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)

	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
1	570301130888122368	positive	0.3486	NaN	0.0000	VirginAmerica	NaN	jnardino	NaN	0	@VirginAmerica plus you've added commercialst...	NaN	2015-02-24 11:15:59 -0800	NaN	Pacific Time (US & Canada)
2	570301083672813571	neutral	0.6837	NaN	NaN	VirginAmerica	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	VirginAmerica	NaN	jnardino	NaN	0	@VirginAmerica it's really aggressi	NaN	2015-02-24 11:15:36 -0800	NaN	Pacific Time (US & Canada)

	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
											ve to blast...				
4	570308170744462722	negative	1.0000	Can't Tell	1.0000	VirginAmerica	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()
```

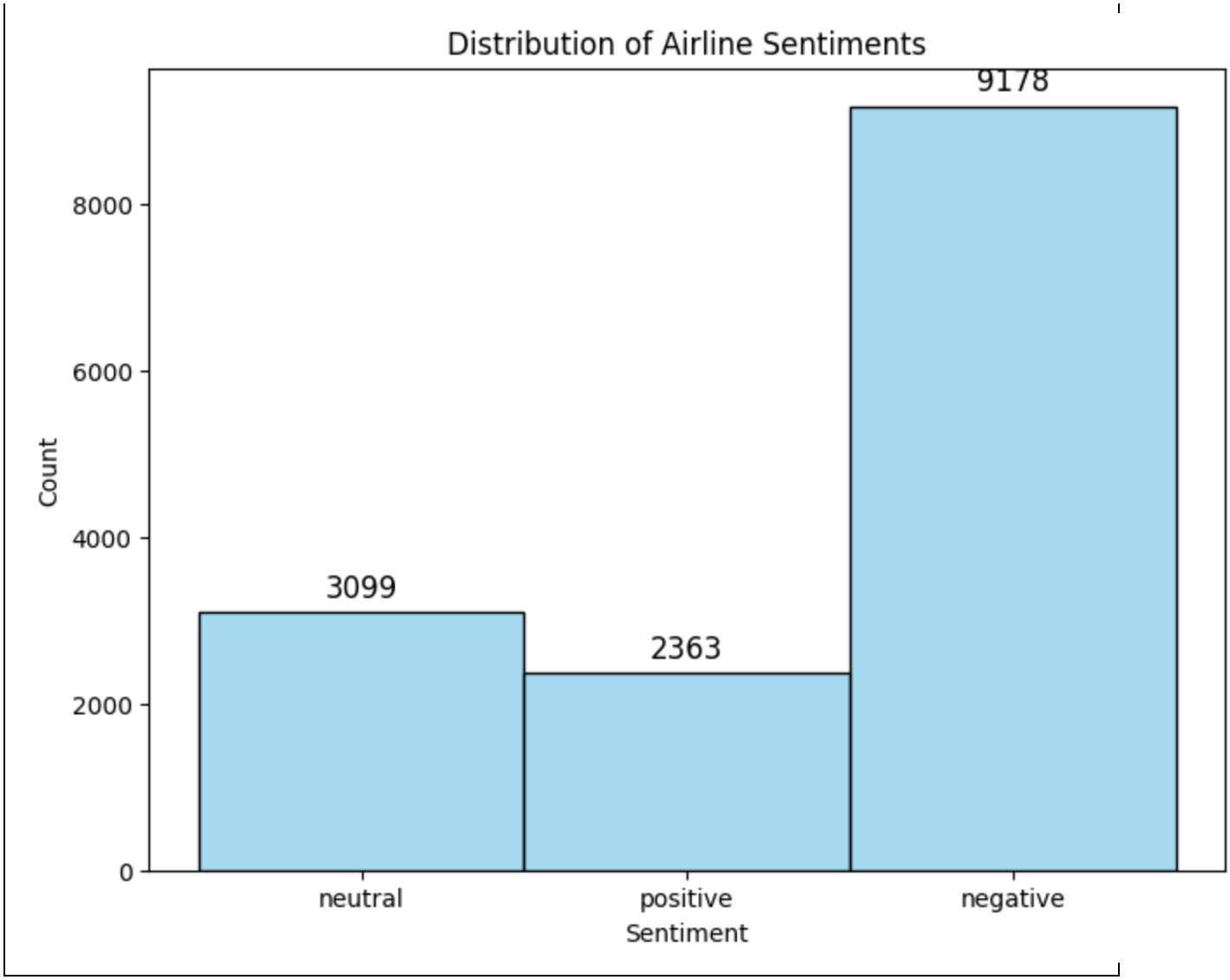
Out[5]:

	airline_sentiment	text
0	neutral	@VirginAmerica What @dhepburn said.

