3.5 URL Analysis

URL analysis confirms that social media short-links (predominantly `t.co`) account for nearly all URLs in both the training and test sets. The domain type distribution is consistent across splits, indicating no domain shift between training and evaluation data—a positive indicator for model generalisability.

4 Preprocessing Pipeline

4.1 Pipeline Architecture

The preprocessing pipeline is encapsulated in a reusable `TextPreprocessor` class. Each transformation stage is stored in a `result` dictionary and persisted as a dedicated DataFrame column, ensuring full transparency and reproducibility. The pipeline proceeds through five stages:

1. **Tokenisation:** Lowercasing; URL, mention, and special character removal; NLTK `word_tokenize`
2. **Stopword filtering:** 198-word English stoplist; prepositions retained (semantically relevant for location/event descriptions)
3. **Lemmatisation:** NLTK `WordNetLemmatizer`
4. **Token cleaning:** Removal of single-character tokens
5. **Final text reconstruction:** Join of cleaned tokens into `clean_text`

4.2 Structural Feature Extraction

Eleven structural features are extracted *before* text normalisation, as downstream transformations would otherwise destroy the raw signal:

- URL presence and count (`has_url`, `url_count`)
- Character count, word count, average word length
- Hashtag count, mention count
- Punctuation count, upper-case ratio
- Keyword context (Top-30 keyword one-hot encoding, 31 binary features)

These 11 numeric features, combined with keyword dummies, produce a 42-dimensional feature matrix (X_{features}) that complements the text-based representations.

4.3 Balancing Strategy

Given the moderate class imbalance (ratio = 1.35), the selected strategy is:

- `class_weight='balanced'` for all scikit-learn estimators
- `scale_pos_weight=1.35` for XGBoost
- Aggressive oversampling (SMOTE) explicitly rejected, as it introduces artificial patterns and risks overfitting at this imbalance level

5 Text Vectorisation

5.1 Vectorisation Strategies

Seven distinct text vectorisation strategies are implemented to systematically compare sparse and dense representational approaches:

Table 3: Overview of Vectorisation Strategies

Method	Variant	Vocabulary	Sparsity	Type
TF-IDF	Unigram (1-gram)	5,705	99.8%	Sparse
TF-IDF	Bigram (1–2 gram)	12,000+	99.9%	Sparse
TF-IDF	Trigram (1–3 gram)	18,130	99.9%	Sparse
Count BOW	Unigram	5,705	99.8%	Sparse
Count BOW	Bigram (1–2 gram)	12,000+	99.9%	Sparse
Word2Vec	Mean pooling (100d)	5,823	0%	Dense
FastText	Mean pooling (100d)	5,823	0%	Dense

5.2 Key Observations from Vectorisation Analysis

Dimensionality explosion with N-grams: Vocabulary size increases from 5,705 terms (unigrams) to 18,130 terms (trigrams, +218%), adding computational overhead with marginal classification benefit.

Extreme sparsity of classical methods: TF-IDF and Count vectors exhibit 99.8–99.9% sparsity, meaning over 99% of the feature matrix is empty. Conversely, Word2Vec and FastText produce dense 100-dimensional vectors with 0% sparsity.

FastText robustness: FastText captures subword information, making it more robust to typos and informal language typical of social media (e.g., *"flod"* ≈ *"flood"*). FastText embeddings also show higher L2-norms for disaster tweets, suggesting better class separation in embedding space.

PCA/t-SNE analysis: Dimensionality reduction visualisations confirm that both classes are discernible in all representational spaces. TF-IDF unigram PCA plots show cleaner cluster separation than dense embedding plots, foreshadowing the superior downstream classification performance of sparse methods.

6 Model Training and Cross-Validation

6.1 Experimental Design

The model selection framework systematically evaluates 7 classifiers across 7 vectorisation strategies using Stratified 5-Fold Cross-Validation. The model candidates are:

1. Logistic Regression (LogReg)
2. Naive Bayes (MultinomialNB / ComplementNB)
3. Linear SVM (LinearSVC)
4. Random Forest
5. Multilayer Perceptron (MLP)
6. XGBoost
7. K-Nearest Neighbours (KNN)

Naive Bayes is only applied to sparse (non-negative) vectorisations. This yields up to 49 model–vectoriser combinations, all evaluated on five metrics: Accuracy, Precision, Recall, F1-Score, and ROC-AUC.

