**SENTIMENT ANALYSIS FOR MARKETING**

**TEAM MEMBER**

**620121243036: Poornima.A**

**Phase2 Sentiment Analysis for Marketing**

**Project:** Sentiment Analysis For Marketing



**INTRODUCTION:**

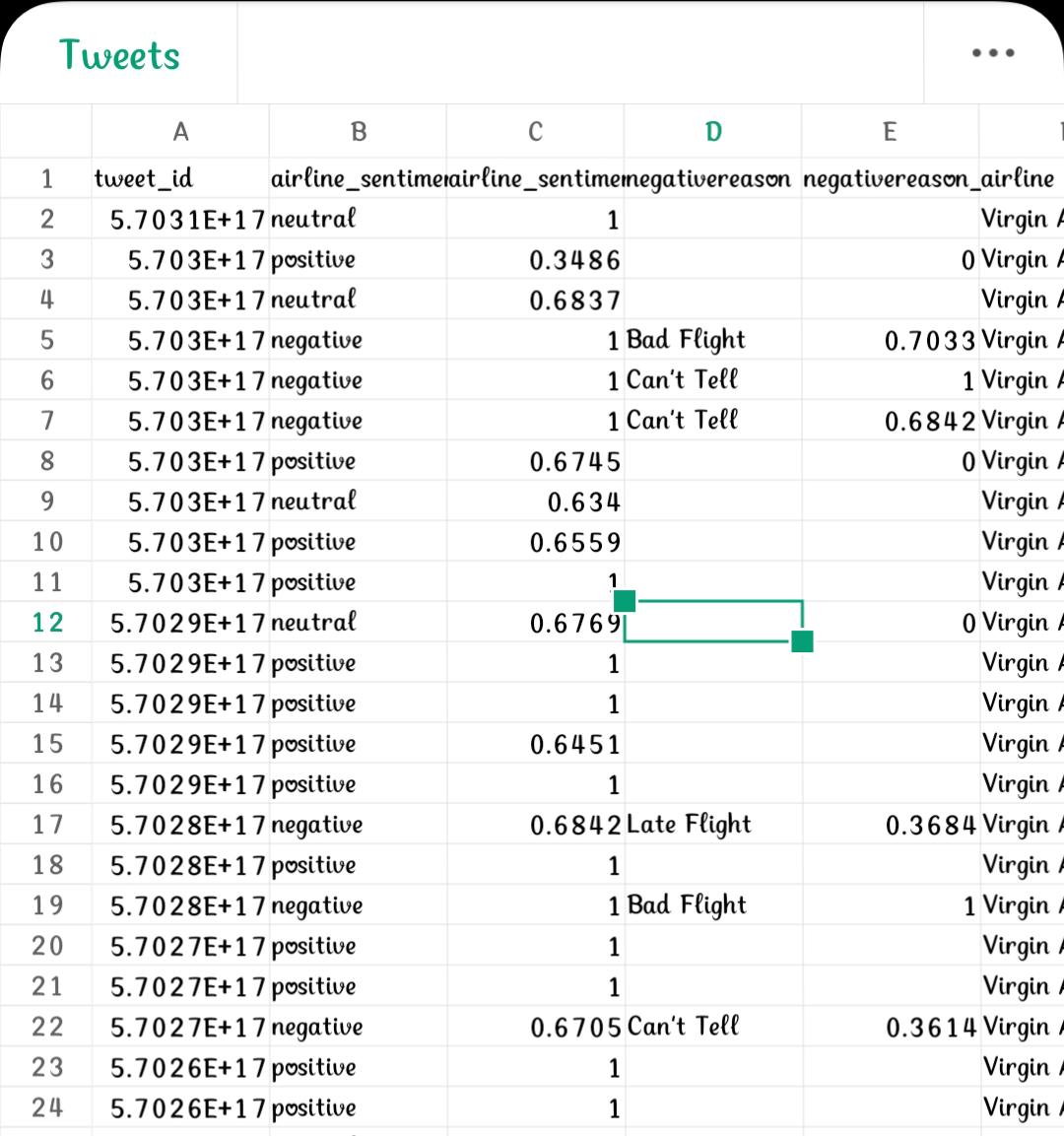
Sentiment analysis for marketing is a valuable technique that involves using natural language processing and machine learning to analyze and understand the emotions, opinions, and attitudes expressed by customers or the public towards a product, brand, or service. By gauging sentiment, marketers can:

* **Customer Feedback Analysis:** Monitor social media, reviews, and customer feedback to gain insights into how customers perceive a brand or product.
* **Competitor Analysis:** Compare sentiment towards your brand with that of competitors to identify strengths and weaknesses.
* **Product Development:** Use sentiment analysis to uncover what features or improvements customers desire in products and services.
* **Campaign Evaluation:** Assess the success of marketing campaigns by analyzing sentiment before and after launch.
* **Reputation Management:** Detect and manage negative sentiment to protect and enhance a brand’s reputation.
* **Customer Segmentation:** Segment customers based on sentiment to tailor marketing strategies and messages.
* Sentiment analysis tools and techniques vary in complexity, from basic sentiment scoring using predefined sentiment lexicons to more advanced machine learning models that can handle nuanced language and context. It empowers marketers to make data-driven decisions, enhance customer engagement, and improve overall marketing strategies.

**DATA SOURCE:**

A good data source for sentiment analysis for marketing should be Accurate, Complete, Coverings the geographic area of interest, Accessible.

Dataset link:(<https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>)



**Social Media Monitoring:**

* Marketers use sentiment analysis tools to monitor social media platforms, forums, and blogs to track what customers are saying about their brand or products. This helps in staying updated with real-time feedback.

**Review and Feedback Analysis:**

* Sentiment analysis can automatically categorize and analyze customer reviews, ratings, and feedback to identify trends and sentiments. Positive reviews can be used for marketing and negative ones for improvement.

**Brand Reputation Management:**

* Marketers can proactively manage their brand’s online reputation by identifying and addressing negative sentiment promptly. This might involve responding to complaints or taking corrective actions.

**Competitor Analysis:**

* By analyzing sentiment towards competitors, marketers can gain insights into their own brand’s strengths and weaknesses and identify opportunities for differentiation.

**Product Development:**

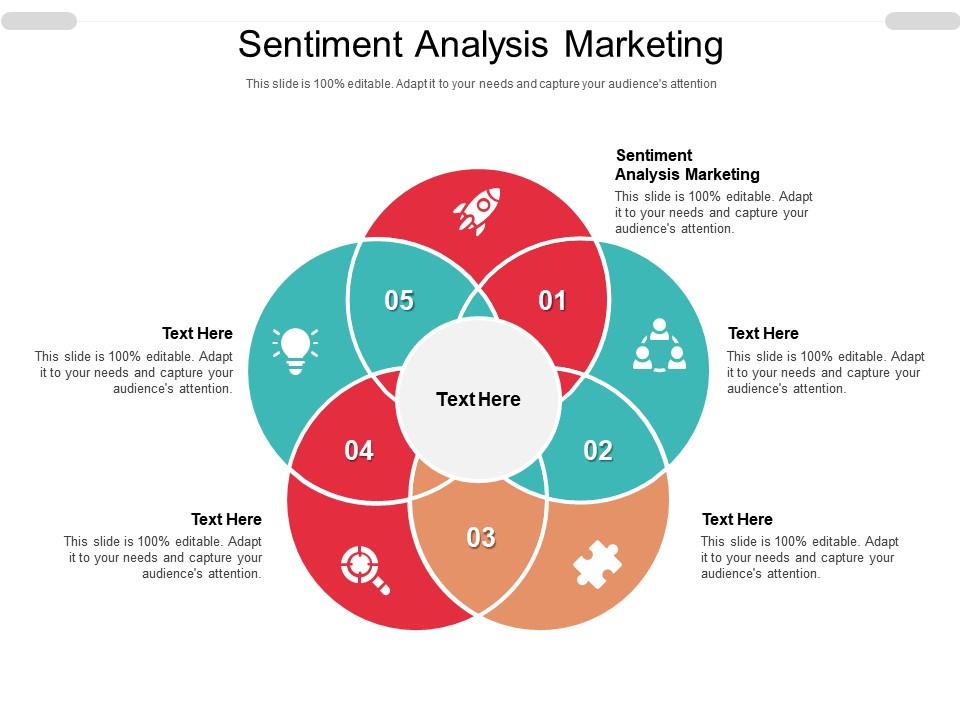
* Sentiment analysis can provide insights into customer preferences, pain points, and feature requests, aiding in product development decisions.

