**SENTIMENT ANALYSIS FOR MARKETING**

Abstract:

Sentiment analysis is a crucial tool in modern marketing, enabling businesses to gain insights into customer opinions, emotions, and attitudes towards their products or services. This module aims to provide a comprehensive sentiment analysis solution tailored for marketing purposes. It involves data collection, preprocessing, sentiment classification, and reporting. Leveraging natural language processing techniques and machine learning models, this module empowers marketers to understand customer sentiment, identify trends, and make data-driven decisions to enhance their marketing strategies.

Module: Sentiment Analysis for Marketing

1. Data Collection:

- Gather data from various sources, including social media, customer reviews, surveys, and other relevant platforms.

- Utilize web scraping, APIs, or manual data entry to compile a diverse dataset.

- Ensure data quality by handling duplicates, missing values, and noisy text.

2. Data Preprocessing:

- Text cleaning: Remove special characters, punctuation, and irrelevant symbols.

- Tokenization: Split text into individual words or tokens.

- Stopword removal: Eliminate common words that carry little sentiment information.

- Text normalization: Convert text to lowercase and handle word variations (e.g., "run" and "ran").

- Lemmatization or stemming: Reduce words to their base form.

3. Sentiment Classification:

- Select a sentiment analysis model, such as a pre-trained neural network (e.g., BERT, GPT) or traditional machine learning algorithms (e.g., Naive Bayes, SVM).

- Train the model on a labeled sentiment dataset or fine-tune a pre-trained model on domain-specific data.

- Apply the model to classify text into sentiment categories (e.g., positive, negative, neutral).

- Generate sentiment scores or probabilities for nuanced analysis.

4. Visualization and Reporting:

- Create visualizations (e.g., word clouds, sentiment distribution charts) to summarize sentiment trends.

- Generate reports and dashboards to present insights to marketing teams.

- Identify key themes, sentiment drivers, and actionable insights from the analyzed data.

5. Real-time Monitoring and Feedback:

- Implement a real-time monitoring system to track sentiment changes.

- Set up alerts for significant shifts in sentiment.

- Enable feedback loops to incorporate sentiment analysis results into marketing strategies.

6. Sentiment-Driven Marketing Strategies:

- Use sentiment insights to tailor marketing campaigns, product development, and customer engagement strategies.

- A/B testing to validate the effectiveness of sentiment-informed changes.

- Continuously adapt and refine marketing strategies based on sentiment feedback.

This sentiment analysis module equips marketing professionals with the tools and insights needed to understand customer sentiment, refine marketing strategies, and build stronger customer relationships in an ever-evolving digital landscape.

# **Summary:**

This code performs sentiment analysis on airline tweets using a logistic regression model and visualizes various aspects of the analysis. Below is a step-by-step summary of the code's strategy:

1. **Data Import and Preprocessing:**
   * The code begins by importing the necessary libraries, including pandas for data handling, matplotlib and seaborn for visualization, and scikit-learn for machine learning.
   * The airline tweet dataset is loaded from a CSV file.
2. **Data Selection:**
   * Only two columns, 'airline\_sentiment' (the sentiment label) and 'text' (the tweet content), are selected from the dataset.
3. **Sentiment Distribution Visualization:**
   * The code creates a histogram to visualize the distribution of airline sentiments.
   * It also creates a pie chart to visualize the sentiment distribution using percentages.
4. **Data Label Mapping:**
   * A mapping dictionary is defined to convert sentiment labels ('positive,' 'negative,' 'neutral') to numerical values (1, 0, 2).
   * Leading and trailing whitespaces in the 'airline\_sentiment' column are removed.
   * A new 'target' column is created to store the numerical sentiment values.
5. **Data Splitting:**
   * The dataset is split into training and testing sets using the train\_test\_split function.
6. **Text Vectorization (TF-IDF):**
   * Text data is vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. It generates TF-IDF features for both the training and testing sets.
7. **Model Training (Logistic Regression):**
   * A logistic regression model is trained using the training data.
8. **Model Evaluation:**
   * The code evaluates the model using the training and testing datasets:
     + It calculates and prints train and test accuracies.
     + It calculates ROC AUC scores for both the train and test sets.
     + It creates normalized confusion matrices and visualizes them using heatmaps for both the train and test sets.
9. **Binary Sentiment Classification:**
   * The code filters the dataset to focus on binary sentiment classification (positive vs. negative).
   * The same TF-IDF vectorizer is used to vectorize text data for binary classification.
   * Another logistic regression model is trained for binary classification.
10. **Model Evaluation (Binary Classification):**
    * The code evaluates the binary classification model:
      + It calculates and prints train and test accuracies.
      + It calculates ROC AUC scores for both the train and test sets.
11. **Feature Weight Visualization:**
    * The code plots a histogram of feature weights (coefficients) learned by the binary classification model.
12. **Identifying Most Positive and Negative Words:**
    * The code identifies and prints the most positive and negative words based on feature weights and a specified threshold.

