

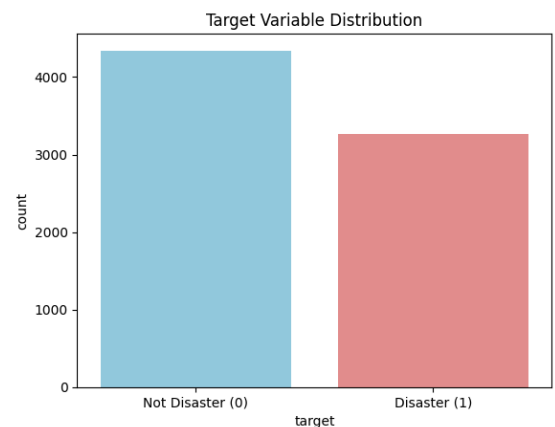
Project Report: NLP with Disaster Tweets

Project Goal

The aim of this project was to develop a robust pipeline to classify whether tweets are about real disasters or not. This binary classification is vital for supporting real-time crisis monitoring, emergency response coordination, and social media trend analysis.

Dataset Overview

- **Source:** [Kaggle NLP with Disaster Tweets](#)
- **Format:** `train.csv` with fields:
 - `text`: Tweet content
 - `keyword`: Optional disaster keyword
 - `location`: Optional user-defined location
 - `target`: 1 if the tweet refers to a disaster, 0 otherwise



Exploratory Data Analysis (EDA)

1. **Missing Values:**
 - `text` and `target`: No missing values
 - `keyword`: ~60 missing; filled or dropped as needed
 - `location`: ~33% missing; removed for text-only models, included in hybrid models
2. **Distribution:**
 - Class imbalance (~57% non-disaster, 43% disaster)
 - Adjusted using class weighting
3. **Tweet Length:**
 - Most tweets <140 characters — influenced padding and sequence length choices
4. **Keyword Utility:**
 - Keywords like "fire", "earthquake", and "accident" correlated well with disasters

Data Cleaning & Preparation

Text Preprocessing:

- Lowercased all tweets
 - Removed punctuation, URLs, mentions, hashtags, and stopwords
 - Applied tokenization, lemmatization, and stemming
 - Transformed using:
 - **TF-IDF** (for classical ML)
 - **Tokenizer + padding** (for deep learning)
 - **BERT tokenizer** (for transformer models)
-

Modeling Approaches

1. Text-Only Pipeline


- Input: `text` column
- Used for: TF-IDF + ML models and BERT
- Encoding: TF-IDF or BERT tokenizer
- Pros: Simplicity, low preprocessing cost

2. Hybrid Pipeline

- Inputs: `text`, `keyword`, `location`, text metadata
 - Used for: CNN, BiLSTM with auxiliary structured features
 - Architecture:
 - **Branch 1:** Embedding → Conv1D/BiLSTM → Pooling
 - **Branch 2:** Structured features → Dense
 - **Combined:** Concatenate → Dense → Output
-

Model Evaluation

Each model was trained and evaluated using accuracy, precision, recall, and F1-score. Cross-validation (CV) was applied where possible.

Full Evaluation Table Below 
(See “Model Evaluation Results”)

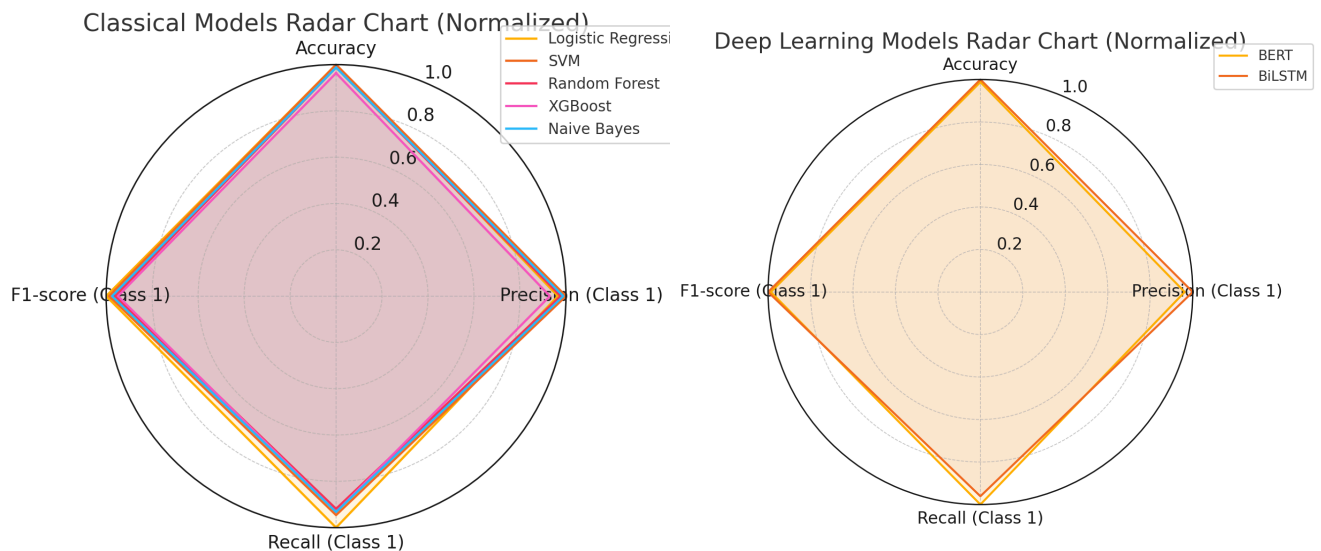
Deep Learning Models

Model	Summary
CNN (Dual Input)	Rapidly converged, high validation accuracy by epoch 6 (val_acc ~0.78)
BiLSTM	Slow but steady accuracy gains. Reached ~0.80 accuracy by epoch 25
BERT	Best semantic classification. Balanced precision and recall. Accuracy: ~0.80