**Campaign Performance Evaluation:**

* Sentiment analysis is used to gauge the success of marketing campaigns. Marketers can monitor sentiment before, during, and after a campaign to assess its impact on customer sentiment and brand perception.

**Reputation Management:**

* Monitoring sentiment allows marketers to proactively manage and respond to negative feedback, helping to protect and enhance a brand’s reputation.

**Program:**

**Sentiment Analysis For Marketing**

**In[1]:**

Import pandas as pd

Import numpy as np

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

From sklearn.feature\_extraction.text import CountVectorizer

From sklearn.ensemble import RandomForestClassifier

From sklearn.metrics import confusion\_matrix, classification\_report

Import matplotlib.pyplot as plt

Import seaborn as sns

From nltk.corpus import stopwords

From nltk.stem import WordNetLemmatizer

Import re

Import nltk

Nltk.download(‘stopwords’)

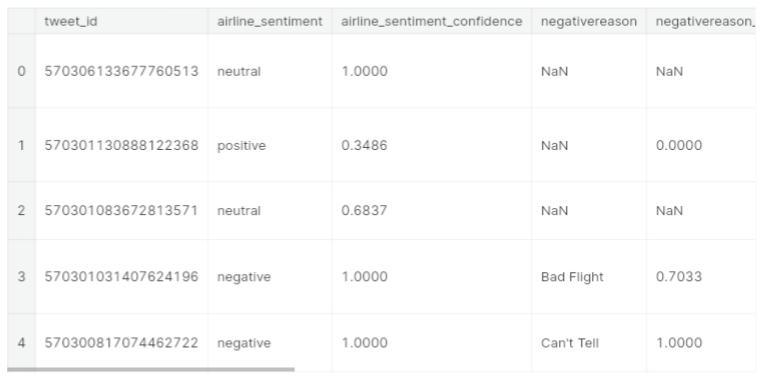
Nltk.download(‘wordnet’)

# Load the dataset

Df = pd.read\_csv(‘/kaggle/input/twitter-airline-sentiment/Tweets.csv’)

# Display the first 5 rows of the dataframe

Df.head()

**Out[1]:**

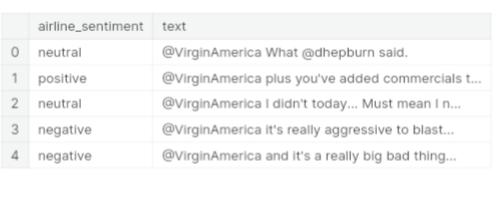
**In[2]:**

*# Drop unnecessary columns*

df = df[['airline\_sentiment', 'text']]

*# Display the first 5 rows of the dataframe after dropping unnecessary columns*

df.head()

**Out[2]:**

**In[3]:**

*# Function to preprocess the text*

def preprocess\_text(text):

*# Remove punctuations and numbers*

text = re.sub('[^a-zA-Z]', ' ', text)

*# Single character removal*

text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)

*# Removing multiple spaces*

text = re.sub(r'\s+', ' ', text)

*# Converting to Lowercase*

text = text.lower()

*# Lemmatization*

*#text = text.split()*

*#lemmatizer = WordNetLemmatizer()*

*#text = [lemmatizer.lemmatize(word) for word in text if not word in set(stopwords.words('english'))]*

