Tweet Sentimental Analysis

A Project Report Submitted in Partial Fulfilment of the Requirements for the completion of

Software Systems Development

by

Shivendra Pratap Singh (Roll No. 2024202022)
Sanchit Kumar (Roll No. 2024201042)
Anubhav Mishra (Roll No. 2024201001)
Sayam Agarwal (Roll No. 2024201025)

Under the Guidance of

Prof. Rahul Mishra

Department of Computer Science and Engineering
International Institute of Information and Technology, Hyderabad

December, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Sentiment Analysis, also known as opinion mining, is a technique in natural language processing (NLP) used to determine the emotional tone behind a body of text. It helps in understanding the sentiment expressed, whether it is positive, negative, or neutral.

Twitter is a popular microblogging platform where users share opinions, thoughts, and reactions on various topics in real time. Analyzing sentiments on Twitter can provide valuable insights into public opinion and trends for various applications, such as:

- Brand Monitoring: Companies analyze tweets to gauge customer satisfaction and public perception of their products or services.
- Political Sentiment: Governments and analysts track public sentiment about political events, elections, and policies.
- Market Research: Sentiment analysis helps businesses understand market trends and customer preferences.

1.1.1 Need For Society

The need for society arises from the fundamental human requirement to live and thrive in a structured community. Humans are inherently social beings, and societies provide the framework within which individuals can collaborate, share resources, and grow collectively. Below are some key aspects that highlight the importance of society:

1.1.2 Facts and Figures

Over 500 million tweets are sent daily, which amounts to approximately 6,000 tweets per second. 350 million monthly active users provide a diverse range of opinions, making Twitter a rich source for sentiment analysis.

Chapter 2

Methodology

2.1 Methodology

Methodology of Tweet sentimental analysis includes several steps namely Data Preprocessing, Feature Selection, Feature Extraction, Models Evaluation and Classifier Combining.

2.1.1 Data Preprocessing

Preprocessing involved several steps to clean and transform the raw data into a usable format. This included removing duplicate entries, irrelevant content (such as URLs or special characters), and stopwords. Tokenization was also performed to split text into individual words or tokens, which is crucial for further processing and feature extraction. Lowercase was applied to make the text case insensitive, and lemmatization was used to reduce the words to their root forms.

1. **Data loading:** Use the function load data(file path) to load the dataset from a CSV file. If the file fails to load (e.g., formatting issues), display a user-friendly error message and stop the

program. Display the first 10 rows of the dataset for a quick visual check of the data's structure and content.Exit the program if the dataset is missing or invalid.

2. **Data Cleaning:** Remove duplicate rows in the cleaned tweet column and reset the index for a consistent dataset. Drop rows with missing or empty tweets, ensuring only valid data is processed. Filter rows with labels 0 (Sad) or 4 (Happy), converting 4 to 1 for binary classification. Check that both classes (0 and 1) are present; if either is missing, raise a ValueError and stop execution.

3. **Feature Selection and Label Preparation:** Set X (features) as the cleaned tweet column, containing the preprocessed tweet text. Set y (labels) as the label column, representing sentiment as binary values (0 for Sad, 1 for Happy). Ensure X and y are properly aligned and clean before moving to vectorization. These variables (X and y) are then passed to subsequent steps for machine learning.

2.1.2 Feature Extraction

Text vectorization is the process of converting text into numerical vectors, which serve as features for machine learning models. Since most machine learning algorithms operate on numerical data, vectorization is a crucial step in the pipeline. In this project we use TF-IDF as a text vectorization technique.

- ⋆ **TF-IDF**:- TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a way to measure the importance of a word in a document, based on how often it appears and how rare it is across a collection of documents. Term Frequency measures how often a word appears in a single document. If a word appears many times in a document, it's likely to be important. TF-IDF combines the two scores to give you a final importance score for each word.

2.2 Models and Algorithms

2.2.1 Logistic Regression

Logistic Regression is a statistical method used for binary classification problems. Despite its name, it is a classification algorithm, not a regression algorithm. It predicts the probability that a given input belongs to a certain class, typically assigning outputs to one of two categories.