	airline_sentiment	text
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...

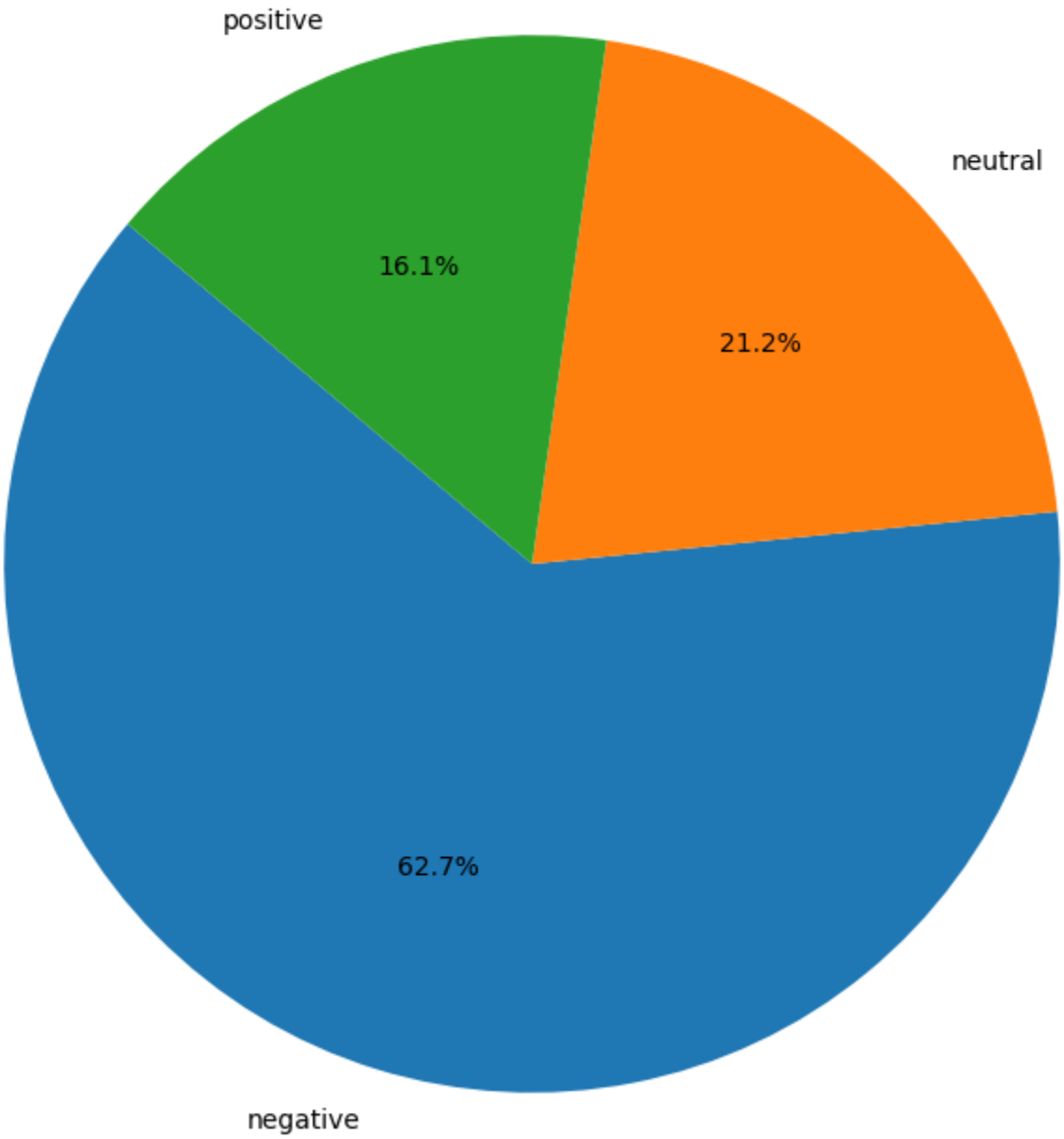
In [6]:

```
# 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 v
alues
```

```
