**Introduction**

The dataset used in this analysis contains tweets related to various airlines, focusing on customer experiences and feedback. It includes multiple columns, such as 'text', 'airline\_sentiment', and other metadata like 'airline', 'tweet\_location', and 'user\_timezone'. For this analysis, only the 'text' and 'airline\_sentiment' columns were retained to concentrate on sentiment classification. The 'airline\_sentiment' column captures the overall sentiment of the tweet, labeled as 'positive', 'neutral', or 'negative'.

**1. Text Preprocessing**

The primary focus of text preprocessing was to clean and standardize the textual data from tweets. Below are the key steps followed:

Data Selection: The dataset originally contained multiple columns. Only the columns 'text' and 'airline\_sentiment' were selected for the analysis to focus on sentiment classification.

Column Renaming: The 'airline\_sentiment' column was renamed to 'sentiment' for simplicity.

Data Cleaning: The tweets were subjected to cleaning using the following methods:

Conversion of text to lowercase.

Removal of URLs, HTML tags, punctuation, numbers, and special characters.

Removal of newline characters.

Stopword Removal: Stopwords, which are common words that do not contribute significantly to understanding sentiment (such as 'the', 'is', etc.), were removed using NLTK's stopwords list.

Lemmatization: Each word in the text was converted to its base or dictionary form using WordNet Lemmatizer. This helped in standardizing similar words that had different suffixes.

Sentiment Encoding: The sentiment labels were converted to numerical values: 'negative' = 0, 'neutral' = 1, 'positive' = 2.

**2. Model Training and Evaluation**

Feature Extraction: The text was converted into a numerical format using CountVectorizer. This transformed the textual data into a bag-of-words representation suitable for machine learning models.

Train-Test Split: The dataset was split into training and testing sets using an 80-20 split ratio.

Model Training: A Multinomial Naive Bayes (MNB) model was used for sentiment classification. This model is often effective for text classification tasks due to its handling of categorical feature counts.

**Model Performance:**

Accuracy: The accuracy achieved by the model on the test dataset was {accuracy\_mnb:.2f}%.

Confusion Matrix: The confusion matrix was plotted to provide insight into the number of correct and incorrect predictions across the sentiment classes.

Classification Report: Precision, recall, and F1-score metrics were reported for each sentiment class to assess the quality of the model's predictions.

ROC Curve: The ROC curve and AUC score were plotted for each class to evaluate the classifier's ability to distinguish between the sentiment classes effectively.

**3. Insights Gained from the Analysis**

Sentiment Distribution: The analysis indicated an imbalance in the dataset, with a higher number of 'negative' tweets compared to 'positive' and 'neutral'. This suggests that customers were more vocal about their negative experiences.

WordCloud Analysis: Two WordClouds were generated to visualize the most frequent words for 'positive' and 'negative' sentiments.

For positive sentiment, words like 'thanks', 'love', and 'great' were common, indicating frequent expressions of gratitude and satisfaction.

For negative sentiment, words like 'delayed', 'cancelled', and 'poor' were prevalent, highlighting the common issues experienced by customers.

Model Performance Insights: The Multinomial Naive Bayes model showed promising accuracy for sentiment classification, but there was some overlap in the 'neutral' and 'negative' categories, indicating potential confusion when distinguishing between dissatisfaction and neutral experiences. This could be addressed by using more advanced techniques, such as TF-IDF vectorization or exploring other classifiers like Random Forest or SVM.