Overall, this code provides a comprehensive analysis of airline tweets, including sentiment distribution, model training, evaluation, and visualization of feature weights and influential words for sentiment classification.

In [1]:

*# Import Libraries*

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import roc\_auc\_score, confusion\_matrix

from sklearn.model\_selection import train\_test\_split

In [2]:

*# Importing dataset*

df\_init = pd.read\_csv("/kaggle/input/twitter-airline-sentiment/Tweets.csv")

In [3]:

df\_init.head()

Out[3]:

|  | tweet\_id | airline\_sentiment | airline\_sentiment\_confidence | negativereason | negativereason\_confidence | airline | airline\_sentiment\_gold | name | negativereason\_gold | retweet\_count | text | tweet\_coord | tweet\_created | tweet\_location | user\_timezone |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 570306133677760513 | neutral | 1.0000 | NaN | NaN | Virgin America | NaN | cairdin | NaN | 0 | @VirginAmerica What @dhepburn said. | NaN | 2015-02-24 11:35:52 -0800 | NaN | Eastern Time (US & Canada) |
| 1 | 570301130888122368 | positive | 0.3486 | NaN | 0.0000 | Virgin America | NaN | jnardino | NaN | 0 | @VirginAmerica plus you've added commercials t... | NaN | 2015-02-24 11:15:59 -0800 | NaN | Pacific Time (US & Canada) |
| 2 | 570301083672813571 | neutral | 0.6837 | NaN | NaN | Virgin America | NaN | yvonnalynn | NaN | 0 | @VirginAmerica I didn't today... Must mean I n... | NaN | 2015-02-24 11:15:48 -0800 | Lets Play | Central Time (US & Canada) |
| 3 | 570301031407624196 | negative | 1.0000 | Bad Flight | 0.7033 | Virgin America | NaN | jnardino | NaN | 0 | @VirginAmerica it's really aggressive to blast... | NaN | 2015-02-24 11:15:36 -0800 | NaN | Pacific Time (US & Canada) |
| 4 | 570300817074462722 | negative | 1.0000 | Can't Tell | 1.0000 | Virgin America | NaN | jnardino | NaN | 0 | @VirginAmerica and it's a really big bad thing... | NaN | 2015-02-24 11:14:45 -0800 | NaN | Pacific Time (US & Canada) |

In [4]:

*# Select only the necessary columns for sentiment analysis*

df = df\_init[['airline\_sentiment', 'text']].copy()

In [5]:

*# Checking the result*

df.head()

Out[5]:

|  | airline\_sentiment | text |
| --- | --- | --- |
| 0 | neutral | @VirginAmerica What @dhepburn said. |
| 1 | positive | @VirginAmerica plus you've added commercials t... |
| 2 | neutral | @VirginAmerica I didn't today... Must mean I n... |
| 3 | negative | @VirginAmerica it's really aggressive to blast... |
| 4 | negative | @VirginAmerica and it's a really big bad thing... |

In [6]:

*# Count the occurrences of each sentiment category*

sentiment\_counts = df['airline\_sentiment'].value\_counts()

*# Visualize the distribution using a histogram with counts on bars*

plt.figure(figsize=(8, 6))

ax = sns.histplot(df['airline\_sentiment'], bins=3, color='skyblue', discrete=True)

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.title('Distribution of Airline Sentiments')