The logistic regression model is trained on the Xtrain and ytrain subsets using the .fit() function. The model learns to differentiate between Sad and Happy tweets based on the TF-IDF numerical representation. Hyperparameters like maxiter=30000 ensure the model has sufficient iterations to converge during training. Training prepares the model to make accurate predictions on new data based on learned patterns.

2.2.2 Naive Bayes

The Naive Bayes model is a family of simple yet effective probabilistic classifiers based on Bayes' Theorem. It is called "naive" because it assumes that the features in a dataset are conditionally independent, given the class label—an assumption that rarely holds in real-world data but often leads to surprisingly good results in practice.

Model Initialization and Training:

We initialized the Multinomial Naive Bayes model with a smoothing parameter alpha=1 to handle zero probabilities. The model was trained on our training dataset (Xtrain and ytrain). Prediction and Evaluation:

After training, we used the model to predict labels for the test dataset (Xtest). The accuracy of the predictions was calculated using accuracy score, providing a measure of how well the

model performed on the test set. We generated a detailed classification report, which included precision, recall, f1-score, and support for each class ('Sad' and 'Happy'). These steps allowed us to assess the performance of the Multinomial Naive Bayes model and understand its effectiveness in classifying the data.

2.3 Ensembling

Ensemble Method

The Voting Classifier combines the outputs of different base models, which could be classifiers like Decision Trees, Logistic Regression, or Support Vector Machines. It falls under the broader category of ensemble techniques because it leverages the diversity of individual model predictions to make a more reliable final prediction.

Types of Voting

- **Hard Voting:** Predicts the class label based on the majority vote from individual classifiers.
- **Soft Voting:** Uses the predicted probabilities for each class and averages them to make the final prediction. It tends to perform better if individual classifiers are well-calibrated.

Purpose

Voting classifiers help reduce overfitting and improve generalization by combining weak or strong learners.

2.3.1 Voting

A Voting Classifier is an ensemble machine learning model that aggregates the predictions of multiple models to improve overall performance.

Estimator List Creation

A list is created containing tuples where the first element is a string identifier for the model, and the second element is the model itself. In this example, two models are included: modelNB (e.g., Naive Bayes) and model (e.g., Logistic Regression). Additional models can be added as needed.

Voting Classifier Initialization

The Voting Classifier is initialized with the list of estimators. The voting parameter is set to 'soft', which means the classifier predicts the class label based on the average of predicted probabilities. The alternative is 'hard', which predicts based on the majority class label.

Model Training

The voting classifier is trained using the training data. This step aggregates the predictions of the individual models to form the final prediction.

Model Evaluation

The trained model predicts the labels for the test data. The accuracy of the predictions is calculated and displayed.

Detailed Performance Report

A detailed classification report is generated, including precision, recall, f1-score, and support for each class (e.g., 'Sad' and 'Happy'). The report is displayed in a formatted manner.

Summary

The Voting Classifier combines the strengths of multiple models to potentially achieve better performance than individual models. By using 'soft' voting, it averages the predicted probabilities to make the final prediction. This approach can be particularly useful when different models capture different aspects of the data, leading to improved generalization on unseen test data

2.4 Model Evaluation

Model was evaluated by displaying evaluation metrics including accuracy, confusion matrix, classification report, ROC curve, precision-recall curve, word clouds for both sentiments, distribution of tweet lengths, and class distribution.

