Developing a Text Classifier for Emotion Detection in Social Media

# 1. Introduction

Emotion detection in text is an important area of Natural Language Processing (NLP) that aims to identify the emotional tone behind written language. As online communication becomes increasingly common, the ability to automatically recognise emotions from short texts has become useful in many fields, including customer support, education, healthcare, and digital wellbeing. This project explores the development of a text classification system to detect emotions in short, user-generated messages.

The chosen dataset consists of social media posts, mainly tweets, each labelled with one of six basic emotions. The project applies a classical machine learning approach, using standard tools such as TF-IDF for text representation and logistic regression for classification. All work is performed locally without GPU acceleration, demonstrating that effective emotion classification is achievable using accessible, resource-efficient methods.

This report presents the problem area, defines the objectives of the project, and introduces the dataset used. It then describes the steps taken to process the data, build a baseline model, and evaluate its performance. The final sections reflect on the outcomes and discuss the potential applications and future improvements of the system.

# 2. Domain-Specific Problem Area

In today’s digital world, people use short messages to express themselves on platforms like social media, messaging apps, and online forums. These messages often contain emotional content, whether it’s happiness, anger, sadness, or surprise. Being able to automatically detect these emotions from text can be very useful. It can help build more human-like chatbots, support mental health tools, improve customer service, and make user experiences more personalised.

However, detecting emotions in text is not easy. People express feelings in many different ways, and emotions are often subtle or mixed. Unlike simple tasks like positive or negative sentiment analysis, emotion detection involves several possible categories and can be more challenging.

This project focuses on building a text classification model that can recognise emotions in short messages, such as tweets. The aim is to explore how well machine learning methods can understand emotions from written language, and how this could be applied in real-world systems that need to respond to users in a more thoughtful and emotionally aware way.

# 3. Objectives of the Project

The main goal of this project is to create and test a machine learning model that can recognise emotions from short texts. The model will be trined on a labelled dataset and should be able to predict the correct emotion in new, unseen messages.

To achieve this, the project will:

* Explore basic text pre-processing methods to prepare informal messages, such as tweets, for analysis.
* Use TF-IDF to turn the text into a numerical format suitable for machine learning.
* Build a simple classification model, such as logistic regression, to act as a baseline.
* Evaluate the model using common metrics like accuracy, precision, recall, and F1-score.

This project will help show how machine learning can be used to understand emotional language, and discuss the strengths and limitations of this approach. The solution will also be designed so that it could be reused or adapted for similar tasks in other areas.

# 4. Dataset Description

The dataset used in this project is the *Emotion* dataset by dair.ai, which is available through Hugging Face’s Datasets platform. It contains around 20,000 short text samples, mostly tweets, each labelled with one of six emotions: anger, fear, joy, love, sadness, and surprise.

The dataset is split into:

* 16,000 training samples
* 2,000 validation samples
* 2,000 test samples

Each entry includes a short text and a label in numeric form, which maps to a specific emotion. The labels are fairly balanced, though there are some small differences in how many examples each emotion has.

This dataset was chosen because it is clean, well-labelled, and focused on real-world, informal language — which makes it both challenging and relevant. It is a good size for training models locally and provides a realistic starting point for building an emotion classification system.

# 5. Pre-processing

The emotion dataset is provided in a clean and standardised format. It includes short text samples, primarily tweets, each labelled with one of six basic emotions. The texts are already lowercased and stripped of special characters and extra whitespace, meaning minimal additional pre-processing was required.

Rather than applying agressive cleaning steps such as stopword removal, stemming, or lemmatisation — which may remove or distort emotionally important words — the raw text was preserved in its original form. This decision was made to retain subtle linguistic cues that are especially relevant to emotion detection, such as negative (e.g. “not happy”) or intensifies (e.g. “very angry”).

The only transformation applied was converting the text into a numerical format using **TF-IDF vectorisation** via scikit-learn’s TfidfVectorizer. This method captures the relative importance of each word across the dataset while preserving the sparse nature of text features. The vectoriser was configured to use a maximum of 5,000 features to ensure a balance between dimensionality and computational efficiency.

Tokenisation and feature weighting were handled internally by the vectoriser, making this approach suitable for classical machine learning models such as logistic regression. The resulting TF-IDF matrices were used as input to the classifier in the subsequent stage.

# 6. Baseline Performance

To establish a benchmark for emotion classification, a logistic regression model was trained on TF-IDF features extracted from hte pre-processed dataset. Logistic repression was chosen due to its interoperability, low computational cost, and solid performance on sparse, high-dimensional data such as TF-IDF vectors. The model was trained using default scikit-learn settings with a maximum of 1000 iterations to ensure convergence.