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
```

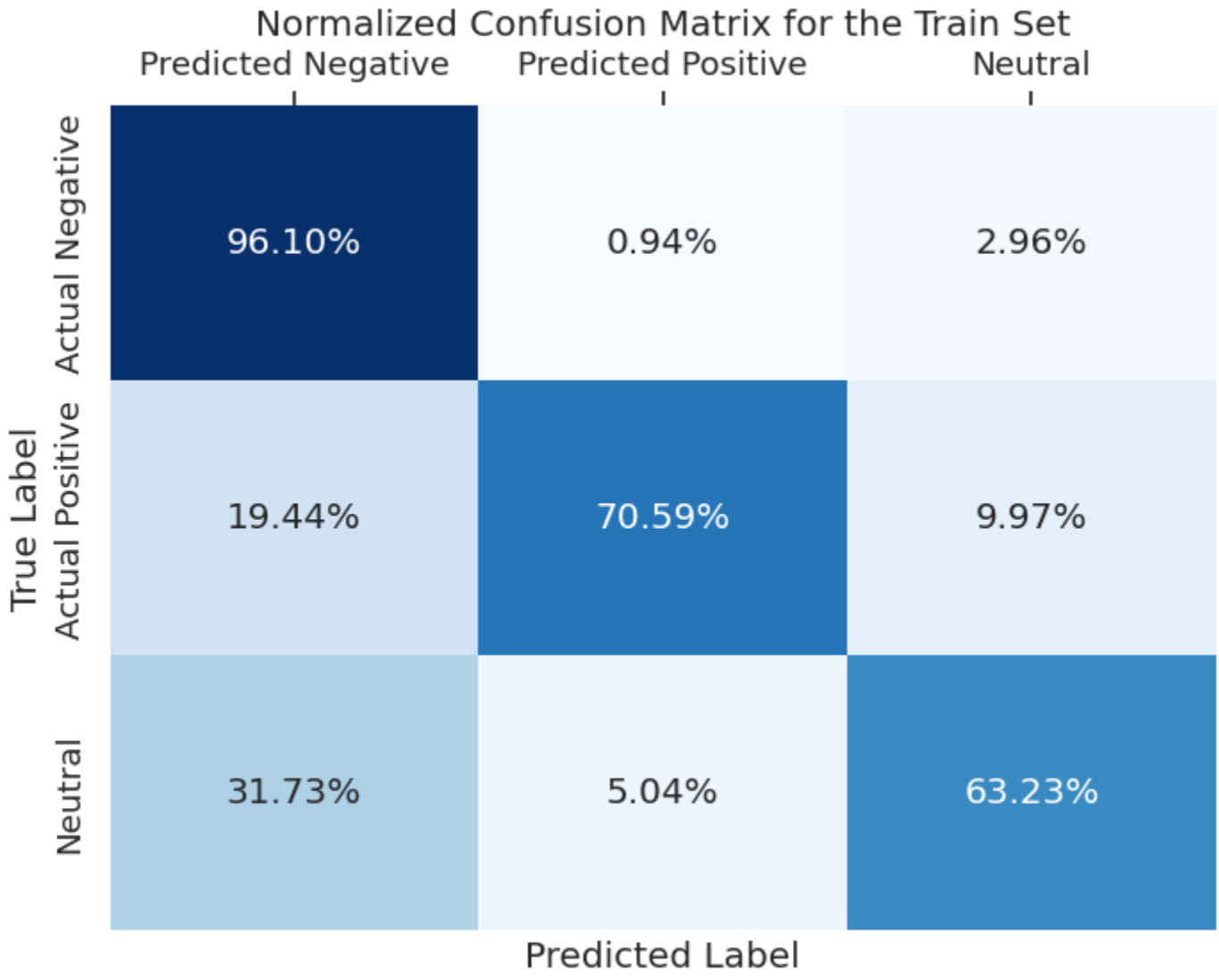
Out[17]:

```
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=Fa
lse,
            xticklabels=['Predicted Negative', 'Predicted Positive', 'Neutra