*#text = ' '.join(text)*

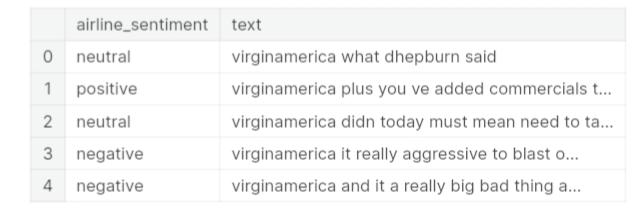
return text

*# Apply the preprocessing to the 'text' column*

df['text'] = df['text'].apply(preprocess\_text)

*# Display the first 5 rows of the dataframe after preprocessing*

df.head()

**Out[3]:**

**In[4]:**

*# Splitting the data into training and testing sets*

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)

*# Feature Extraction*

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()

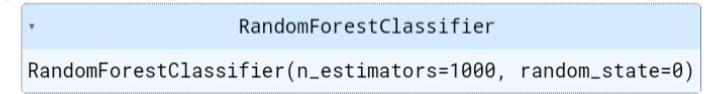
X\_test = vectorizer.transform(X\_test).toarray()

*# Model Training*

from sklearn.ensemble import RandomForestClassifier

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

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

**Out[4]**:

**In[5]:**

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

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

print('Classification Report:')

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

print('Confusion Matrix:')

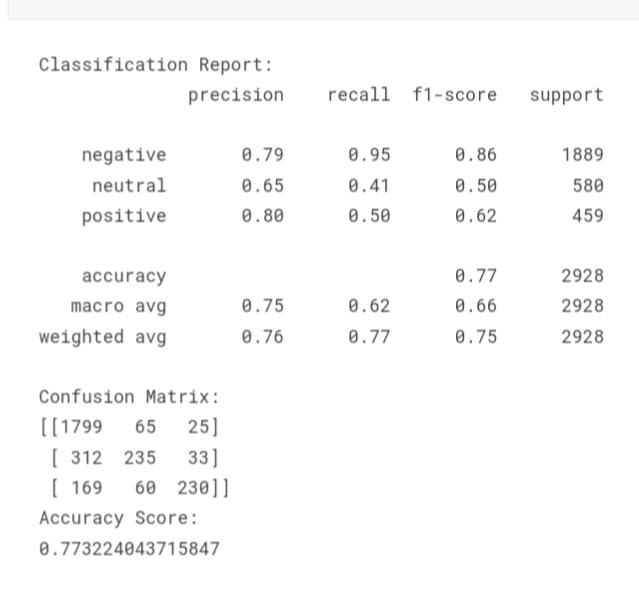
print(confusion\_matrix(y\_test, y\_pred))

print('Accuracy Score:')

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

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

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

**Out[5]:**

**In[6]:**

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')