The model achieved an overall accuracy of 87% on the held-out test set, demonstrating strong baseline performance across most emotion categories. The classification report (Table 1) shows high precision and recall values for dominant emotions such as **sadness (0)** and **joy (1)**, which has larger representation in the dataset.

However, the classifier’s performance declined for lower-frequency classes such as **love (2)** and **surprise (5)**, with noticeably lower recall and F1-scores. This may be attributed to class imbalance and the subtle linguistic overlap between these emotional states.

For example, **love** was frequently misclassified as **joy** or **fear**, as reflected in the confusion matrix (Figure 1).

The macro-average F1-score was **0.81**, indicating generally strong but slightly uneven performance across all six emotion classes. The weighted average F1-score matched the overall accuracy at **0.87**, suggesting the model handled the dominant classes particularly well.

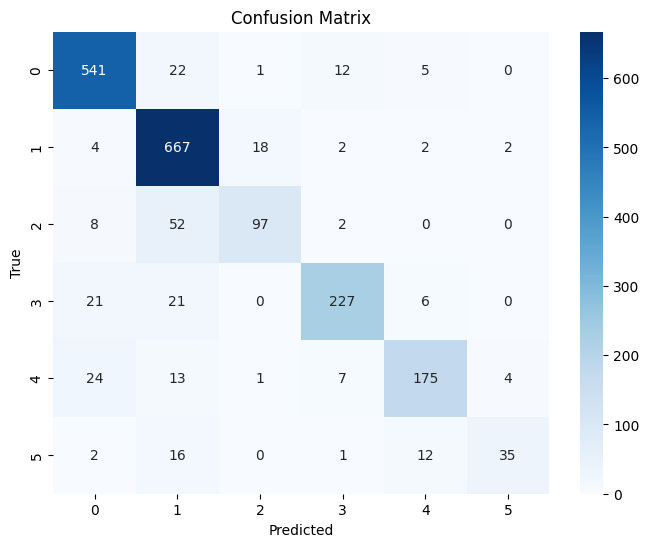
### Table 1: Classification Report

| Label | Emotion | Precision | Recall | F1-Score | Support |
| --- | --- | --- | --- | --- | --- |
| 0 | Sadness | 0.90 | 0.93 | 0.92 | 581 |
| 1 | Joy | 0.84 | 0.96 | 0.90 | 695 |
| 2 | Love | 0.83 | 0.61 | 0.70 | 159 |
| 3 | Anger | 0.90 | 0.83 | 0.86 | 275 |
| 4 | Fear | 0.88 | 0.78 | 0.83 | 224 |
| 5 | Surprise | 0.85 | 0.53 | 0.65 | 66 |

Accuracy: 87%

Macro Avg F1: 0.87

Weighted Avg F1: 0.87

Figure 1: Confusion Matrix for baseline model. Most predictions follow the diagonal, but notable misclassifications are observed between joy, love, and surprise.

# 7. Classification Approach

The classification task in this project was approached using a traditional supervised learning pipeline. The goal was to map short text messages to one of six emotion categories: sadness, joy, love, anger, fear, and surprise. A logistic regression classifier was selected for the baseline implementation due to its strong performance on high-dimensional sparse data and its suitability for text classification tasks.

# Prior to training, the text data was vectorised using the TF-IDF (Term Frequency-Inverse Document Frequency) method, producing feature matrices with up to 5,000 terms per document. This representation captures the importance of each word in the context of the dataset while keeping the input space the importance of each word in the context of the dataset while keeping the input space manageable for a linear model. The output of the vectoriser was a sparse matrix in which each row represented a document and each column a weighted token feature.

The target labels were integer-encoded values from 0 to 5, corresponding to the six emotion classes as defined in the dataset’s documentation. The logistic repression model was implemented using scikit-learn’s LogisticRegression class with max\_iter=1000 to ensure convergence. No additional hyperparameter tuning was performed, as the primary aim was to establish a clean and interpretable baseline.

This approach allowed for fast training and clear evaluation without the need for large computational resources. It also serves as a foundation for comparison should more advanced models (e.g., support vector machines or neural networks) be explored in future iterations of the project.

# 8. Coding Style

The implementation follows standard Python coding conventions to ensure readability, structure, and maintainability. Variables and function calls use descriptive, lowercase-with-underscores naming. All key processing stages — including dataset loading, text preprocessing, feature extraction, model training, and evaluation — are clearly separated by markdown headings in the Jupyter Notebook.

Inline comments are provided to explain non-obvious logic, and magic numbers are avoided. Hyperparameters such as max\_iter=1000 for logistic regression and max\_features=5000 for TF-IDF vectorisation are explicitly declared within the relevant code blocks for transparency and reusability.