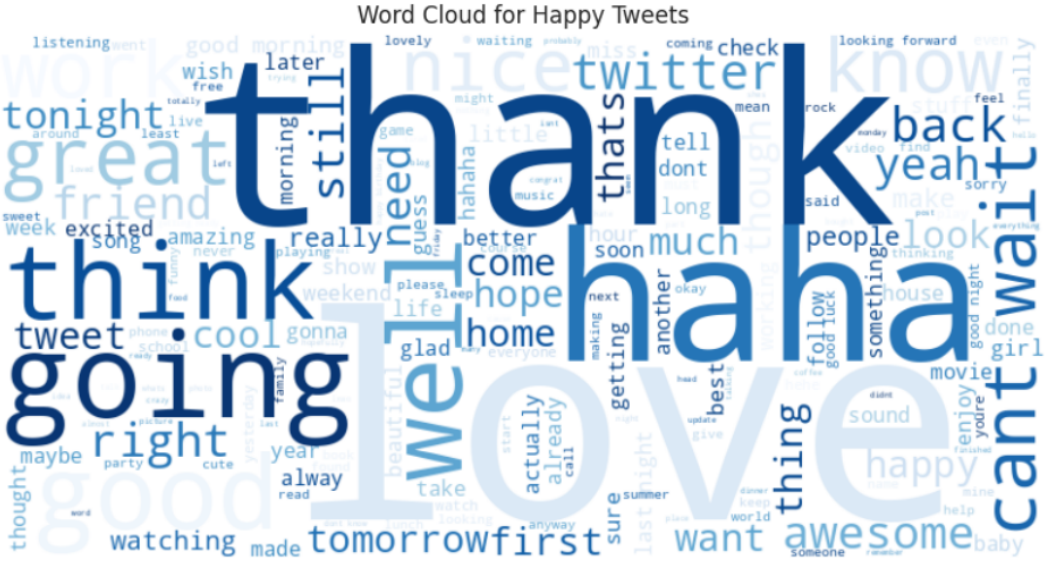

FIGURE 2.1: Word cloud of Positive Words

Figure 2.2: Word cloud of Negative Words

Figure 2.3: Confusion Matrix

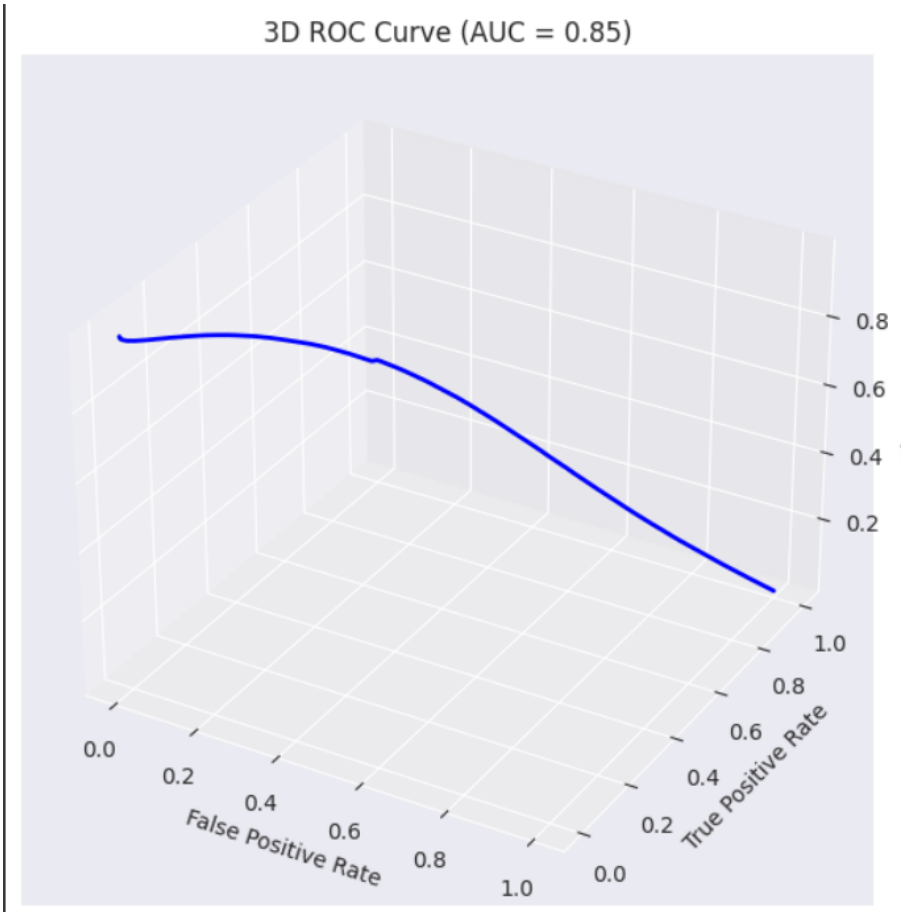

FIGURE 2.4: ROC Curve

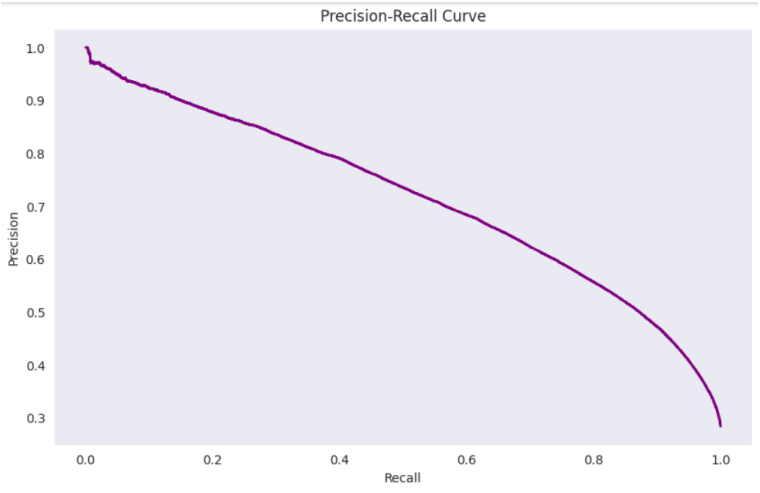

FIGURE 2.5: Precision Recall Curve

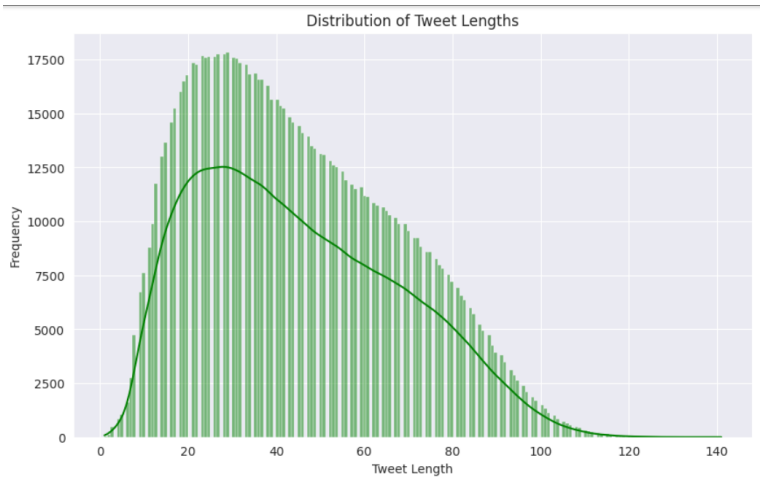

FIGURE 2.6: Distribution Curve

2.5 Saving the Model

Save the trained model and vectorizer using pickle for future use.

2.6 Implementation Environment

Implementation Environment is given in following table:

Tech & tools	Version
Jupyter	7.2.2
Python	3.12.1
NumPy	1.26.0
scikit-learn	1.3.1
seaborn	0.12.3
Pandas	2.1.2
matplotlib	3.8.1
flask	3.1x
Html	HTML5
CSS	CSS3

TABLE 2.1: Technology used and its Version

Chapter 3

Experimental Results

3.1 Experimental Result

- Data Loading and Cleaning: Successfully loaded cleanedtweets.csv, cleaned duplicates, and ensured balanced classes for Sad and Happy sentiments.
- Vectorization: Converted text data into numerical form using TF-IDF, capturing important words.
- Model Training: Trained a Logistic Regression model with high capacity (maxiter=30000) ensuring convergence.
- Model Evaluation: Achieved an accuracy of 78% on the test set with a balanced precision and recall for both classes.
- Confusion Matrix: Demonstrated effective classification with minimal misclassifications (e.g., 112329 True Sad, 115778 True Happy).
- ROC Curve: High AUC of 0.85 indicating strong discriminative ability between classes.

