```
# Dataset link: https://huggingface.co/datasets/osanseviero/twitter-airline-sentiment
```

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
```

```
import pandas as pd

df = pd.read_csv("hf://datasets/osanseviero/twitter-airline-sentiment/Tweets.csv")
```

```
df.shape
```

```
(14640, 15)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   tweet_id                      14640 non-null  int64  
 1   airline_sentiment             14640 non-null  object 
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object 
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object 
 6   airline_sentiment_gold        40 non-null     object 
 7   name                          14640 non-null  object 
 8   negativereason_gold           32 non-null     object 
 9   retweet_count                 14640 non-null  int64  
 10  text                          14640 non-null  object 
 11  tweet_coord                   1019 non-null   object 
 12  tweet_created                 14640 non-null  object 
 13  tweet_location                9907 non-null   object 
 14  user_timezone                 9820 non-null   object 
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

```
df.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retweet_count
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	0
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN	0
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN	0
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	NaN	0

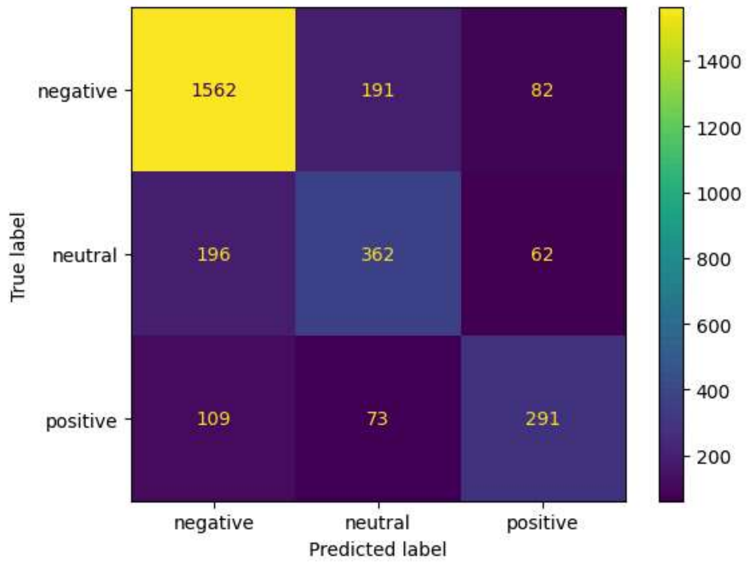
```
print(df['airline_sentiment'].value_counts())
```

```
airline_sentiment
negative    9178
neutral     3099
positive    2363
Name: count, dtype: int64
```

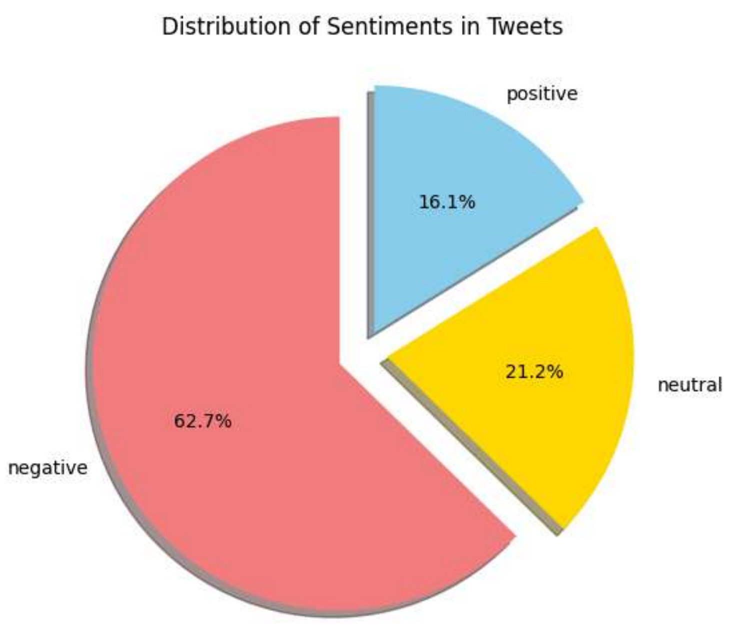
```
# Import necessary libraries
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

# Download necessary NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')

# Define text cleaning function
def clean_text(text):
    text = re.sub(r'@\w+|#\w+|http\S+|www\S+', '', text)  # Remove mentions, hashtags, and URLs
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters and numbers
    text = text.lower()  # Convert to lowercase
    tokens = text.split()  # Tokenize
    tokens = [word for word in tokens if word not in stopwords.words('english')]  # Remove stopwords
    lemmatizer = WordNetLemmatizer()  # Initialize lemmatizer
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmatize tokens
    return ' '.join(tokens)

