**Train and testing the model**

**Step 11:**

**Splitting the data:**

* Split your data into two parts **training set** and **testing set**. The training set is used to train your machine learning model, and the testing set is used to evaluate its performance. Use the training data (X\_train and y\_train) to train your sentimental analysis model.

from sklearn.model\_selection

import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['text'], df['airline\_sentiment'], test\_size=0.2, random\_state=42)

* You can use various machine learning algorithms like Support Vector Machines, or deep learning models like LSTM or GRU for this task. Once the model is trained, use the testing data (X\_test and y\_test) to evaluate its performance

**Step 12:**

**Feature extraction:**

Feature extraction in sentiment analysis for marketing involves identifying and extracting relevant information from textual data to understand customer sentiments effectively.

* Key techniques include recognizing specific product aspects, detecting emotions, extracting opinion words, identifying industry-specific keywords, categorizing feedback, analyzing temporal patterns, and understanding social interactions.
* These extracted features serve as the basis for analyzing customer sentiments, preferences, and feedback, enabling businesses to make informed marketing decisions.

from sklearn.feature\_extraction.text

import TfidfVectorizer

vectorizer = TfidfVectorizer(max\_features=2500, min\_df=7, max\_df=0.8)

X\_train = vectorizer.fit\_transform(X\_train).toarray()

X\_test = vectorizer.transform(X\_test).toarray()

**Step 13:**

**Model Training:**

* Feed the preprocessed and feature-extracted data into the selected model. During training, the model learns the patterns and relationships between features and sentiment labels from the training data.
* Use a separate set of data (validation or test set) to evaluate the model's performance.
* Once the model performs satisfactorily, deploy it to analyze customer feedback, reviews, or social media posts, providing valuable insights for marketing strategies.

Top of Form

from sklearn.ensemble

import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators=1000, random\_state=0)

classifier.fit(X\_train, y\_train)

**Distribution of sentiments:**

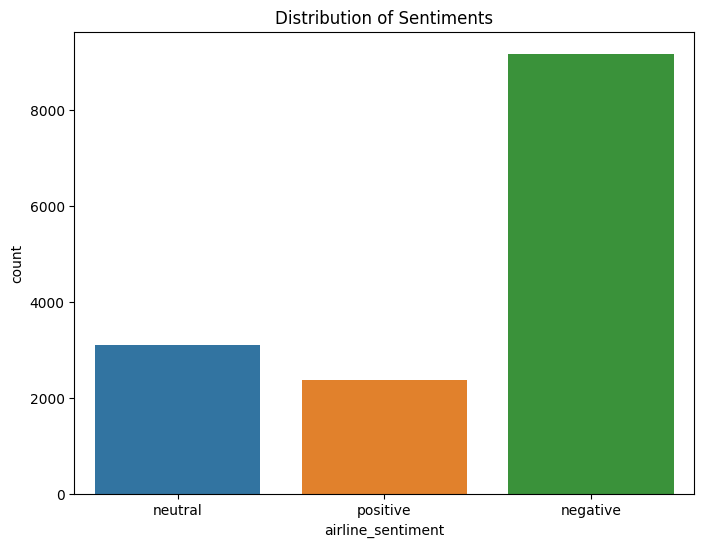
* To visualize the distribution of sentiments in the Twitter airline dataset, you can use a bar chart or any other appropriate visualization method. In this code, the **value\_counts()** function is used to count the occurrences of each sentiment class in the 'sentiment' column of the dataset. The resulting counts are then plotted as a bar chart using matplotlib.
* Make sure to adjust the code according to the specifics of your dataset file and the column names used in the dataset. This will provide you with a visual representation of the distribution of sentiments in the Twitter airline dataset.

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

sns.countplot(x='airline\_sentiment', data=df)

plt.title('Distribution of Sentiments')

plt.show()



**Histogram of tweet lengths:**

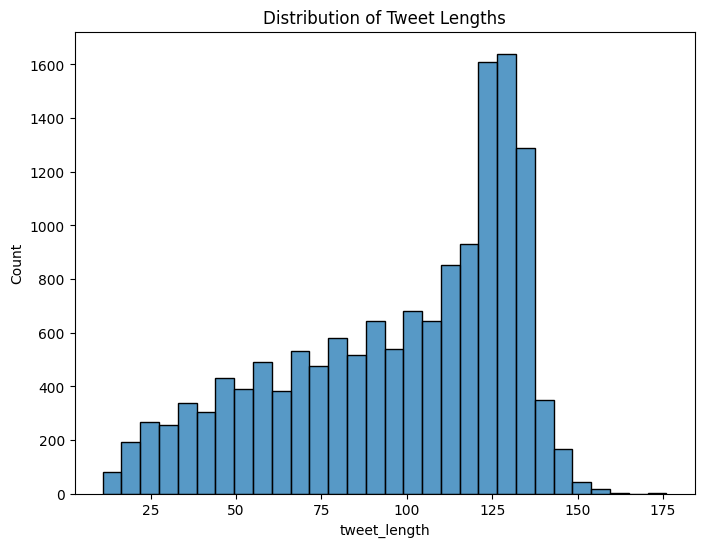
* Load your dataset containing a column named 'text'. Calculate the length of each tweet in the 'text' column.Plot a histogram with tweet lengths on the x-axis and frequency on the y-axis.