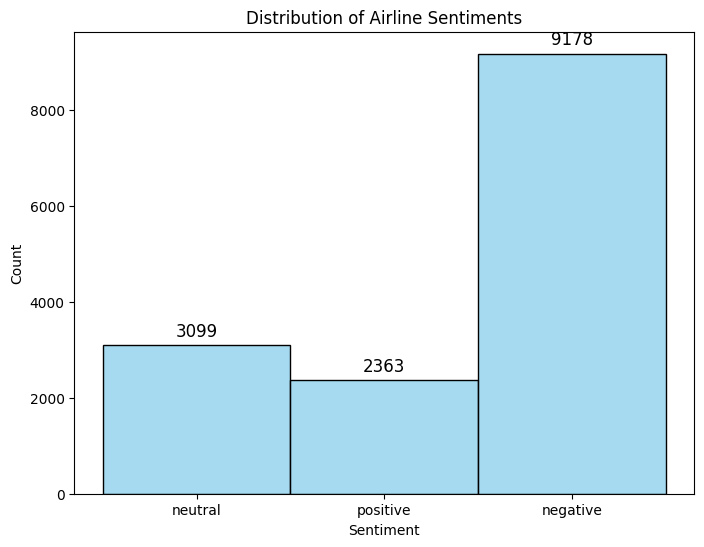
*# Add counts on top of the bars*

for p **in** ax.patches:

ax.annotate(f'**{**p.get\_height()**}**', (p.get\_x() + p.get\_width() / 2., p.get\_height()), ha='center', va='center', fontsize=12, xytext=(0, 10), textcoords='offset points')

plt.xticks()

plt.show()



In [7]:

*# Visualize the distribution of airline sentiments using a pie chart*

sentiment\_counts = df['airline\_sentiment'].value\_counts()

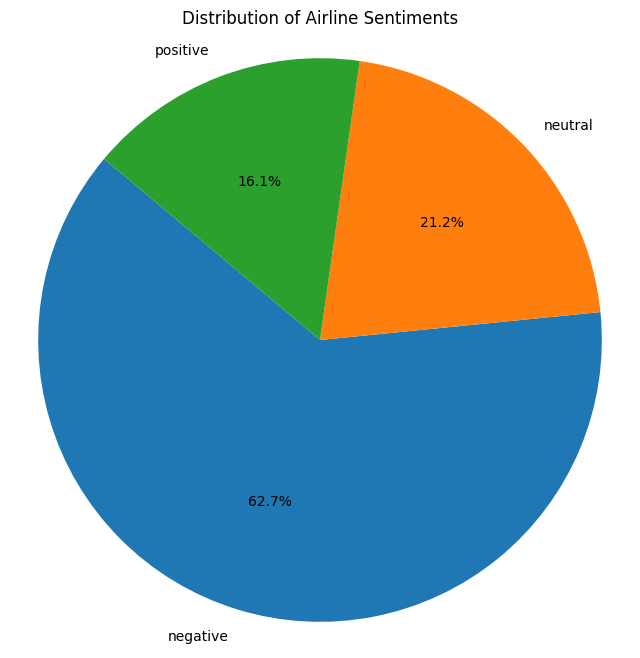
plt.figure(figsize=(8, 8))

plt.pie(sentiment\_counts, labels=sentiment\_counts.index, autopct='**%1.1f%%**', startangle=140)

plt.title('Distribution of Airline Sentiments')

plt.axis('equal') *# Equal aspect ratio ensures that pie is drawn as a circle.*

plt.show()



In [8]:

*# Create a mapping dictionary to convert sentiment labels to numerical values*

target\_map = {'positive': 1, 'negative': 0, 'neutral': 2}

*# Remove leading and trailing whitespaces from 'airline\_sentiment' column*

df['airline\_sentiment'] = df['airline\_sentiment'].str.strip()

*# Apply the mapping to create a new 'target' column with numerical sentiment values*

df['target'] = df['airline\_sentiment'].map(target\_map)

In [9]:

*# Checking the result*

df.head()

Out[9]:

|  | airline\_sentiment | text | target |
| --- | --- | --- | --- |
| 0 | neutral | @VirginAmerica What @dhepburn said. | 2 |
| 1 | positive | @VirginAmerica plus you've added commercials t... | 1 |
| 2 | neutral | @VirginAmerica I didn't today... Must mean I n... | 2 |
| 3 | negative | @VirginAmerica it's really aggressive to blast... | 0 |
| 4 | negative | @VirginAmerica and it's a really big bad thing... | 0 |

In [10]:

*# Split the dataset into training and testing sets*

df\_train, df\_test = train\_test\_split(df)

In [11]:

*# Checking the Result*

df\_train.head()