# Apply cleaning function to the text column
df['cleaned_text'] = df['text'].apply(clean_text)

# Split the dataset
X = df['cleaned_text']
y = df['airline_sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
```

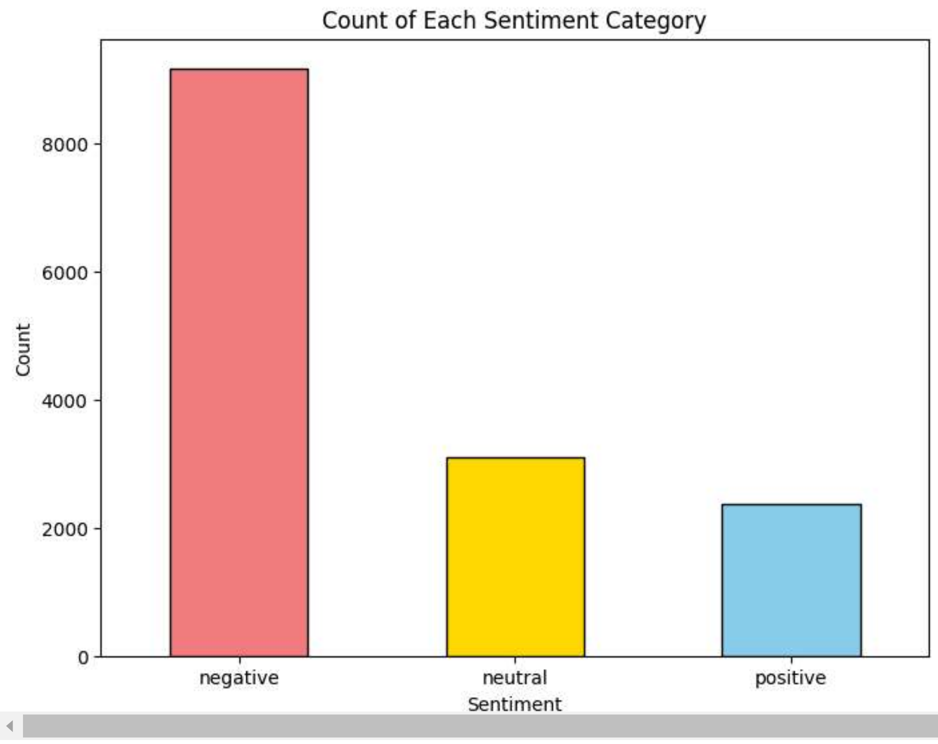
```
X_test_vec = vectorizer.transform(X_test)

# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_vec, y_train)

# Train a Random Forest model
clf = RandomForestClassifier(class_weight='balanced', random_state=42)
clf.fit(X_train_resampled, y_train_resampled)

# Make predictions
y_pred = clf.predict(X_test_vec)

# Evaluate the model
print(classification_report(y_test, y_pred))

# Display the confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=['negative', 'neutral', 'positive'])
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['negative', 'neutral', 'positive'])
disp.plot(cmap='viridis')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
              precision    recall  f1-score   support

    negative       0.84      0.85      0.84      1835
     neutral       0.58      0.58      0.58       620
    positive       0.67      0.62      0.64       473

    accuracy                           0.76      2928
   macro avg       0.69      0.68      0.69      2928
weighted avg       0.75      0.76      0.76      2928

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7eef215d6650>
```




```
import matplotlib.pyplot as plt

# Count the number of tweets in each sentiment category
sentiment_counts = df['airline_sentiment'].value_counts()

# Plot pie chart
plt.figure(figsize=(8, 6))
sentiment_counts.plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    colors=['lightcoral', 'gold', 'skyblue'],
    explode=(0.1, 0.1, 0.1),
    shadow=True
)
plt.title('Distribution of Sentiments in Tweets')
plt.ylabel('')  # Remove the y-label
plt.show()

# Plot bar chart
plt.figure(figsize=(8, 6))
sentiment_counts.plot.bar(
    color=['lightcoral', 'gold', 'skyblue'],
    edgecolor='black'
)
plt.title('Count of Each Sentiment Category')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```






```
from sklearn.metrics import classification_report, accuracy_score
import numpy as np
nltk.download('vader_lexicon')  # Download VADER lexicon if not already present
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Assuming y_test and y_pred are already defined (test labels and predicted labels)

# Compare predictions with actual labels
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the model: {accuracy:.2f}")

# Detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Find the most negative and most positive sentiment tweets
df['sentiment_polarity'] = df['cleaned_text'].apply(lambda x: analyzer.polarity_scores(x)['compound'])

# Most negative sentiment tweet
most_negative_tweet = df[df['sentiment_polarity'] == df['sentiment_polarity'].min()]
print("\nMost Negative Tweet:")
print(most_negative_tweet[['text', 'sentiment_polarity']])

# Most positive sentiment tweet
most_positive_tweet = df[df['sentiment_polarity'] == df['sentiment_polarity'].max()]
print("\nMost Positive Tweet:")
print(most_positive_tweet[['text', 'sentiment_polarity']])
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
Accuracy of the model: 0.76

Classification Report:
              precision    recall  f1-score   support

    negative       0.84      0.85      0.84      1835
     neutral       0.58      0.58      0.58       620
    positive       0.67      0.62      0.64       473

    accuracy                           0.76      2928
   macro avg       0.69      0.68      0.69      2928
weighted avg       0.75      0.76      0.76      2928


Most Negative Tweet:
                                                   text  sentiment_polarity
1214  @united is the worst. Worst reservation polici...             -0.9792

Most Positive Tweet:
                                                   text  sentiment_polarity
4511  @SouthwestAir I love this airline so much! Tha...              0.9716
8922  @JetBlue huge fan of great brands and people d...              0.9716
```

```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from wordcloud) (1.26.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from wordcloud) (11.0.0)
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from wordcloud) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

```python
# prompt: create word cloud for positive and negative sentiment

from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame with 'cleaned_text' and 'airline_sentiment' columns

# Separate positive and negative tweets
positive_tweets = df[df['airline_sentiment'] == 'positive']['cleaned_text']
negative_tweets = df[df['airline_sentiment'] == 'negative']['cleaned_text']

# Create word clouds
positive_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(" ".join(positive_tweets))
negative_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(" ".join(negative_tweets))

# Plot the word clouds
plt.figure(figsize=(20, 10))

plt.subplot(1, 2, 1)
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Positive Sentiment Word Cloud")

plt.subplot(1, 2, 2)
plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Negative Sentiment Word Cloud")

plt.show()
```

Positive Sentiment Word Cloud

Negative Sentiment Word Cloud