l'],
            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
```

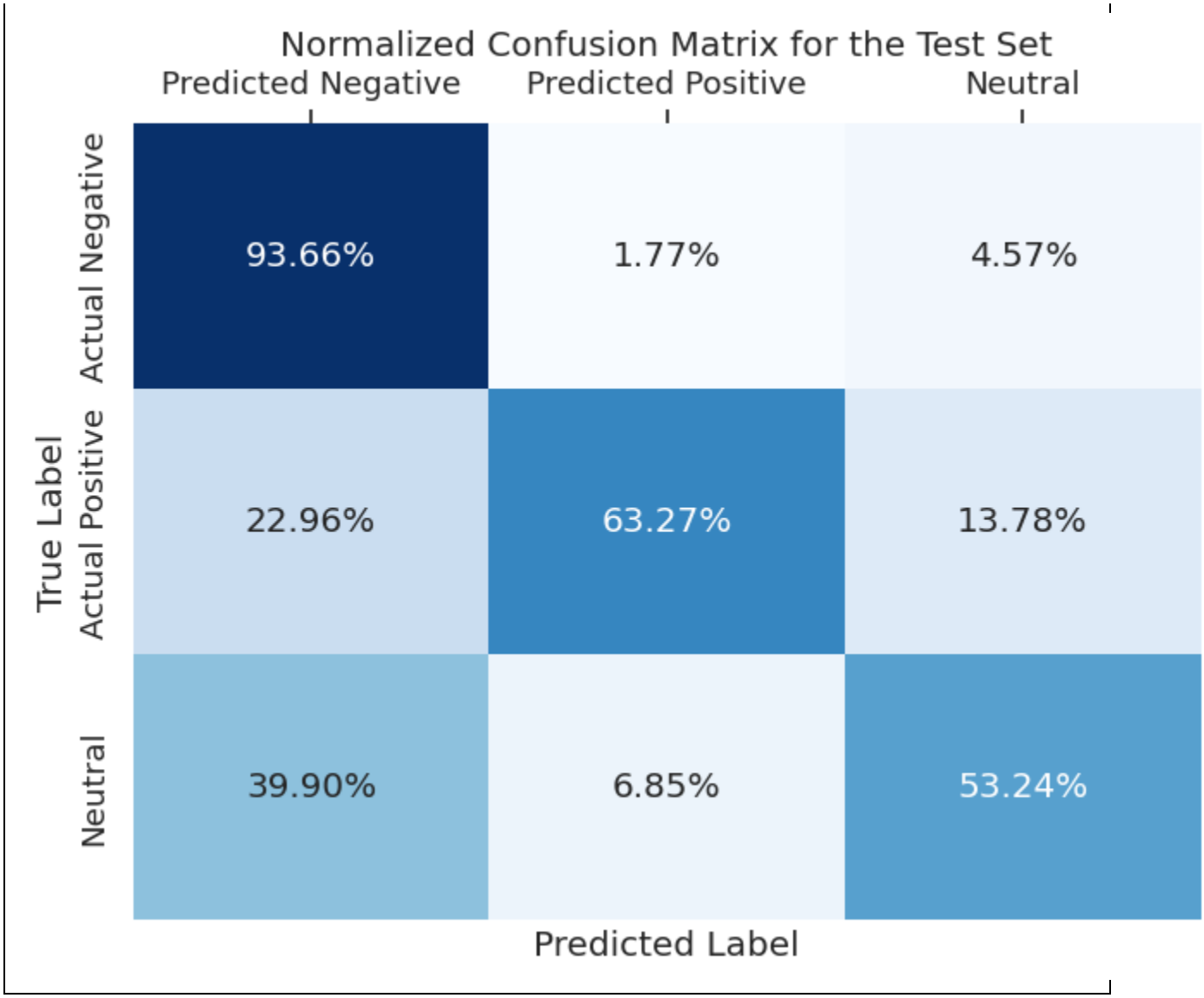
Out[19]:

```
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=Fal
se,