Out[11]:

|  | airline\_sentiment | text | target |
| --- | --- | --- | --- |
| 2665 | positive | @united Great landing in Denver, next Rapid Ci... | 1 |
| 14376 | negative | @AmericanAir are you kidding me? No one answe... | 0 |
| 10789 | neutral | @USAirways That's not the question. Question i... | 2 |
| 10833 | negative | @USAirways flight is already over. I think the... | 0 |
| 2795 | neutral | @united how long does it take for customer fee... | 2 |

In [12]:

*# Vectorize text data using TF-IDF*

vectorizer = TfidfVectorizer(max\_features=2000)

x\_train = vectorizer.fit\_transform(df\_train['text'])

x\_test = vectorizer.transform(df\_test['text'])

y\_train = df\_train['target']

y\_test = df\_test['target']

In [13]:

*# Train a logistic regression model*

model = LogisticRegression(max\_iter=500)

model.fit(x\_train, y\_train)

Out[13]:

LogisticRegression

LogisticRegression(max\_iter=500)

In [14]:

*# Evaluate the model*

train\_accuracy = model.score(x\_train, y\_train)

test\_accuracy = model.score(x\_test, y\_test)

print('Train accuracy: ', train\_accuracy)

print('Test accuracy: ', test\_accuracy)

Train accuracy: 0.8514571948998179

Test accuracy: 0.7975409836065573

In [15]:

*# Predict probabilities for ROC AUC calculation*

Pr\_train = model.predict\_proba(x\_train)

Pr\_test = model.predict\_proba(x\_test)

train\_auc = roc\_auc\_score(y\_train, Pr\_train, multi\_class='ovo')

test\_auc = roc\_auc\_score(y\_test, Pr\_test, multi\_class='ovo')

print('Train AUC: ', train\_auc)

print('Test AUC: ', test\_auc)

Train AUC: 0.9420130197115535

Test AUC: 0.9005944653741471

In [16]:

*# Predict labels for confusion matrix*

P\_train = model.predict(x\_train)

P\_test = model.predict(x\_test)

In [17]:

*# Create a normalized confusion matrix for the training set*

cm\_train = confusion\_matrix(y\_train, P\_train, normalize='true')

cm\_train

Out[17]:

array([[0.96099957, 0.00938899, 0.02961144],

[0.1943662 , 0.70591549, 0.09971831],

[0.31726556, 0.05039439, 0.63234005]])

In [18]:

*# Create a heatmap for the confusion matrix (training set)*

plt.figure(figsize=(8, 6))

sns.set(font\_scale=1.2)

heatmap = sns.heatmap(cm\_train, annot=True, fmt='.2%', cmap='Blues', cbar=False,

xticklabels=['Predicted Negative', 'Predicted Positive', 'Neutral'],

yticklabels=['Actual Negative', 'Actual Positive', 'Neutral'])

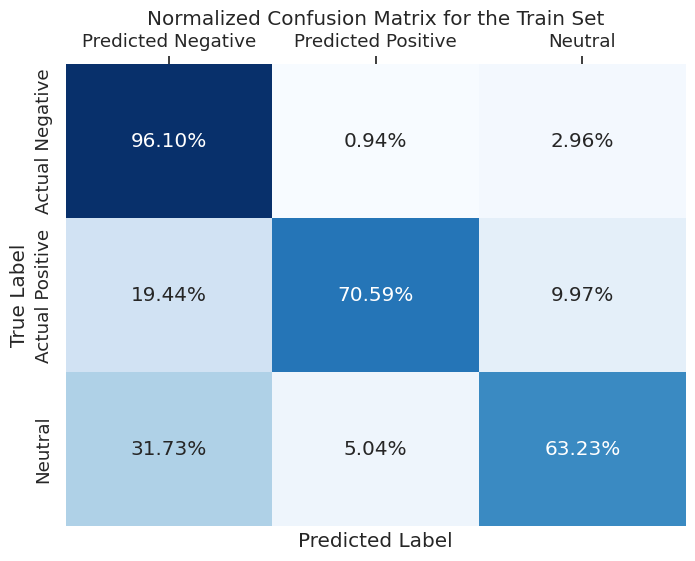
heatmap.xaxis.set\_ticks\_position('top') *# Move x-axis labels to the top*

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Normalized Confusion Matrix for the Train Set')

plt.show()