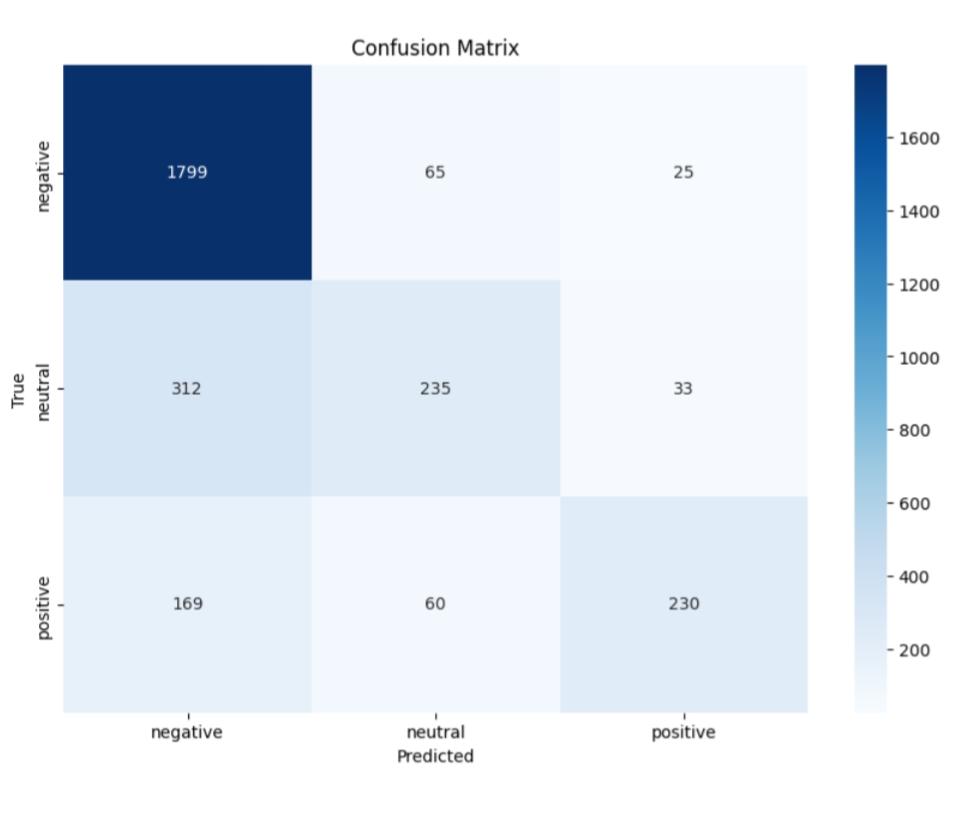
plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

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

**Out[6]: **

**In[7]:**

import seaborn as sns

import matplotlib.pyplot as plt

*# Creating column 'tweet\_length'*

df['tweet\_length'] = df['text'].apply(len)

*# distribution of sentiments*

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

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

plt.title('Distribution of Sentiments')

plt.show()

*# Histogram of tweet lengths*

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

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

plt.title('Distribution of Tweet Lengths')

plt.show()

*# Boxplot of tweet lengths*

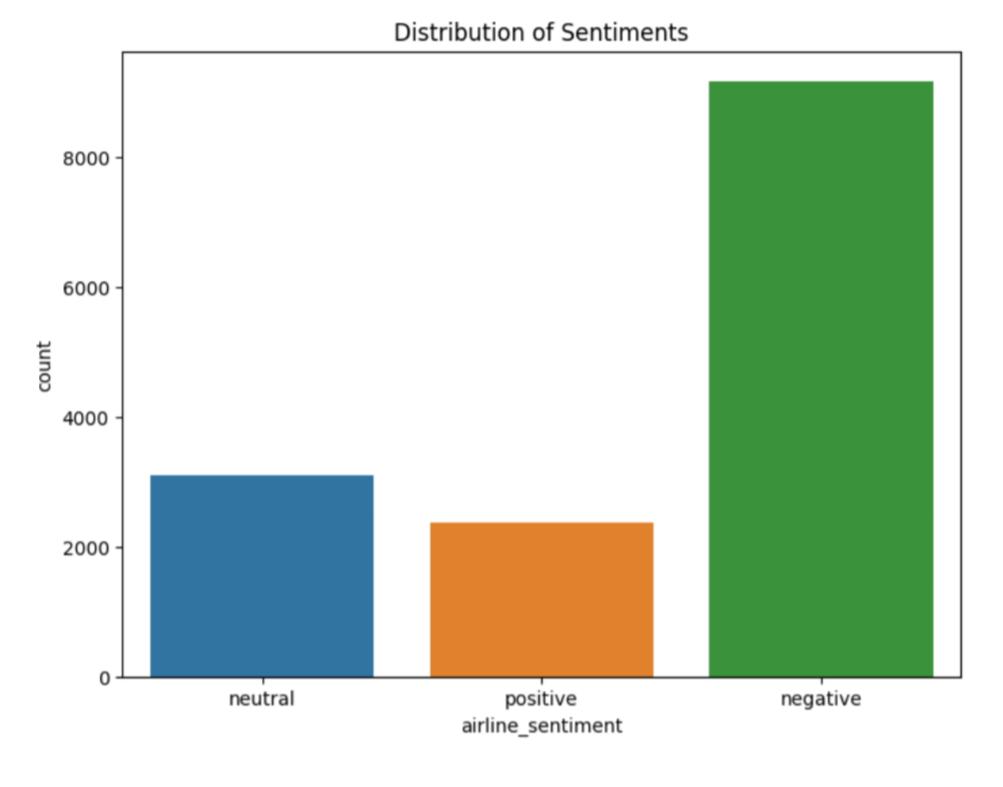
plt.figure(figsize=(8,6))

