**Submitted by:**

Sristi Singh

(Email Id: s.sristi90@gmail.com)

**Domain:** Artificial Intelligence

**Batch:** Artificial Intelligence, September 2025

Abstract

The primary goal of this project is to develop and deploy a system capable of accurately classifying the sentiment (Positive or Negative) expressed in social media text, specifically using Twitter (now X) data. The challenge lies in dealing with noisy, unstructured, and informal text while creating a reliable and highly available prediction service.

Social Media Sentiment Analysis Project

Report: Logistic Regression Baseline

# Social Media Sentiment Analysis Project

# Report: Logistic Regression Baseline

## 1. Problem Definition: The Challenge

The primary goal of this project is to develop and deploy a system capable of accurately classifying the sentiment (Positive or Negative) expressed in social media text, specifically using Twitter (now X) data. The challenge lies in dealing with noisy, unstructured, and informal text while creating a reliable and highly available prediction service.

### 1.1. Sub-Problem 1: Sentiment Classification (Binary)

This sub-problem focuses on training a machine learning model to classify a given text snippet into one of two categories:

* **Task:** Binary Classification (Positive vs. Negative).
* **Goal:** Create a performant and lightweight baseline model for fast inference.

### 1.2. Sub-Problem 2: Deployment and Real-time Inference

This sub-problem addresses the need to expose the trained model to end-users via an interactive, real-time application.

* **Task:** Creating a user-friendly web interface (Dashboard) for instant sentiment prediction.
* **Goal:** Ensure fast preprocessing and model loading for a seamless user experience.

## 2. Data Collection & Source

The model was trained on the **Sentiment140 dataset**, a large, publicly available corpus of tweets.

* **Source:** Sentiment140 Dataset (often used for academic benchmarks).
* **Size:** Contains approximately 1.6 million tweets.
* **Structure:** Each record includes a target label (0 for Negative, 4 for Positive) and the original tweet text.

## 3. Data Pre-processing and Engineering

Robust preprocessing is critical to clean noisy social media data and transform text into numerical features suitable for machine learning.

### 3.1. Text Cleaning & Standardization

The cleaning steps ensure text is standardized and free of non-linguistic noise common in tweets:

* **Lowercasing:** All text is converted to lowercase to ensure consistency.
* **Noise Removal:** Regular expressions are used to remove common social media artifacts:
  + URLs (http://, https://)
  + User mentions (@username)
  + Hashtags (#topic)
  + Punctuation and numerical digits.

### 3.2. Linguistic Processing (Tokenization and Normalization)

Linguistic processing reduces vocabulary size and standardizes word forms:

* **Stopword Removal:** Common, less informative words (e.g., "the," "is," "a") are removed using the NLTK English stopwords list.
* **Lemmatization:** Words are reduced to their base or root form (e.g., "running" $\rightarrow$ "run," "better" $\rightarrow$ "good") using the NLTK WordNet Lemmatizer.

### 3.3. Feature Engineering: TF-IDF Vectorization (Crucial Step)

The cleaned text must be converted into a numerical vector format. The **Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer** was used:

* **Process:** TF-IDF assigns a weight to each word based on its frequency within a tweet (TF) relative to its frequency across the entire dataset (IDF). This highlights words that are unique and discriminatory (i.e., strong sentiment indicators).
* **Hyperparameter:** The vectorizer was configured with a max\_features=50000 to limit the vocabulary size, balancing performance and memory usage.

## 4. Machine Learning Model Architecture

### 4.1. Model Summary

A **Logistic Regression** model was chosen as the baseline for this binary classification task due to its effectiveness, speed, and interpretability in high-dimensional text classification problems.

* **Algorithm:** Logistic Regression (a linear classifier).
* **Hyperparameters:** The model was configured with max\_iter=1000 and solver='liblinear', an efficient solver suitable for large linear classification problems.
* **Training:** The model was trained on $80\%$ of the data, with $20\%$ reserved for testing, ensuring reliable validation.

