# Report

Davide Straziota

30th May 2025

## 1 Introduction

This work addresses the problem of sentiment classification on a dataset of tweets directed at U.S. airlines. The data consist of labeled training and test sets, where each tweet is annotated as **Positive**, **Neutral**, or **Negative**. Our goal is to develop and evaluate machine learning models that accurately predict sentiment labels by leveraging natural language processing (NLP) techniques and modern classification algorithms.

The overall workflow comprises the following stages:

- **Data Preprocessing.** We normalize all text to lowercase and remove noisy elements such as URLs, user mentions, hashtags, and non-alphanumeric characters.
- **Tokenization and Embedding.** Tweets are tokenized using NLTK. We eliminate standard stopwords and convert tokens into continuous vector representations via pre-trained word embeddings. Tokens absent from the embedding vocabulary are skipped, and empty tweets are represented by zero vectors.
- **Exploratory Class Analysis.** We examine the distribution of sentiment classes in both training and test sets. The following figures illustrate that the dataset is highly imbalanced: negative tweets constitute the majority class, while positive and neutral tweets are underrepresented. Addressing this imbalance is crucial for robust classification performance, and it motivates the subsequent methodological choices.

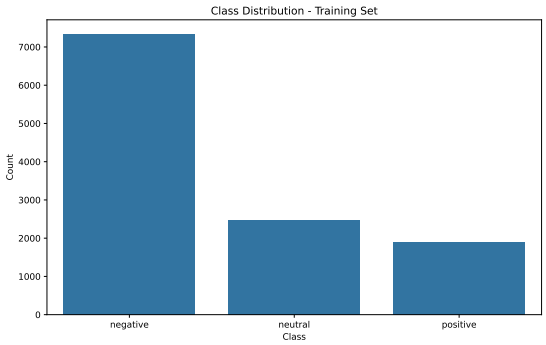


Figure 1: Class distribution in the training set.

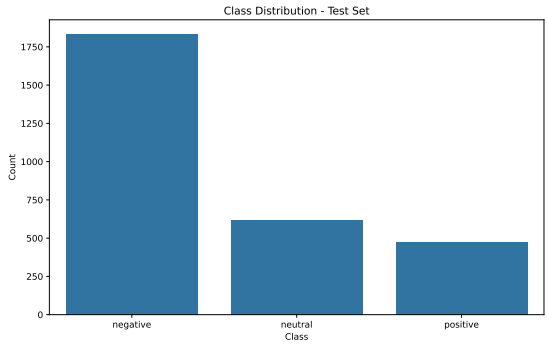


Figure 2: Class distribution in the test set.

- **Model Selection.** We implemented and compared several classification algorithms: Logistic Regression, Multinomial Naive Bayes, Linear Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and XGBoost. Each model resides in its own script for modular experimentation and received basic hyperparameter tuning (including varying MLP architectures). Performance was evaluated using accuracy, precision, recall, and F1-score on the training, validation, and test sets. XGBoost emerged as the best-performing method, and we focus on its results and diagnostics (e.g., confusion matrices) in the following sections.
- **Imbalance Mitigation Strategies.** To counteract class imbalance, we explored three approaches:

Table 1: Performances of the models on the test set

| Model | Accuracy | Precision | recall | F1 score |
|---|---|---|---|---|
| Logistic Regression | 0.7445 | 0.7324 | 0.7445 | 0.7238 |
| Linear SVM | 0.7425 | 0.7318 | 0.7425 | 0.7196 |
| Naive Bayes | 0.6680 | 0.6736 | 0.6680 | 0.6629 |
| MLP | 0.7517 | 0.7430 | 0.7517 | 0.7397 |
| XGBoost | 0.7579 | 0.7477 | 0.7579 | 0.7446 |