- Word Clouds: Revealed common words associated with each sentiment, providing qualitative insights into the data.
- Precision-Recall Curve: Highlighted the balance between precision and recall, useful for understanding performance in different thresholds.
- Additional Visualizations: Provided insights into tweet lengths and class distribution, ensuring data quality and further analysis.

Chapter 4

Differences Between Phase 1, Phase 2 and Phase 3

4.1 Differences Between Phase 1, Phase 2 and Phase 3

Phase 1 (30 percent): Project Planning and Dataset Acquisition

Objective: Search the appropriate dataset for modeling by loading, cleaning, and preprocessing the data.

Explanation:

Dataset Acquisition and Configuration: Search for an appropriate dataset to configure our model. Various datasets were trained initially but we found some inconsistencies like class imbalance and smaller dataset size. Finally, we found Google Scholar dataset on Kaggle which was appropriate to move forward.

Data Inspection and Cleaning: Inspected the dataset and cleaned hashtags, user mentions and URLs which provide no valuable insights for sentimental classification.

Class Balance Verification: Ensured that both sentiment classes ('Sad' labeled as 0 and 'Happy' labeled as 1) are adequately represented in the dataset. Verified that the count of each label confirms a balanced class distribution, which is crucial for unbiased model training.

Role distribution: Assigned roles and work to various members of the team so that they can start their preparation in advance.

Phase 2 (30 percent): Feature Extraction and Model Training

Objective: Convert textual data into numerical features, split the data for training and testing, and train a Logistic Regression model.

Explanation:

Feature and Label Definition: Extracted features (X) from the 'cleaned tweet' column and labels (y) from the 'label' column of the preprocessed DataFrame.

Text Vectorization with TF-IDF: Utilized TfIdfVectorizer to transform the text data into numerical form. This method captures the importance of words by considering their frequency in individual tweets and across the entire dataset, using n-grams ranging from unigrams to trigrams.

Data Splitting: Split the vectorized data into training and testing sets using train test split, maintaining the same class distribution with stratification. This ensures that both sets are representative of the overall dataset, which is essential for reliable evaluation.

Model Initialization and Training: Initialized a Logistic Regression model with specified parameters to control iterations and regularization strength. Trained the model using the training data while displaying a progress bar to track the training process.

Model Evaluation: Predicted sentiments on the test set and calculated the accuracy score. Generated a classification report detailing precision, recall, and F1-score for both 'Sad' and 'Happy' classes. This provides insights into the model's performance and areas for improvement.

Visualization of Frequent Words: Created word clouds for both 'Sad' and 'Happy' tweets to visualize the most common words associated with each sentiment. This helps in understanding the key terms that the model might be using to make predictions.

Phase 3 (40 percent): Advanced Modeling and Evaluation

Objective: Implement additional models (Naive Bayes, Voting Classifier), perform detailed evaluations, and save the final model for deployment.

Explanation:

Sentiment Prediction Functionality: Defined functions to preprocess new input text by converting to lowercase, removing URLs, mentions, hashtags, and non-alphabetic characters. This standardizes input data for consistent predictions. Implemented a function to predict the sentiment of user-provided text using the trained model. This allows for interactive testing and demonstrates the model's practical application.

Result Visualization: Plotted a confusion matrix to visualize the model's correct and incorrect predictions, providing a clear picture of performance across classes. Plotted the ROC curve to assess the trade-off between true positive and false positive rates, calculating the Area Under the Curve (AUC) as a performance metric.

Training a Naive Bayes Model: Trained a Multinomial Naive Bayes classifier as an alternative to Logistic Regression. Evaluated its performance using accuracy and classification reports to compare with the Logistic Regression model.

