# Mini Project 2

**Vinay Sanga**

## 1 Introduction

Sentiment analysis is a critical component of natural language processing that helps in understanding the underlying emotions, opinions, and attitudes expressed in textual data. With the exponential growth of social media platforms, sentiment analysis has become indispensable for businesses, policymakers, and researchers to gauge public sentiment. The goal is to build and compare two distinct machine learning models to accurately analyze and predict sentiments expressed in tweets.

## 2 Data Preprocessing

Initially, I loaded into a DataFrame. An examination of the sentiment_label column revealed two unique values: 0 representing negative sentiments and 4 representing positive sentiments. The subsequent preprocessing phase involved several steps to clean and prepare the textual data for analysis:

- **Text Normalization:** I converted all tweets to lowercase to maintain consistency across the dataset.
- **Noise Removal:** Using regular expressions, I removed URLs, usernames, and special characters, which are generally irrelevant to sentiment analysis.
- **Tokenization:** I then tokenized the cleaned text, breaking it down into individual words or tokens. This step was crucial for analyzing the text data at a granular level.
- **Stopwords Removal:** To focus on the most meaningful words in the tweets, I removed common English stopwords. This not only reduced the dataset size but also improved the quality of the analysis.
- **Lemmatization:** Finally, I lemmatized the words to their base form. This helped in generalizing the analysis by reducing the complexity of the language used in the tweets.

This cleaned and processed text was then ready for vectorization, transforming the textual data into a numerical format suitable for machine learning models.

## 3 Modelling

### 3.1 Logistic Regression

I chose Logistic Regression for its straightforwardness and efficiency in binary classification tasks. I trained the model using TF-IDF vectorized input from the preprocessed tweets. The TF-IDF technique was instrumental in highlighting the importance of words based on their frequency and uniqueness across the dataset. To ensure the model's reliability and general applicability, I employed cross-validation with 5 folds, allowing me to assess its performance on different data subsets.

### 3.2 Linear Support Vector Classifier (Linear SVC)

My second model was a Linear Support Vector Classifier, which is renowned for its performance in high-dimensional spaces, such as those encountered in text classification tasks. Like the Logistic Regression model, I trained the Linear SVC on TF-IDF vectorized tweets. I also used cross-validation here to thoroughly evaluate its effectiveness.

### 3.3 Model Evaluation

The models were evaluated based on their accuracy scores and detailed classification reports, which included precision, recall, and F1-score for both sentiment classes.

| Metric | SVC | Logistic Regression |
|---|---|---|
| Accuracy | 0.74 | 0.74 |
| F1-score | 0.76 | 0.76 |
| Precision | 0.74 | 0.74 |
| Recall | 0.76 | 0.76 |

Table 1: Mean values of performance metrics for SVC and Logistic Regression models

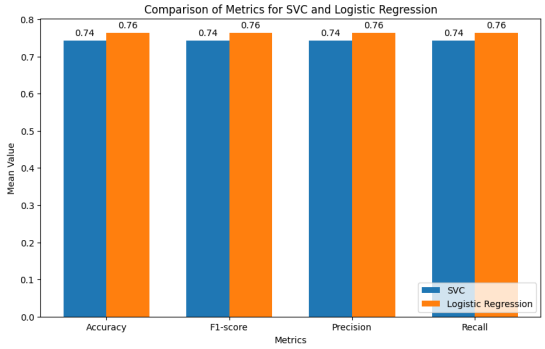


Figure 1: Comparison of Metrics for SVC and Logistic Regression.

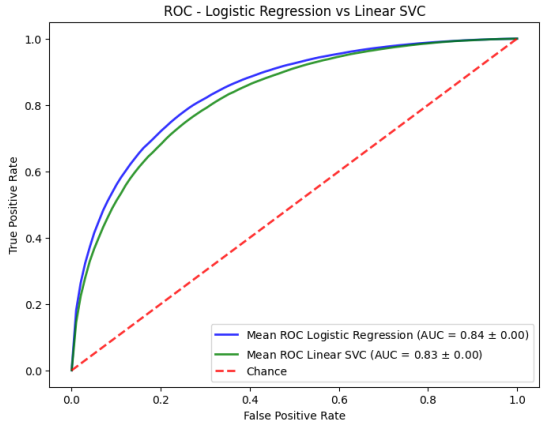


Figure 2: ROC for SVC and Logistic Regression.

- **Accuracy**: Both models exhibit an identical mean accuracy of 0.74. This metric is indicative of the overall correct predictions made by the models out of all predictions. The equivalence in accuracy suggests that both models are equally adept at identifying the correct sentiment in tweets when evaluated in a general context where false positives and false negatives carry similar weight.
- **F1-score**: The F1-score, a balance between precision and recall, stands at 0.76 for both models. A high F1-score indicates that the model has a lower occurrence of false positives and false negatives, which is essential in cases where finding the correct balance between precision and recall is critical.
- **Precision**: With a precision of 0.74, both models demonstrate the same likelihood of true positive sentiment predictions against all positive predictions. This metric is crucial when the cost of a false positive is high, as it indicates the reliability of the model's positive predictions.
- **Recall**: The recall for both models is at 0.76, implying that they are equally capable of identifying all the relevant instances of the positive class. This metric is particularly important in scenarios where it is essential to capture as many true positives as possible.
- I also plotted the ROC (fig. 2) for the two models, and realised the following:
  - The AUC for Logistic Regression is 0.84, indicating a strong predictive capability.
  - The AUC for Linear SVC is 0.83, which is comparable to that of Logistic Regression.
  - Both models exhibit an AUC significantly higher than 0.5, which would correspond to a random guess, hence they perform well.
  - The curves are above the chance diagonal, denoting that both classifiers have a good measure of separability.
  - The closeness of the two curves suggests similar performance across different thresholds.
  - The standard deviation for AUC is reported to be 0.00 for both models, signifying consistent performance across cross-validation folds.

As shown in Figure 1, the mean values of accuracy, F1-score, precision, and recall are quite similar for both the SVC and Logistic Regression models, indicating their comparable performance on the sentiment analysis task.

## 4 Conclusion

- One of the key learnings from this project was the importance of a methodical approach to preprocessing text data. By converting text to lowercase, removing noise, tokenizing, excluding stopwords, and lemmatizing, I was able to transform raw tweets into a clean and analyzable format.
- The comparative analysis of the Logistic Regression and SVC models provided me with insights into the strengths and limitations of each model. Despite their similar performance metrics, the experience highlighted the fact that no single model is a one-size-fits-all solution.