6.2 Cross-Validation Results

Table 4: Top-10 Model–Vectoriser Combinations by Mean F1-Score (5-Fold CV)

Rank	Combination	F1 (mean)	F1 (std)	Accuracy	AUC
1	tfidf_unigram × LogReg	0.7589	±0.0069	–	–
2	tfidf_bigram × LogReg	0.7557	±0.0058	–	–
3	count_bigram × LogReg	0.7551	±0.0054	–	–
4	count_bow × LogReg	0.7543	±0.0063	–	–
5	tfidf_trigram × LogReg	0.7512	±0.0062	–	–
6	tfidf_unigram × NaiveBayes	0.7505	±0.0081	–	–
7	count_bow × LinearSVM	0.7480	±0.0070	–	–
8	tfidf_bigram × LinearSVM	0.7467	±0.0072	–	–
9	tfidf_unigram × LinearSVM	0.7461	±0.0068	–	–
10	count_bigram × LinearSVM	0.7448	±0.0074	–	–

Central Finding: Linear models (LogReg, LinearSVM) with sparse token features (TF-IDF, Count-BOW) dominate the ranking. This pattern holds consistently across all evaluation metrics and is *not* an artefact of any single metric.

6.3 Why Simplicity Wins

Four structural properties of the corpus explain the consistent advantage of sparse token features with linear classifiers:

1. **High signal density in short texts:** Tweets are limited to 280 characters. Each word carries proportionally more predictive weight than in longer documents. Token-based models exploit this directly; embeddings average it away.
2. **Domain-specific vocabulary:** Disaster tweets have a clearly bounded semantic field (*fire, flood, earthquake, debris, victims*). TF-IDF precisely identifies class-specific terms without semantic noise from contextually similar but irrelevant words.
3. **Keyword-dominated signal structure:** The most predictive features are lexical, not positional or compositional. Bag-of-words approaches are optimally suited for this signal type.
4. **Linear separability:** PCA plots confirm that the classes are approximately linearly separable in TF-IDF space. Non-linear models (XGBoost, MLP) learn boundaries that are more complex than necessary, introducing overfitting risk.

7 Baseline Training and Hyperparameter Optimisation

7.1 Baseline Systems

The best model per vectoriser family is selected from the CV study and trained on the full training set. Four baseline systems are established:

Table 5: Baseline System Performance on Validation Set

System	Vectoriser × Model	Acc.	Prec.	Recall	F1	AUC
TF-IDF	tfidf_unigram × LogReg	0.810	0.793	0.750	0.771	0.866
Count	count_bigram × LogReg	0.806	0.801	0.723	0.760	0.859
Word2Vec	word2vec_mean × LinearSVM	0.782	0.747	0.739	0.743	0.852
FastText	fasttext_mean × LinearSVM	0.756	0.713	0.717	0.715	0.820

7.2 Hyperparameter Optimisation

RandomizedSearchCV with 30 iterations and 5-fold stratified CV is applied to all four baseline systems. Model-specific search spaces include:

- **LogReg:** C $\in \{0.01, 0.1, 0.5, 1, 5, 10, 50\}$; penalty $\in$ {l1, l2}; solver, max_iter
- **LinearSVM:** C, max_iter, loss function (hinge / squared_hinge)

Tuning outcomes:

Table 6: Performance Before and After Hyperparameter Tuning

System	Baseline F1	Tuned F1	ΔF1	Best Params
TF-IDF × LogReg	0.7711	0.7705	−0.0006	C=1.0, l2, liblinear
Count × LogReg	0.7603	0.7715	+0.0112	C=0.5, l2, saga
Word2Vec × LinearSVM	0.7431	∼0.745	∼+0.002	–
FastText × LinearSVM	0.7150	∼0.718	∼+0.003	–

The tuning results confirm that the baseline models are already near-optimal for their respective feature types. Count Bigram + LogReg benefits most from tuning (+1.1%), narrowly surpassing TF-IDF + LogReg and becoming the final selected model.