            xticklabels=['Predicted Negative', 'Predicted Positive', 'Neutra
l'],
            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
```

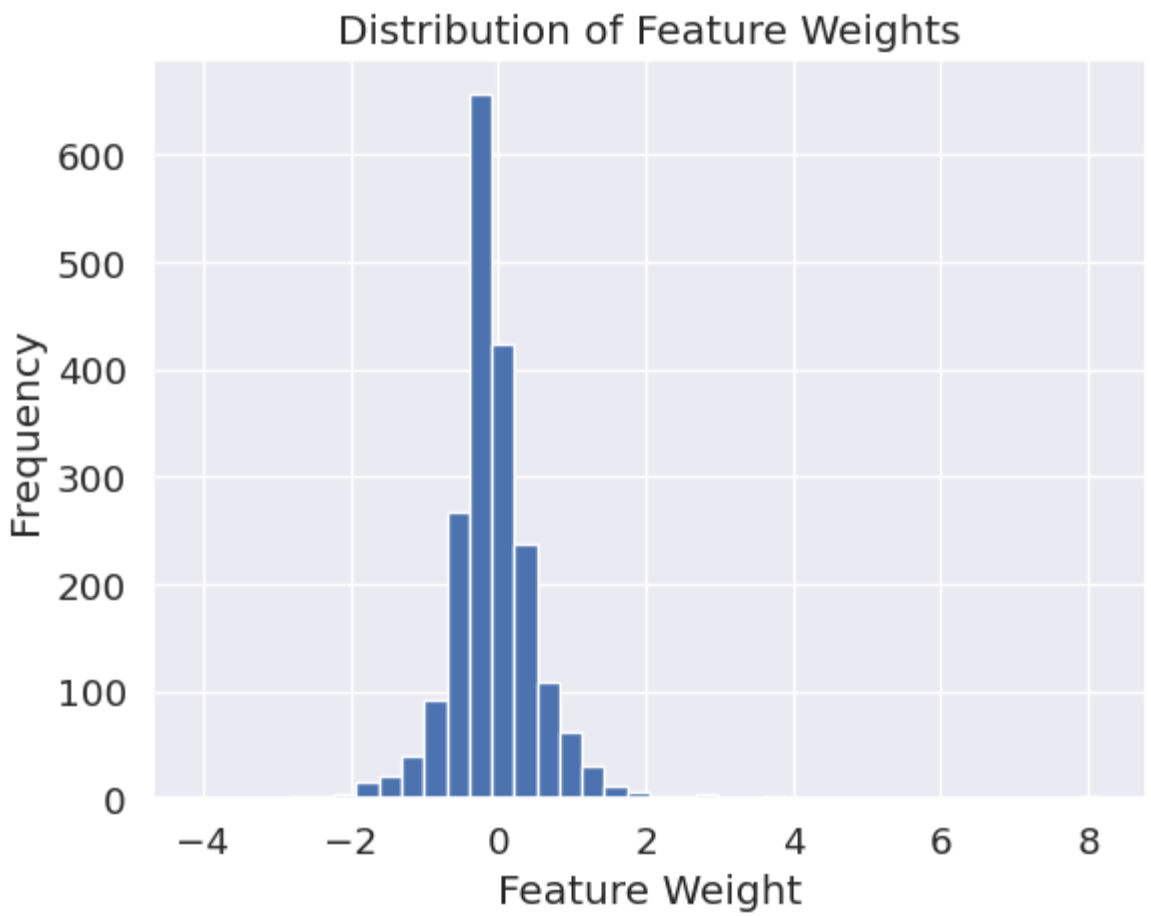
In [24]:

```
# 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.