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

sns.histplot(df['tweet\_length'], bins=30)

plt.title('Distribution of Tweet Lengths')

plt.show()



**Boxplot of tweet lengths:**

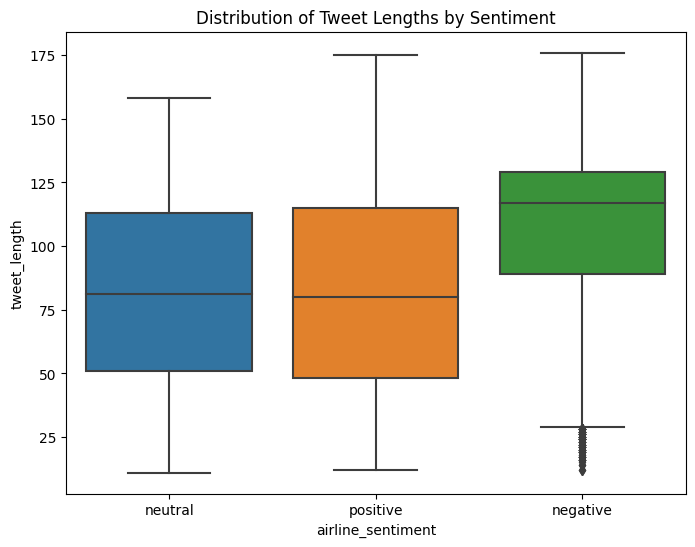
The boxplot shows the range and distribution of tweet lengths, allowing for a quick understanding of the variation in text lengths within the dataset. Use the 'tweet\_length' column to create a boxplot, displaying the distribution of tweet lengths. Compute the length of each tweet and store it in a new column, like 'tweet\_length'.

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

sns.boxplot(x='airline\_sentiment', y='tweet\_length', data=df)

plt.title('Distribution of Tweet Lengths by Sentiment')

plt.show()



**Step 14:**

**Classification report:**

* To create a classification report for sentiment analysis using the Twitter airline dataset, you would first need to preprocess the data, extract features, train a machine learning model, make predictions on the test data, and then evaluate the model using the **classification\_report** function from scikit-learn.
* Assuming you have a dataset with columns **text** and **sentiment**, where **text** contains the tweet text and **sentiment** contains the sentiment labels (positive, negative, neutral), you can preprocess the data. Preprocessing steps may include lowercasing, removing special characters, etc. Train a machine learning model on the training data.

def evaluate\_model(y\_test, y\_pred):

    print('Classification Report:')

    print(classification\_report(y\_test, y\_pred))

    print('Confusion Matrix:')

    print(confusion\_matrix(y\_test, y\_pred))

    print('Accuracy Score:')

    print(accuracy\_score(y\_test, y\_pred))

**Output:**

Classification Report:

precision recall f1-score support

negative 0.79 0.95 0.86 1889

neutral 0.65 0.41 0.50 580

positive 0.80 0.50 0.62 459

accuracy 0.77 2928

macro avg 0.75 0.62 0.66 2928

weighted avg 0.76 0.77 0.75 2928

Confusion Matrix:

[[1799 65 25]

[ 312 235 33]

[ 169 60 230]]

**Step 15:**

**Accuracy score:**

* Clean and split the text data into training and testing sets.Train a machine learning model using the training dataUse the model to predict sentiments for the test data.
* Compare the predictions with the actual sentiments to calculate the accuracy score using a simple formula.The accuracy score indicates the percentage of correctly predicted sentiments, providing insight into the model's performance.

y\_pred = classifier.predict(X\_test)

evaluate\_model(y\_test, y\_pred)

**Output:**

Accuracy Score:

0.773224043715847

**Step 16:**

**Confusion matrix:**

* Train a machine learning model on the training data. the confusion matrix is calculated using the test data and then visualized as a heatmap using the seaborn library. The rows of the confusion matrix represent the actual classes (negative, neutral, positive), and the columns represent the predicted classes.
* The values in the matrix show how many samples were classified into each category.Make sure to adjust the code according to the specifics of your dataset and the preprocessing steps you have performed.

import matplotlib.pyplot as plt

import seaborn as sns

def plot\_confusion\_matrix(y\_test, y\_pred):

    cm = confusion\_matrix(y\_test, y\_pred)

    df\_cm = pd.DataFrame(cm, index = [i for i in ['negative', 'neutral', 'positive']],

                  columns = [i for i in ['negative', 'neutral', 'positive']])

    plt.figure(figsize = (10,7))

    sns.heatmap(df\_cm, annot=True, fmt='d', cmap='Blues')

    plt.title('Confusion Matrix')

    plt.xlabel('Predicted')

    plt.ylabel('True')

    plt.show()

plot\_confusion\_matrix(y\_test, y\_pred)