In [19]:

*# Create a normalized confusion matrix for the testing set*

cm\_test = confusion\_matrix(y\_test, P\_test, normalize='true')

cm\_test

Out[19]:

array([[0.93658537, 0.01773836, 0.04567627],

[0.22959184, 0.63265306, 0.1377551 ],

[0.39902081, 0.06854345, 0.53243574]])

In [20]:

*# Create a heatmap for the confusion matrix (testing set)*

plt.figure(figsize=(8, 6))

sns.set(font\_scale=1.2)

heatmap = sns.heatmap(cm\_test, annot=True, fmt='.2%', cmap='Blues', cbar=False,

xticklabels=['Predicted Negative', 'Predicted Positive', 'Neutral'],

yticklabels=['Actual Negative', 'Actual Positive', 'Neutral'])

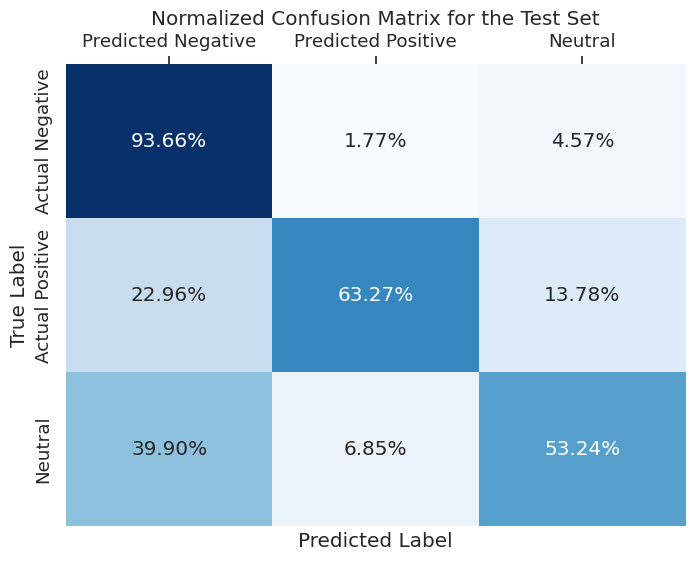
heatmap.xaxis.set\_ticks\_position('top') *# Move x-axis labels to the top*

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Normalized Confusion Matrix for the Test Set')

plt.show()



In [21]:

*# Filter the dataset to focus on binary sentiment classification (positive vs. negative)*

binary\_target\_list = [target\_map['positive'], target\_map['negative']]

df\_b\_train = df\_train[df\_train['target'].isin(binary\_target\_list)]

df\_b\_test = df\_test[df\_test['target'].isin(binary\_target\_list)]

In [22]:

*# Vectorize text data for the binary sentiment classification*

x\_train = vectorizer.fit\_transform(df\_b\_train['text'])

x\_test = vectorizer.transform(df\_b\_test['text'])

y\_train = df\_b\_train['target']

y\_test = df\_b\_test['target']

In [23]:

*# Train a logistic regression model for binary classification*

model = LogisticRegression(max\_iter=500)

model.fit(x\_train, y\_train)

binary\_train\_accuracy = model.score(x\_train, y\_train)

binary\_test\_accuracy = model.score(x\_test, y\_test)

print('Binary Train accuracy: ', binary\_train\_accuracy)

print('Binary Test accuracy: ', binary\_test\_accuracy)

Binary Train accuracy: 0.926304897677627

Binary Test accuracy: 0.9141751670770313

In [24]:

*# Predict probabilities for ROC AUC calculation (binary classification)*

Pr\_train = model.predict\_proba(x\_train)[:, 1]

Pr\_test = model.predict\_proba(x\_test)[:, 1]

binary\_train\_auc = roc\_auc\_score(y\_train, Pr\_train)

binary\_test\_auc = roc\_auc\_score(y\_test, Pr\_test)

print('Binary Train AUC: ', binary\_train\_auc)

print('Binary Test AUC: ', binary\_test\_auc)

Binary Train AUC: 0.9777289012131434

Binary Test AUC: 0.9605713682368735

In [25]:

*# Plot a histogram of feature weights (coefficients)*

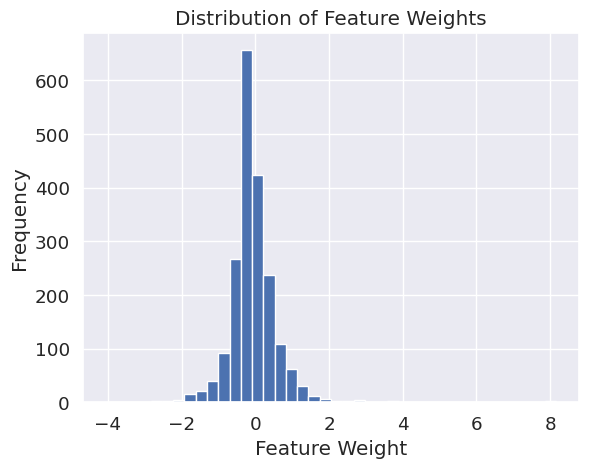
plt.hist(model.coef\_[0], bins=40)

plt.xlabel('Feature Weight')

plt.ylabel('Frequency')

plt.title('Distribution of Feature Weights')

plt.show()



In [26]:

*# Get the vocabulary index map*

word\_index\_map = vectorizer.vocabulary\_

In [27]:

*# Define a threshold for identifying most positive and most negative words*

threshold = 2

In [28]:

*# Identify and print the most positive words*

print('Most Positive Words')

for word, index **in** word\_index\_map.items():

weight = model.coef\_[0][index]

if weight > threshold:

print(word, weight)

Most Positive Words

great 5.516378614880334

virginamerica 3.4165631737297506

thank 8.172492647617368

southwestair 2.728627527382746

jetblue 3.1586422137139065

thanks 8.083441401654769

good 2.805464965619352

love 4.449114200749592

best 3.8620140153411207

appreciate 2.336612511736386

awesome 4.091284298701974

nice 2.16154339981104

thx 2.4222423243948117

amazing 3.6943805117897175

excellent 2.6209683927563843

worries 2.7557781608971568

wonderful 2.240905852132964

kudos 2.87036770762045

In [29]:

*# Identify and print the most negative words*

print('Most Negative Words')

for word, index **in** word\_index\_map.items():

weight = model.coef\_[0][index]

if weight < -threshold:

print(word, weight)

Most Negative Words

no -3.58181655382038

not -4.065478719629374

cancelled -2.731104617701844

why -2.396502989548685

hold -2.57304890251383

delayed -2.8279386530435775

hours -3.228530121027036

rude -2.0859089592041653

delay -2.0750718875362524

hour -2.336870367368504

don -2.0598653786797785

worst -3.1406265902569985

nothing -2.0462535544430076

OUTPUT:

1. Data Visualization:

- Distribution of Airline Sentiments: The code creates a histogram and a pie chart to visualize the distribution of airline sentiments.

2. Data Preprocessing:

- The dataset is loaded and columns ‘airline\_sentiment’ (sentiment label) and ‘text’ (tweet content) are selected.

- Sentiment labels are mapped to numerical values (0 for negative, 1 for positive, 2 for neutral).

3. Model Training and Evaluation:

- The dataset is split into training and testing sets.

- Text data is vectorized using TF-IDF.

- A logistic regression model is trained and evaluated for multiclass sentiment classification.

- Train and test accuracies, ROC AUC scores, and normalized confusion matrices are displayed.

4. Binary Sentiment Classification:

- The dataset is filtered to focus on binary sentiment classification (positive vs. negative).

- Text data is vectorized again.

- Another logistic regression model is trained and evaluated for binary sentiment classification.

5. Feature Weight Visualization:

- A histogram of feature weights (coefficients) learned by the binary classification model is plotted.

6. Identifying Most Positive and Negative Words:

- The code identifies and prints the most positive and negative words based on feature weights and a specified threshold.

The code does not provide visualizations for the ROC curves or the binary classification confusion matrices. However, it outputs various performance metrics and lists the most positive and negative words based on feature weights.

The key output metrics include:

- Train accuracy for multiclass classification

- Test accuracy for multiclass classification

- Train AUC for multiclass classification

- Test AUC for multiclass classification

- Normalized confusion matrices for both the train and test sets for multiclass classification

- Train accuracy for binary classification (positive vs. negative)

- Test accuracy for binary classification (positive vs. negative)

- Train AUC for binary classification (positive vs. negative)

- Test AUC for binary classification (positive vs. negative)

Additionally, it lists the most positive and negative words based on feature weights and a specified threshold.