Implementing a Voting Classifier: Combined the Logistic Regression and Naive Bayes models using a Voting Classifier to potentially improve prediction accuracy. This ensemble method aggregates the predictions of multiple models for a more robust result.

Final Model Evaluation: Evaluated the Voting Classifier's performance, noting any improvements in accuracy or other metrics. This helps in selecting the best-performing model for deployment.

Model Saving for Deployment: Saved the final model (Voting Classifier) and the vectorizer using pickle, ensuring that the model can be loaded and used in future applications without retraining.

Final Results

Logistic Regression Model: Achieved an accuracy of approximately 77% on the test set.
Performance Metrics: Demonstrated balanced precision and recall for both 'Sad' and 'Happy' classes, indicating reliable performance.

Naive Bayes Model: Achieved an accuracy of approximately 76% on the test set.
Performance Metrics: Slightly lower performance compared to Logistic Regression, with variations in precision and recall.

Voting Classifier: Improved accuracy to approximately 78% by combining both models.
Performance Metrics: Showed enhanced performance, leveraging the strengths of both Logistic Regression and Naive Bayes classifiers.
Final Model Selection: The Voting Classifier was chosen as the final model due to its superior performance.

Model Deployment

Successfully saved the Voting Classifier and vectorizer for future use.

Sentiment Prediction: Enabled user input to predict sentiment, demonstrating the model's practical application.

Conclusion

The project successfully developed and evaluated multiple models for sentiment analysis of tweets. By employing advanced modeling techniques and thorough evaluation, the final model achieved high accuracy and is ready for deployment to predict sentiments in real-world data.

Chapter 5

Team Member Contributions

5.1 Team Member Contributions

Team Member 1 (Anubhav Mishra): Dataset Acquisition, Initial Analysis, Naive Bayes and Voting Classifier

Responsibilities:

- **Initial Dataset Acquisition and Analysis:** Searched for a appropriate dataset and analyzed various datasets before confirming the one to move ahead with.
- **Advanced Modeling with Naive Bayes:** Train a Multinomial Naive Bayes model as an alternative to the Logistic Regression model. Evaluate the Naive Bayes model using the same metrics as before.
- **Ensemble Learning with Voting Classifier:** Implement a Voting Classifier that combines the Logistic Regression and Naive Bayes models. Choose between 'hard' or 'soft' voting strategies. Train the Voting Classifier on the training data. Evaluate its performance and compare it to the individual models.

Key Points to Cover:

- **Alternative Models:** Explain the reasoning behind trying different models like Naive Bayes and how they differ from Logistic Regression.
- **Ensemble Methods:** Discuss how combining models using a Voting Classifier can improve overall performance by leveraging the strengths of each model.

Team Member 2 (Sayam Agarwal): Data Cleaning and Feature Extraction

Responsibilities:

- **Data Cleaning:** Remove duplicate tweets to prevent bias in the model. Drop any rows with missing or empty tweets to maintain data quality. Standardize sentiment labels by replacing label 4 with 1 (representing 'Happy') for consistency.
- **Class Balance Verification:** Ensure both sentiment classes ('Sad' labeled as 0 and 'Happy' labeled as 1) are adequately represented. Display the count of each label to confirm balanced class distribution.
- **Define Features and Labels:** Extract features (X) from the 'cleaned tweet' column. Extract labels (y) from the 'label' column.
- **Text Vectorization using TF-IDF:** Use TfIdf Vectorizer to transform text data into numerical features suitable for machine learning. Remove English stop words to focus on meaningful words. Consider n-gram ranges (unigrams, bigrams, trigrams) to capture context.

Key Points to Cover:

- **Data Cleaning Impact:** Explain how removing duplicates and empty entries improves model accuracy by ensuring the dataset reflects true sentiment expressions.
- **Label Standardization:** Discuss the need for consistent labeling for accurate model training and evaluation.
- **TF-IDF Vectorization:** Describe how TF-IDF works by assigning weights to words based on their frequency and importance. Highlight the benefit of using n-grams to capture phrases that contribute to sentiment.