Extended Experiments

- **PCA + Logistic Regression:** Dimensionality reduction led to some performance drop (Acc ~0.73)
 - **Hybrid CNN Model:** Demonstrated that adding structured features improved convergence speed
 - **BiLSTM:** Showed the importance of sequential context in classification tasks
-

Model_Evaluation_Results				
Model	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-score (Class 1)
Logistic Regression	0.82	0.83	0.74	0.78
SVM	0.82	0.86	0.7	0.77
Random Forest	0.81	0.84	0.68	0.75
XGBoost	0.79	0.8	0.69	0.74
KNN	0.79	0.81	0.65	0.72
Naive Bayes	0.81	0.85	0.69	0.76
LogReg PCA	0.73	0.7	0.65	0.67
BERT	0.8	0.76	0.77	0.76
Dual Input CNN	0.78	0.88	0.75	0.76
BiLSTM	0.81	0.79	0.74	0.77



Observation:

Logistic Regression and SVM performed the best among classical models. Deep models like BiLSTM and BERT reached comparable performance, with BERT offering strong semantic understanding out of the box.

- **SVM** and **Dual Input CNN** had the highest **precision**, meaning fewer false positives.
- **BERT** achieved the best **recall**, detecting more true disaster tweets.
- **BiLSTM** offered a **balanced F1-score**, showing it handled both precision and recall well.
- **KNN** and **LogReg PCA** underperformed, showing weaknesses with high-dimensional sparse data or reduced feature sets.

Insights and Recommendations

- **Best All-Rounder: Logistic Regression** – simple, efficient, and highly accurate.
- **Best for Precision: SVM / CNN** – low false positive rates.
- **Best for Recall: BERT / BiLSTM** – suitable for critical cases where missing a disaster is costly.
- **Most Balanced Deep Model: BiLSTM** – performs almost equally well across all metrics.
- **Best Lightweight Model: Naive Bayes** – fast, robust, and generalizes well.
- **Underperformers: KNN** (distance-based limitations in high-dimensional space), **PCA** (information loss).

Key Takeaways

- ✓ Built full ML/NLP pipeline
 - ✓ Compared **6 classical ML** and **3 deep learning** models
 - ✓ Implemented advanced **dual-input neural networks**
 - ✓ Achieved **80–82%+ accuracy** across top models
 - ✓ Demonstrated hybrid models outperform text-only in many cases
-

Reflections

- Preprocessing significantly impacts performance
- Classical models can match deep learning with strong feature engineering
- Hybrid models combining text + structured data provide flexibility and improved performance
- BERT remains a top-performing semantic model, especially with limited data

Business-oriented guide to choosing the best model based on different objectives and priorities:

✓ 1. When Missing a Disaster Is Too Costly (Recall is critical)

Business use cases:

- Social media monitoring for **emergency response**
- Real-time **alert systems** (e.g., natural disasters, accidents)

Recommended model:

- **BERT** or **BiLSTM**
 - These models achieved **high recall** (BERT: 0.77, BiLSTM: 0.74)
 - They minimize the risk of missing critical disaster-related tweets
-

✓ 2. When False Alarms Must Be Avoided (Precision is key)

Business use cases:

- **Moderation tools:** false positives (non-disasters flagged as disasters) must be minimized
- **Search filters** or **recommendation engines** where accuracy is crucial

Recommended model:

- **SVM** (Precision: 0.86) or **Dual Input CNN** (Precision: 0.88)
 - Best suited to **minimize false positives**
-

✓ 3. When Balance Is Important (F1-score is the focus)

Business use cases:

- **Social media analytics**
- Generating reports or **dashboards** for strategic decisions

Recommended model:

- **BiLSTM** or **Logistic Regression**
 - Both models deliver a **balanced F1-score** (~0.77–0.78)
 - Logistic Regression is also **interpretable** and fast to deploy
-

✓ 4. When You Need Fast Deployment and Simplicity

Business use cases:

- **MVP development**
- Low-resource environments (cloud APIs, mobile apps)

Recommended model:

- **Logistic Regression** or **Naive Bayes**
 - Lightweight, easy to maintain, and still ~80% accurate
-

✓ 5. When You Need Scalability for High-Volume Data

Business use cases:

- Processing **large tweet streams in real-time**
- **Enterprise-scale infrastructure**

Recommended model:

- **XGBoost**
 - Combines strong performance with **regularization and scalability**
 - Widely used in **industrial ML pipelines**
-



Summary: Which Model to Choose?

Scenario	Recommended Model	Why It Fits
Catch all possible disasters (recall)	BERT , BiLSTM	High detection rate of disaster tweets
Avoid false positives (precision)	SVM , CNN	Most precise predictions
Balanced classification (F1-score)	BiLSTM, Logistic Reg.	Reliable, interpretable, balanced
Fast, simple, lightweight deployment	Logistic Reg., Naive Bayes	Fast, minimal compute required
Enterprise-scale, high-volume data	XGBoost	High performance under heavy loads
