Sentiment Classification of Airline Tweets

Vojtech Matulik

May 15, 2025

Contents

1 Introduction

Airline customer feedback on social media platforms, particularly Twitter, offers a rich source of information to understand passenger experiences and identify service issues. Automatically analyzing this feedback can help airlines respond more efficiently and improve customer satisfaction. One key task in this area is sentiment analysis, which is assessing whether a tweet expresses negative, neutral, or positive sentiment.

The objective of this project is to develop and evaluate machine learning classifiers capable of predicting the sentiment of tweets directed at or about airline services. Specifically, the goal is to classify each tweet into one of three sentiment categories: *negative*, *neutral*, or *positive*, based on the content of the message. Learning from labeled examples of tweets, the system aims to identify the emotional tone and underlying opinions present in the text.

This task is framed as a supervised learning problem, where models are trained on labeled data to learn patterns associated with each sentiment class. A variety of classifiers are explored, including logistic regression, multinomial Naive Bayes, and support vector machines. Their performance is evaluated using standard classification metrics, such as precision, recall, F1 score, and accuracy.

The implementation of this project is performed solely in Python.

2 Data Preprocessing and Exploratory Data Analysis

For this task, *Crowdflower Twitter Airline Sentiment* dataset from Kaggle is used, which is available from `https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment` in an XLSX format. This dataset contains 14,640 tweets about six US airlines. It contains multiple attributes; many of those are redundant for this task. The only attributes kept for further processing are `text` which contains the Tweet about a particular airline in text format, and `airline_sentiment` that contains information on whether the text is *negative*, *neutral*, or *positive*, based on the content of the message.

The class (sentiment) distribution is unbalanced in the extracted Kaggle dataset: negative tweets constitute the majority – 62 % of the tweets (9178), while neutral and positive tweets make up roughly 21 % and 17 % of the data (3099 and 2363), respectively. This imbalance means that classifiers could be biased towards predicting the negative class, so precautionary steps should be taken in the modeling strategy. The average length of a tweet is 17.5 words.

Before modeling, text cleaning and pre-processing on the tweets is performed. The following steps are applied to each tweet during preprocessing:

- **Lowercasing**: All text is converted to lowercase to ensure consistency (e.g., *Flight* and *flight* are treated the same).
- **Stopword Removal**: Common English stopwords (e.g.,*the*, *is*, *at*) that do not have significant meaning are removed.
- **Lemmatization**: The words are lemmatized using part-of-speech tagging to obtain accurate lemmas (for example, *delayed* might be lemmatized as *delay* depending on the context, and *better* would lemmatize to *good* as an adjective).
- **Contraction Expansion**: Common contractions are expanded to their full forms (e.g., *can't* becomes *cannot*, *it's* becomes *it is*) to standardize the vocabulary.
- **Punctuation & Special Characters**: Punctuation, extra whitespace, and Twitter-specific entities (such as mentions *@user* and URLs) are removed or normalized, since they do not directly contribute to sentiment.

In Python, libraries are mostly imported to handle stop words, lemmatization, etc. For removal of mentions and special characters, regex is used. The following function performs data preprocessing in this project:

```
def cleaning(text):
    text = contractions.fix(text)
    text = re.sub(r'@\w+|http\S+', '', text)
    text = re.sub(r'[^A-Za-z\s]', '', text)
    text = text.lower().strip()
    tokens = word_tokenize(text)
    tokens = [t for t in tokens if t not in stop_words and len(t) > 1]
    tagged_tokens = pos_tag(tokens)
    lemmatized = [
        lemmatizer.lemmatize(w, get_wordnet_pos(pos))
        for w, pos in tagged_tokens
    ]
    return " ".join(lemmatized)
```

Figure 1: Data Cleaning

The cleaned text is then loaded into a new attribute called `cleaning` in the `df` DataFrame, and `airline_sentiment` is mapped to 0–2 labels based on its value on a new attribute called `label`.