### 4.2. Two-Stage Fine-Tuning (Skin Model)

*(Adaptation of the original outline section)*

While the Logistic Regression model does not require two-stage fine-tuning, the overall pipeline utilizes a two-stage approach for robustness:

1. **Stage 1: Feature Extraction:** The TF-IDF Vectorizer is first **fitted** on the training data ($\text{X}\_{\text{train}}$) to learn the vocabulary and word weights.
2. **Stage 2: Model Training:** The transformed numerical data ($\text{X}\_{\text{train\\_tfidf}}$) is then fed into the Logistic Regression classifier. This ensures the model and the feature space are learned separately and consistently.

## 5. Model Performance and Interpretability

### 5.1. Evaluation Metrics

The model's performance was evaluated on the held-out test set:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Result** | **Significance** |
| **Accuracy** | $\approx 78\%$ | Overall percentage of correctly classified tweets. |
| **Precision/Recall/F1** | High for both Negative and Positive classes. | Shows consistent performance across both sentiment labels, indicating the model handles the balanced dataset well. |
| **Confusion Matrix** | Clearly defines False Positive and False Negative rates. | Allows for fine-tuning based on whether minimizing missed negative posts or false alarms is prioritized. |

### 5.2. Interpretability: Feature Importance

*(Adaptation of the original outline section from Grad-CAM to Linear Model Interpretability)*

For a linear model like Logistic Regression, interpretability is achieved by examining the **model coefficients**.

* **Analysis:** High positive coefficients correspond to words that strongly predict **Positive** sentiment (e.g., "love," "great," "happy"). High negative coefficients correspond to words that strongly predict **Negative** sentiment (e.g., "fail," "sad," "bad").
* **Benefit:** This provides transparency, allowing developers to verify that the model is making predictions based on linguistically sensible terms.

## 6. Implementation and Analytics (The Cloud Dashboard)

The core purpose of the project is realized through the deployment of the model via a Streamlit dashboard (app.py).

### 6.1. Architecture and Data Flow

The application utilizes the following components:

* **Persistence:** The trained log\_reg\_model and tfidf\_vectorizer are saved as .pkl files using joblib.
* **Deployment:** The app.py script, utilizing the streamlit framework, loads the .pkl assets upon startup.
* **Inference Pipeline:** User input $\rightarrow$ preprocess\_text\_for\_prediction function $\rightarrow$ vectorizer.transform() $\rightarrow$ model.predict() $\rightarrow$ Result displayed.

### 6.2. Operational Analytics

A production deployment of this dashboard would enable various operational analytics:

* **Prediction Volume:** Tracking the number of sentiment requests over time.
* **Input Monitoring:** Analyzing trends in the length, vocabulary, and cleanliness of incoming user text to identify changes in user behavior or platform noise.
* **Latency:** Monitoring the delay between button click and result display to ensure the model load and inference remain fast (typically milliseconds).

## 7. Conclusion and Future Work

### 7.1. Conclusion

This project successfully developed and deployed a fully functional social media sentiment analysis tool. The Logistic Regression model, trained on 1.6 million tweets, provides a highly efficient baseline with an estimated accuracy of $\approx 78\%$. By serializing the model and vectorizer with joblib, the Streamlit application ensures the model is instantly available for real-time inference.

### 7.2. Future Enhancements

1. **Deep Learning Upgrade:** Replace the Logistic Regression baseline with an advanced model like a **Bidirectional LSTM** or a **Transformer** (e.g., BERT) to capture long-range dependencies and contextual sentiment, potentially increasing accuracy into the $85\%+$ range.
2. **Emotion Detection:** Expand the project from binary sentiment to a multi-class emotion classification (e.g., Happy, Sad, Angry, Fear, Surprise).
3. **Cross-Platform Adaptation:** Introduce a pipeline that specifically handles platform-specific features like Twitter emojis, Reddit formatting, or Discord markdown, treating them as valuable features rather than noise.