8 Final Model and Results

8.1 Final Model Selection

The model with the highest F1-score on the validation set after hyperparameter tuning is selected as the production candidate:

Final Model: Count Bigram × Logistic Regression

Vectoriser:	`count_bigram` (1–2 gram Count Vectoriser)
Classifier:	Logistic Regression
Hyperparameters:	solver=saga, penalty=l2, max_iter=2000, C=0.5
Validation F1:	**0.7715**
Validation ROC-AUC:	**0.8625**

8.2 Detailed Performance Report

Table 7: Final Model: Classification Report on Validation Set

Class	Precision	Recall	F1-Score	Support
Non-Disaster (0)	0.82	0.87	0.84	861
Disaster (1)	0.81	0.74	0.77	641
Weighted Avg	–	–	0.81	1,502

8.3 Feature Importance Analysis

Logistic Regression coefficient analysis identifies the following as the most discriminative features for the *Disaster* class:

- High-positive coefficients: *suicide bombing, body bag, derailment, wildfire, typhoon, earthquake, debris, casualties*
- High-negative coefficients: *like, lol, omg, haha, cute, beautiful, love*

This pattern confirms that the model has learned semantically meaningful class boundaries rather than spurious correlations.

8.4 Error Analysis

False Positive analysis (Non-Disaster classified as Disaster) reveals that most misclassifications arise from:

- **Metaphorical language:** "*The market is on fire*", "*This performance was explosive*"
- **Irony and sarcasm:** "*I am literally dying from cringe*"

False Negatives (missed Disaster tweets) typically involve:

- Disaster descriptions without canonical event keywords
- Highly contextual or indirect references to events

These error modes are inherent limitations of bag-of-words representations and would require transformer-based architectures (BERT, RoBERTa) for meaningful improvement.

9 Business Evaluation and Deployment Recommendations

9.1 Production Readiness Assessment

Table 8: Production Readiness Assessment Matrix

Dimension	Rating	Justification
Technical Maturity	**Good**	F1=0.77, AUC=0.86 — industry standard for this task
Interpretability	**Excellent**	LogReg coefficients fully explainable
Inference Speed	**Excellent**	<1 ms per tweet — real-time capable
Robustness	**Good**	Low CV variance (±0.007); stable across subsets
Scalability	**Good**	TF-IDF + LogReg scales to millions of tweets
Generalisability	**Limited**	English only; trained on specific time period

9.2 Recommended Deployment Architecture

Recommendation 1 — Threshold Optimisation: The default threshold of 0.5 is sub-optimal across all use cases. Stakeholder-specific thresholds are recommended:

- *Emergency services (Recall priority):* threshold = 0.30–0.35 (more True Positives; false alarm costs are low relative to missed events)
- *Content moderation (Precision priority):* threshold = 0.65–0.70 (fewer false alarms; some missed disasters are acceptable)

Recommendation 2 — Confidence-Based Routing: Not all predictions are equally reliable. A three-tier routing logic is proposed:

- *High confidence* ($p \geq 0.75$): Automated processing
- *Medium confidence* ($0.40 \leq p < 0.75$): Human review queue
- *Low confidence* ($p < 0.40$): Discard or additional feature analysis

Recommendation 3 — Continuous Retraining: Given the dynamic nature of social media language, periodic retraining (monthly or event-triggered) is recommended to maintain performance as linguistic patterns evolve.

Recommendation 4 — BERT Upgrade Path: For use cases requiring higher recall on ambiguous or metaphorical tweets (e.g., brand reputation monitoring), upgrading to a fine-tuned `distilbert-base-uncased` or `roberta-base` model is expected to yield F1 improvements of 5–10 percentage points.

10 Methodological Limitations

10.1 Data Limitations

- The dataset originates from Kaggle and represents a specific historical snapshot. Linguistic patterns on social media shift over time, reducing the model's out-of-distribution performance.
- Inter-annotator agreement (IAA) is unknown; label noise is probable, as evidenced by the 110 exact duplicate tweets with inconsistent labels in the raw data.

- The corpus is exclusively English. Disaster events reported in other languages are not covered.
- Geographic and temporal biases are likely present but unquantifiable without metadata.