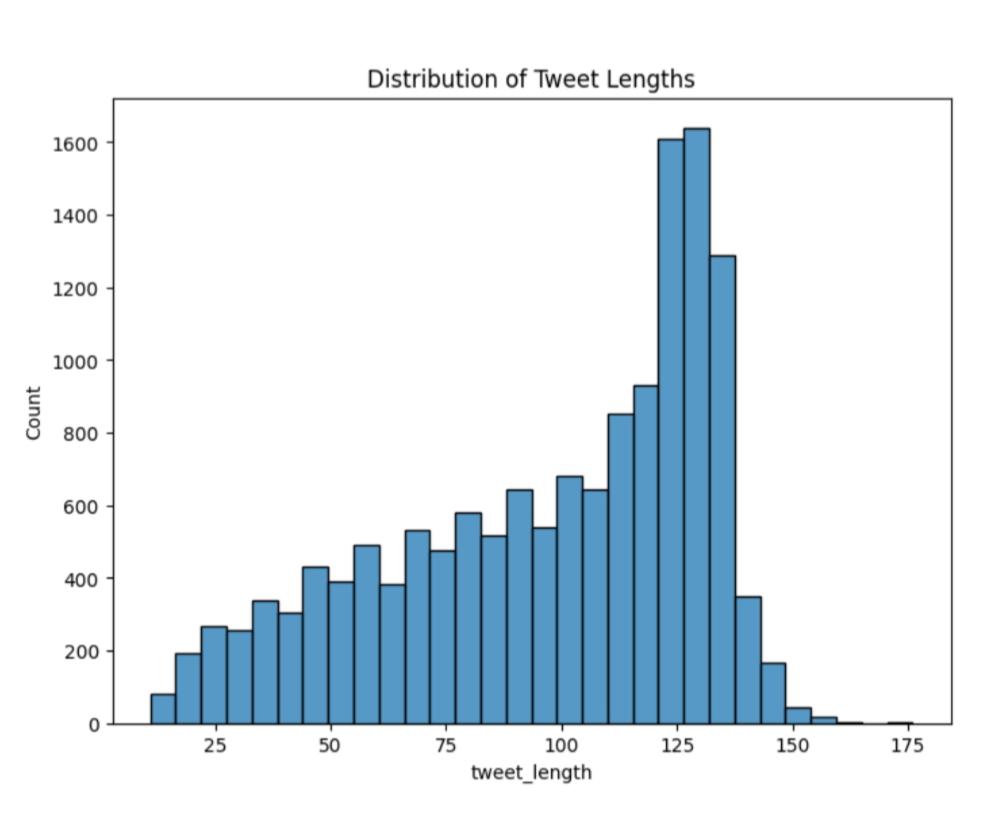
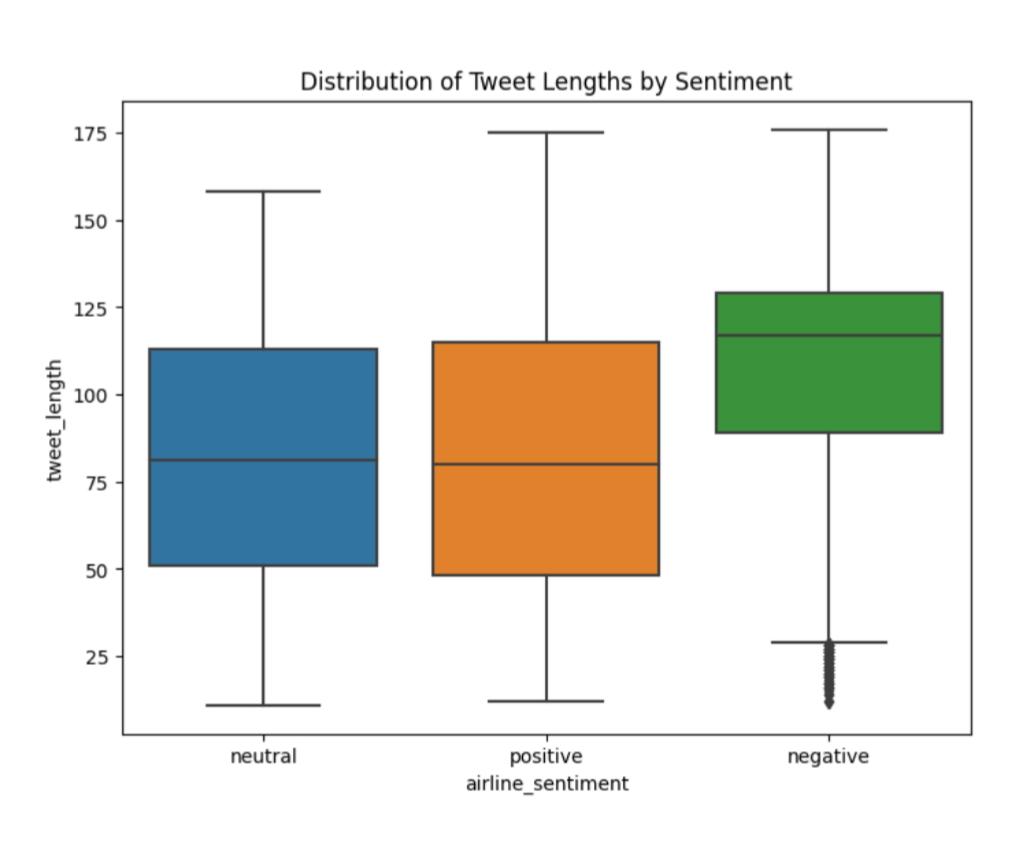
sns.boxplot(x='airline\_sentiment', y='tweet\_length', data=df)