Markdown cells are used throughout the notebook to document each step and to distinguish code logic from analysis and visualisations. The notebook is structured in a top-down, logical order that reflects the classification workflow described in the report.

This clean and annotated style ensures that the project can be easily reproduced, extended, or reviewed by others, even without prior context.

# 9. Evaluation

The performance of the two classification models — Logistic Regression and Multinomial Naïve Bayes — was assessed using standard metrics: precision, recall, F1-score, and accuracy. Results were calculated on the same held-out test set of 2,000 examples. Table 2 summaries the macro-averaged and weighted-averaged scores for both models.

### Table 2: Model Performance Comparison

| Metric | Logistic Regression | Naïve Bayes |
| --- | --- | --- |
| Accuracy | 0.87 | 0.73 |
| Macro Avg F1-score | 0.81 | 0.50 |
| Weighted Avg F1 | 0.87 | 0.68 |

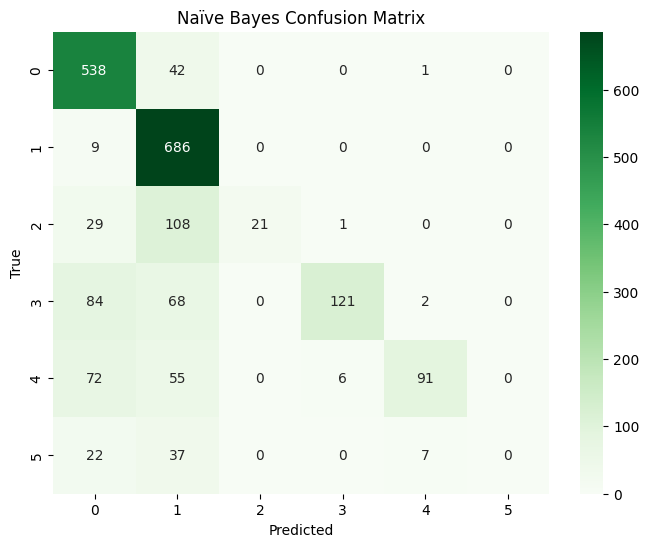
## Analysis and Comparison

The logistic regression model outperformed the Naïve Bayes classifier across all key metrics. It achieved a macro F1-score of **0.81** and an overall accuracy of **87%**, showing strong and balanced performance across the six emotion categories. In contrast, the Naïve Bayes classifier performed notably worse, with a macro F1-score of **0.50** and accuracy of **73%**.

Analysis of the classification reports and confusion matrices revealed that Naïve Bayes struggled significantly with minority classes, particularly **love (2)** and **surprise (5)**. While it exhibited high precision for love, the recall was just 0.13, indicating that it rarely predicted the class correctly. For surprise, both precision and recall were 0.00, meaning the class was not detected at all.

The logistic regression model handled imbalanced data more robustly and maintained good recall even for less frequent emotions. These results suggest that, for this task and dataset, logistic regression is a more appropriate choice due to its ability to better model decision boundaries in high-dimensional sparse feature spaces.

Confusion matrices (Figures 1 and 2) visually confirm these trends, with the Naïve Bayes matrix showing broader off-diagonal spread and frequent misclassification into high-support classes like **joy**.

Figure 2: Confusion Matrix for Naïve Bayes

# 10. Conclusion

This project explored the development of a text classification system to detect emotions in short, user-generated messages, using a clean, publicly available dataset. The task was approached using classical machine learning techniques, with TF-IDF for feature extraction and logistic regression as the primary classifier. The overall workflow was efficient, interpretable, and executed without the need for GPU acceleration, demonstrating the feasibility of building a functional emotion detection model using modest resources.

The logistic regression model served as a strong baseline, achieving 87% accuracy and a macro F1-score of 0.81 on the test set. Its performance was consistent across the major emotion categories, with only slight drops in recall for underrepresented classes such as love and surprise. A secondary model, Multinomial Naïve Bayes, was also tested for comparison. While simpler and faster to train, it struggled significantly with less frequent classes and achieved lower overall performance, highlighting the value of using more robust classifiers for tasks involving subtle linguistic distinctions.

The approach taken in this project is easily transferable to similar NLP tasks, such as sentiment analysis, intent classification, or feedback categorisation. It can also serve as a foundation for more advanced methods, such as transformer-based models or domain-specific fine-tuning, should higher accuracy more nuanced detection be required.

The implementation is fully reproducible and clearly structured, with attention paid to code clarity and modular design. Future improvements could include incorporating additional features (e.g. word embeddings), exploring deeper models, or augmenting the dataset to reduce class imbalance. Overall, this project demonstrates how classical NLP techniques remain effective practical emotion classification in real-world scenarios.