10.2 Modelling Limitations

- An F1-score of 0.77 implies that approximately 23% of tweets are misclassified. In high-stakes operational contexts, this error rate must be explicitly managed.
- Bag-of-words representations cannot capture contextual nuance, irony, sarcasm, or metaphorical language—the primary source of errors in this project.
- The model is calibrated on the Kaggle distribution. Real-world Twitter data will exhibit distribution shift, particularly around novel disaster types not represented in the training set.
- No temporal modelling is performed; tweet timestamps, trending patterns, or geolocation data are not utilised.

11 Conclusion

This project demonstrates that for binary classification of disaster-related tweets, a **systematically validated, interpretable, and computationally efficient model** is achievable with classical NLP methods—without requiring the overhead of deep learning architectures.

11.1 Three Central Findings

1. **Simplicity outperforms complexity—in context.**
   For short, signal-rich tweets with domain-specific vocabulary, sparse token features combined with linear classifiers are demonstrably optimal. This finding is non-trivial: it challenges the prevailing trend toward architectural complexity and demonstrates that data-driven model selection is indispensable.
2. **Systematic evaluation is necessary.**
   Testing 49 model–vectoriser combinations via stratified cross-validation reveals that the performance landscape is not intuitive. Models and features that appear theoretically superior (deep embeddings, XGBoost) are consistently outperformed by simpler baselines. Without systematic evaluation, suboptimal choices would have been made.
3. **Explainability and performance are not in conflict.**
   The best model (Logistic Regression) is also the most interpretable: every prediction can be traced to specific features with quantified contributions. This is a critical advantage for stakeholders who must justify automated decisions in safety-critical or regulatory contexts.

11.2 Path Forward

For future work, the following directions offer the highest expected return on investment:

- **Transformer-based fine-tuning:** Fine-tuning `roberta-base` or `twitter-roberta-base` is expected to yield F1 ≥ 0.83, addressing the core limitation of metaphorical language misclassification.
- **Ensemble methods:** Combining the current LogReg baseline with a fine-tuned transformer via probability averaging could capture both the speed advantage and the contextual modelling benefit.
- **Real-time streaming integration:** The current model is inference-ready. Integration into a Kafka or AWS Kinesis stream processing pipeline would enable genuine real-time disaster monitoring.
- **Multilingual extension:** Extending the pipeline with language detection and language-specific preprocessing would substantially improve coverage for global disaster events.

A Technical Specifications

Table 9: Software Environment

Component	Version
Python	3.12.12
NumPy	2.0.2
Pandas	2.2.2
scikit-learn	1.6.1
XGBoost	3.2.0
Gensim	4.4.0
NLTK	3.9.1
Matplotlib	3.10.0
Seaborn	0.13.2
Plotly	5.24.1
PyTorch	2.10.0 (CPU)

B Pipeline Artefacts

Table 10: Generated Pipeline Artefacts

Artefact	Location	Description
`train_preprocessed.csv`	`data/processed/`	Normalised training set (7,503
`test_preprocessed.csv`	`data/processed/`	Normalised holdout set (3,243
`submission.csv`	`data/results/`	Kaggle submission with final p
`baseline_summary.csv`	`data/metrics/`	All 4 baseline system metrics
Vectoriser plots	`reports/results/plots/vectorizer/`	PCA, t-SNE, TF-IDF term plo
Model plots	`reports/results/plots/models/`	CV ranking, heatmaps, ROC c
CV results CSV	`reports/results/metrics/cv/`	All 49 cross-validation results

C Preprocessing Pipeline Trace

Table 11: Example Preprocessing Pipeline Trace

Stage	Output Example
`text` (raw)	*"Latest: USA: Huge sinkhole swallows up Brooklyn intersection http://t.co/...*
`tokens`	*['latest', 'usa', 'huge', 'sinkhole', 'swallows', 'up', 'brooklyn', ...]*
`tokens_no_stopword`	*['latest', 'usa', 'huge', 'sinkhole', 'swallows', 'brooklyn', 'url']*
`tokens_lemmatized`	*['latest', 'usa', 'huge', 'sinkhole', 'swallow', 'brooklyn', 'url']*
`tokens_cleaned`	*['latest', 'usa', 'huge', 'sinkhole', 'swallow', 'brooklyn', 'url']*
`clean_text`	*"latest usa huge sinkhole swallow brooklyn url"*