Team Member 3 (Shivendra Singh): Model Training and Evaluation

Responsibilities:

- **Data Splitting:** Split the dataset into training and testing sets while maintaining class distribution using stratification. Ensure that the test set represents approximately 20% of the data.
- **Model Initialization and Training:** Initialize a Logistic Regression model with appropriate parameters. Train the model using the training data. Display a progress bar during training to indicate the process.
- **Model Evaluation:** Use the trained model to predict sentiments on the test set. Calculate the accuracy score to measure overall performance. Generate a classification report to provide precision, recall, and F1-score for each class. Display the evaluation results using the utility function.

Key Points to Cover:

- **Logistic Regression Choice:** Suitability of Logistic Regression for binary classification tasks like sentiment analysis.

- **Evaluation Metrics:** Interpret accuracy, precision, recall, and F1-score, and explain what they indicate about model performance.

Team Member 4 (Sanchit Kumar): Model Deployment,Data Visualisation and Frontend Integration

Responsibilities:

- **Model Deployment:** Define functions for text preprocessing to clean new input data (e.g., removing URLs, mentions, hashtags, punctuation). Implement a function that allows user input and predicts the sentiment using the trained model. Save the final model (Voting Classifier) and vectorizer using pickle for future use. Display a confirmation message upon successful model saving.
- **Integrating with Frontend:** Flask web application that loads a pre-trained machine learning model and vectorizer from a pickle file. The application has two routes: the root route ('/') renders a home page ('home.html') where users can input a sentence for prediction, and the '/predict' route handles POST requests to process the input. When a user submits a sentence, the application vectorizes the input using the loaded vectorizer and then uses the model to make a prediction. The result of the prediction is displayed on a separate page ('after.html'), showing either a "SAD" or "HAPPY" message along with a corresponding image. The HTML files include styling and background images to enhance the user interface.
- **Data Visualizations**

  The project involved creating various visualizations to analyze and interpret the model's performance and data distribution. These visualizations included:

  - A confusion matrix to highlight prediction accuracy for the "Sad" and "Happy" classes.

  - The ROC curve, showcasing the trade-off between true positive and false positive rates, along with the computed AUC score.
  - Precision-recall curves to assess the model's predictive power for imbalanced classes.
  - Histograms to explore the distribution of tweet lengths.
  - Word clouds were generated to represent the most common words in tweets labeled as "Sad" and "Happy," providing quick visual insights into each sentiment's dominant vocabulary.

Key Points to Cover:

- **User Interaction and Deployment:** Highlight the importance of making the model accessible for practical applications through user input and saving for future use.
- **Visualisation:** Interpret the word clouds to understand common themes or words that strongly influence the sentiment classification.

Team Collaboration and Integration

- **Ensure Smooth Workflow:** Each member should understand how their part fits into the overall project. Coordinate on data dependencies, such as ensuring that the data cleaned section is correctly used in modeling.
- **Consistent Documentation:** Maintain clear and consistent documentation for each part to facilitate understanding and future reference.
- **Review and Feedback:** Review each other's work to provide feedback and ensure that all parts align with the project's objectives.

Final Results to Present:

- **Model Performance Comparison:** Present the accuracy and evaluation metrics for the Logistic Regression, Naive Bayes, and Voting Classifier models. Highlight any improvements achieved through advanced modeling techniques.
- **Visualizations:** Show the confusion matrix and ROC curve to illustrate the model's effectiveness. Display the word clouds for both sentiments to provide insights into the data.
- **Model Deployment Confirmation:** Confirm that the final model is saved and ready for deployment, demonstrating the project's practical applicability.

Summary:

By dividing the work among four students, each focusing on a specific aspect of the project, the team can efficiently manage the workload and ensure a comprehensive understanding of the entire process. This approach facilitates specialization while promoting collaboration, resulting in a successful sentiment analysis model ready for real-world application.

Appendix A

Project Links

Github Source Code Link

Video Link