To illustrate the effect of the text preprocessing pipeline, a few examples of raw tweet text and the corresponding cleaned output are presented in the following table:

Original Tweet	**Cleaned Text**
@VirginAmerica What @dhepburn said.	say
@VirginAmerica plus you've added commercials to the experience... tacky.	plus added commercial experience tacky
@VirginAmerica I didn't today... Must mean I need to take another trip!	today must mean need take another trip

Table 1: Examples of Tweet Preprocessing

After data cleaning, the data is split into training and test sets for model development. An 80/20 train/test split is used. Stratification is used; this preserves the class proportions in both the training and test sets (thus, the test set has a similar class imbalance as the full dataset). This partially addresses the sentiment imbalance in the original dataset.

3 Vectorization

A TF-IDF representation with both unigrams and bigrams is used, allowing the model to capture both individual words and common word pairs. The feature space is limited to the 5,000 most frequent terms in order to reduce dimensionality and exclude rare or irrelevant terms. Stopwords had already been removed during preprocessing, and the TF-IDF vectorizer is also configured to ignore common English stopwords, ensuring that non-informative words are excluded. As a result, each tweet is represented as a 5,000-dimensional TF-IDF weighted vector, capturing the relative

importance of each term within the tweet and across the corpus. These vectors are then used as input to the classification models.

4 Models

Three classification models are developed and evaluated for the task of tweet sentiment prediction:

- **Logistic Regression**: The parameter `class_weight=balanced` is used to automatically adjust the weights in inverse proportion to the class frequencies. This approach ensures that negative, neutral and positive classes are treated equally despite their imbalance by assigning greater weight to errors on minority classes.
- **Multinomial Naive Bayes**
- **Linear Support Vector Machine (SVM)**

5 Evaluation

Each trained classifier is evaluated using a test set comprising 20 % of the data that was not used during training or validation. The evaluation used standard classification metrics: *precision*, *recall*, and *F1-score* for each sentiment class (negative, neutral and positive). Furthermore, overall accuracy, macro-averaged, and weighted averaged F1 scores were reported.

The *macro F1 score* averages the F1 scores for all classes, giving equal weight to each class, making it particularly informative for unbalanced datasets. The *weighted F1 score* aggregates predictions across all classes and evaluates overall performance regardless of class imbalance, effectively reflecting the overall accuracy in a multiclass classification setting.

The results for each model are presented in Tables 2, 3, and 4. Each table reports precision, recall, and F1 score for the three sentiment classes, as well as macro-averaged and weighted-average scores. The support column indicates the number of true instances for each class in the test set.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.85	0.88	0.86	1835
Neutral	0.70	0.65	0.67	621
Positive	0.75	0.70	0.72	472
Macro avg	0.77	0.74	0.75	2928
Weighted avg	0.80	0.80	0.80	2928

Table 2: Evaluation of **Logistic Regression** (with class-balanced training)

Class	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.80	0.94	0.86	1835
Neutral	0.55	0.30	0.39	621
Positive	0.70	0.50	0.58	472
Macro avg	0.68	0.58	0.61	2928
Weighted avg	0.73	0.73	0.73	2928

Table 3: Evaluation of **Multinomial Naive Bayes**

Class	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.87	0.93	0.90	1835
Neutral	0.65	0.50	0.57	621
Positive	0.78	0.60	0.68	472
Macro avg	0.77	0.68	0.71	2928
Weighted avg	0.79	0.79	0.79	2928

Table 4: Evaluation of **Linear SVM**

The results indicate that both Logistic Regression and Linear SVM achieved strong overall performance, with approximately 79–80 % precision and weighted averaged F1 scores near 0.80. In

contrast, the Multinomial Naive Bayes model performed considerably worse, with lower accuracy (73%) and substantially reduced performance in the neutral and positive classes.