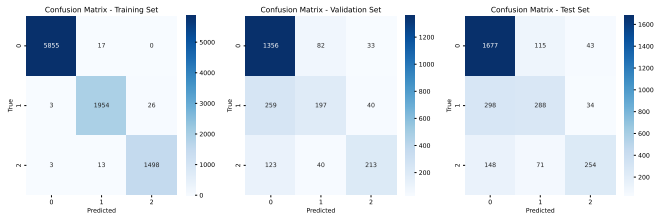


Figure 3: Confusion Matrices XGBoost

1. *Data Augmentation:* Synthetic tweets are generated by replacing words with synonyms, randomly deleting or swapping tokens (see *SendAI_XGBoost_augmented.py*).
2. *Weighted Loss with Augmentation:* The same augmentation pipeline is used in conjunction with a class-weighted loss proportional to class frequencies (see *SendAI_XGBoost_augmented_Balancing.py*).
3. *Resampling Techniques:* We apply oversampling and undersampling methods from the *imbalanced_learn* library, including SMOTE, ADASYN, SMOTEENN, SMOTETomek, and RandomUnderSampler (see *SendAI_XGBoost_imbalance.py*).

The following figure shows the effect of augmentation on class balance in the training set.

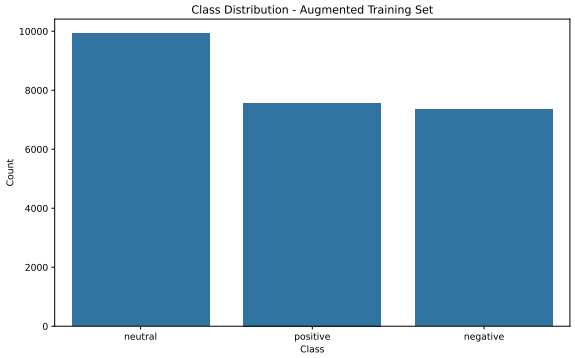


Figure 4: Training set class distribution after data augmentation.

The results of these strategies are compared in the following tabular, and as an example the confusion matrix of the augmented case with weighted loss i reported. For the imbalanced learning techniques I report the results for the best model.

## 2 Conclusions and Future Work

Our experiments demonstrate that careful handling of class imbalance is key to achieving high accuracy and balanced performance across sentiment categories. XGBoost, combined with resampling techniques, yielded the best overall

Table 2: Performances of the models for unbalanced data

| Model | Accuracy | Precision | recall | F1 score |
|---|---|---|---|---|
| XGBoost augmented | 0.7336 | 0.7486 | 0.7336 | 0.7391 |
| XGBoost augmented and weighted | 0.7647 | 0.7550 | 0.7647 | 0.7502 |
| XGBoost imbalanced learning (best) | 0.7462 | 0.7442 | 0.7462 | 0.7445 |

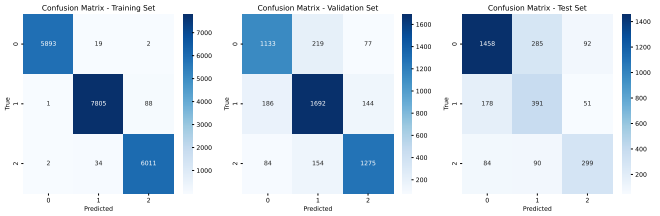


Figure 5: Confusion matrices for XGBoost for the augmented dataset.

results.

For future work, we propose the following enhancements:

- **OOV Token Handling.** Instead of discarding out-of-vocabulary words and empty tweets, we will investigate fallback embedding strategies and explicit flags for missing content to retain more information.
- **Refined Preprocessing.** Emoticons and emojis convey strong sentiment cues. We plan to design a preprocessing pipeline that preserves or encodes these elements rather than removing them.
- **Advanced Hyperparameter Optimization.** A systematic search (e.g., Bayesian optimization) over model architecture and resampling parameters could further improve performance.
- **Combined Augmentation and Resampling.** Although the interplay between augmentation and resampling may risk overfitting, exploring their synergy could uncover configurations that boost generalization.

Overall, this study underscores the importance of both data-centric and model-centric techniques for robust sentiment classification in imbalanced settings.