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

plt.show()

**Out[7]:**

****

****

**Critical Analysis**

The following conclusions may be drawn from the visuals and model evaluation:

**Sentiment Distribution:**

The dataset’s bar plot of sentiment distribution reveals that the bulk of tweets are unfavorable in nature, with neutral and supportive tweets coming in second and third. Due to the dataset’s imbalance, the model may be more likely to correctly predict negative feelings than neutral or positive feelings.

**Model Execution:**

The Random Forest classifier’s total accuracy was around 76%. The neutral and positive classes’ accuracy, recall, and F1-score, however, are lower than those of the negative class. This implies that the model performs better at detecting negative than neutral or positive attitudes, which may be related to the dataset’s imbalance.

**Confusion Matrix:**

The confusion matrix reveals that for the neutral and positive classes, the model has a disproportionately large number of false positives and false negatives. This further demonstrates the model’s bias towards predicting negative feelings since it frequently misclassifies neutral and positive tweets as negative.

**Data Distribution:**

Looking at the histogram, it’s obvious, as mentioned before, that there is a significant imbalance in the data in favor of negative sentiment. This is likely because people with negative sentiments are more motivated to tweet. By examining the length distribution in the box plot and the bar chart, we can conclude that the majority of tweets are between 60 to 100 characters long. Negative tweets are usually longer, also falling within the 60 to 100 character range, which further confirms the data imbalance.

**CONCLUSION:**

In conclusion, the model fails to predict neutral and positive attitudes even if it does a fair job of predicting negative sentiments. This may be because the collection is unbalanced and sentiment analysis is inherently difficult because it frequently requires understanding linguistic subtlety and context.