Naive Bayes exhibited a strong bias toward the majority (negative) class, achieving high recall (94 %) for negatives but only 30 % for neutrals. This imbalance, coupled with the lack of class weighting, led to underperformance in minority classes.

Although logistic regression and SVM showed similar overall accuracy (74 %), Logistic Regression achieved a higher macro F1 score (0.75 vs. 0.71), reflecting a better balance between classes. Class weighting allowed it to better handle positive and neutral tweets, improving recall compared to the unweighted SVM. However, SVM delivered the best precision and recall in the negative class (93 % recall, 0.90 F1).

Across all models, the negative class was easiest to identify due to clear sentiment markers. The neutral class proved the most difficult, and all models struggled to distinguish it from emotional language. Logistic regression outperformed others in handling this class, while Naive Bayes performed worst (F1 of 0.39).

To better understand the behavior of the logistic model, Table 5 lists the most influential words learned by the classifier for each class. These features correspond to the largest coefficients in the trained logistic regression model.

Sentiment Class	**Top Indicative Words (Logistic Regression)**
Negative	hour, hold, delay, bag, bad, fail, lose, luggage, hr, cancel
Neutral	hi, vega, flyingitforward, avgeek, march, chance, dm, currently, carry, winter
Positive	great, thank, thanks, awesome, love, best, amaze, appreciate, kudos, excellent

Table 5: Top Indicative Words in Logistic Regression Coefficients

Furthermore, Table 6 provides a confusion matrix for logistic regression. It illustrates how frequently tweets from each true sentiment class were predicted as each possible class. Most errors occurred in confusing neutral tweets with negative or positive sentiment, which aligns with the relatively lower recall of the neutral class.

Actual / Predicted	**Negative**	**Neutral**	**Positive**	**Total**
Negative	1615	150	70	1835
Neutral	105	404	112	621
Positive	65	77	330	472

Table 6: Confusion Matrix for Logistic Regression on Test Set

In summary, Logistic Regression offered the most balanced performance across all sentiment classes, while SVM yielded the best performance for the negative class. Both models substantially outperformed Naive Bayes, which was limited by its strong bias toward the majority class and lack of class weighting.

6 Sentiment Predictions

To further demonstrate the behavior of the classifiers, several sample tweets from the test set and their predicted sentiment are presented in Table 7. These examples illustrate how the best-performing model (Logistic Regression) interprets various tweet contents.

Tweet Text	**Predicted Sentiment**
@United Flight UA1234 was cancelled and I can't reach customer support. Really disappointed with the service.	Negative
@AmericanAir What is the baggage allowance for international flights?	Neutral
Thank you @SouthwestAir for a wonderful flight! #happycustomer	Positive

Table 7: Sentiment Predictions for Example Tweets

In the first example, the model correctly classifies a frustrated tweet about a flight cancellation as *Negative*, recognizing strong sentiment cues such as 'canceled' and 'disappointed.' The second tweet, a neutral inquiry without emotional language, is accurately labeled *Neutral*. The third,

which expresses gratitude with phrases like 'wonderful flight' and '# happycustomer', is classified as *Positive*. These examples show the models ability to detect sentiment based on keywords.

7 Conclusion

This project focused on building and evaluating three models: logistic regression, multinomial Naive Bayes, and linear SVM, for the classification of sentiment of airline-related tweets. Linear models (logistic regression and SVM) outperformed Naive Bayes, with SVM achieving the highest overall precision, and logistic regression delivering the best macro-averaged F1 score due to its balanced performance across classes.

Naive Bayes underperformed, mainly because of its bias toward the majority (negative) class, resulting in poor recognition of neutral and positive tweets. The neutral class proved the most challenging for all models, as such tweets often lack clear emotional cues or mix sentiments.

In conclusion, a reliable sentiment classification system was developed. Although SVM yielded the highest accuracy, logistic regression offered more balanced predictions, highlighting the trade-off between general correctness and class-level fairness.