Sentiment analysis for marketing

Problem statement:

The goal of project is to use the sentiment analysis for marketing, the core problem revolves around accurately and effectively assessing and interpreting customer sentiment from various data sources. It developing models and Techniques to classify customer sentiments into categories such as positive, negative, neutral or more granular emotions based on Textual or visual data like social media posts, product reviews, and customer feedback.

Design thinking:

Design thinking is a problem-solving approach that emphasizes empathy, ideation, and prototyping to create innovative solutions. When applied to sentiment analysis in marketing, it can help in developing effective strategies for understanding and leveraging customer emotions.

When applied to sentiment analysis using Natural Language Processing (NLP), it can help in developing effective strategies for understanding and leveraging customer emotions.

Says:

Examine social media comments, reviews, and tweets about your brand.
Gather feedback from online platforms like Amazon, twitter, or Google Reviews.

Think and SaysFeel:

Identify their emotional state when interacting with your brand (happy, frustrated, excited, disappointed).
Determine their challenges, concerns, or problems related to your product or service.

Hear:

Determine the tone of their language (positive, negative, neutral)

See:

Analyze their response to your advertisements, both online and offline.
Understand how they perceive your brand in comparison to competitors.

Do:

Outline their actions, such as purchasing, visiting your website, engaging on social media.
Determine what influences their decisions (online reviews, expert opinions).

Pain:

Analyze customer complaints and negative feedback.

Gains:

Analyze positive reviews and testimonials from satisfied customers.

Phase of development:

The development phases for sentiment analysis in marketing using AI can be broken down as follows:

- **Project Planning and Goal Setting**:
- **Data Collection and Preparation**:
- **Model Selection and Training**:
- **Model Evaluation**:
- **Integration with Marketing Platforms**:
- **Real-time Analysis and Reporting**:
- **Feedback Loop and Model Maintenance**:
- **Scalability and Optimization**:
- **Ethical Considerations and Bias Mitigation**:
- **Compliance and Data Privacy**:
- **Documentation and Knowledge Transfer**:
- **Continuous Improvement and Innovation**:

Overview comprehension:

- A Twitter sentiment analysis determines negative, positive, or neutral emotions within the text of a tweet using NLP and ML models. Sentiment analysis or opinion mining refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of people on social media for a variety of topics.
- Twitter sentiment analysis analyses the sentiment or emotion of tweets. It uses natural language processing and machine learning algorithms to classify tweets automatically as positive, negative, or neutral based on their content. It can be done for individual tweets or a larger dataset related to a particular topic or event.

Importance of analysis:

1. **Understanding Customer Feedback:** By analysing the sentiment of customer feedback, companies can identify areas where they need to improve their products or services.
2. **Reputation Management**: Sentiment analysis can help companies monitor their brand reputation online and quickly respond to negative comments or reviews.
3. **Political Analysis**: Sentiment analysis can help political campaigns understand public opinion and tailor their messaging accordingly.
4. **Crisis Management:** In the event of a crisis, sentiment analysis can help organizations monitor social media and news outlets for negative sentiment and respond appropriately.
5. **Marketing Research:** Sentiment analysis can help marketers understand consumer behaviour and preferences, and develop targeted advertising campaigns.

Dataset:

Data Source URL:https://www.kaggle.com/datasets/tweets dataset/market-based-sentiment-analysis

Project detail:

- **Code Structure:** The Jupyter Notebook is structured into well-organized code cells, each serving a specific purpose. The code is documented with comments, explaining the functionality of each cell and the overall logic of the Data Cleaning and further processes.
- **Updated Dataset:** As the original dataset cannot also be clean enough in order take that dataset into Data Analysis Phase, I have cleaned the dataset and stored the cleaned dataset as an “updated_dataset” in xlsx file extension.

Step 1: Install pandas and numpy

```
import pandas as pd

import numpy as np
```

step2: CountVectorizer is a feature extraction technique used in natural language processing (NLP) and text mining to convert a collection of text documents into a numerical feature matrix. It is a part of the scikit-learn library in Python and is used to transform a set of text data to

numerical feature vectors. Each feature represents the frequency of a word in the given text.

```
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
```

step3: WordNetLemmatizer is a part of the NLTK (Natural Language Toolkit) library in Python. It is used for lemmatizing words, which means reducing a word to its base or root form.

```
import matplotlib.pyplot as plt

import seaborn as sns

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import re

import nltk
```

step4: Upload dataset file(twitter airlines US)

```
from google.colab import files

data=files.upload()
```

step5: In NLTK (Natural Language Toolkit) and other NLP libraries in Python, stopwords can be easily accessed and used. Here's how you can

use stopwords using NLTK:

```
nltk.download('stopwords') nltk.download('wordnet')
```

step6: Load the dataset

```
df = pd.read_csv('Tweets.csv')
```

step7: Display the frames

```
# Display the first 5 rows of the dataframe
df.head()
```

```
tweet_id airline_sentiment airline_sentiment_confidence negativer
0 570306133677760513
1 570301130888122368
neutral
1.0000
positive
0.3486
```

step8: Install nltk (natural processing language)

```
!pip install nltk
```

```
Requirement already satisfied: nltk in
/usr/local/lib/python3.10/dist-packages (3.8.1)
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) (1.3.2)
```

```
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in
/usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

step9: To drop unnecessary columns from a DataFrame in Python, you can use the drop() method provided by pandas, a powerful data manipulation library. Here's how you can do it

```
# Drop unnecessary columns

df =

df[['airline_sentiment', 'text']]

# Display the first 5 rows of the dataframe after dropping unnecessary
columns df.head(125)
```

```
airline_sentiment

text

0

neutral

virginamerica what dhepburn said

1

positive

virginamerica plus you ve added commercials t...

2

neutral

virginamerica didn today must mean need to ta...

3

negative

4

negative

virginamerica it really aggressive to blast o...
```

virginamerica and it a really big bad thing a...

120

121

122

negative

negative virginamerica use another browser amp brand w...

negative virginamerica and now the flight flight booki...

virginamerica like the customer service but m...

123

positive

124

positive

virginamerica thanks to your outstanding nyc ..

virginamerica you have the absolute best team...

125 rows × 2 columns

TT B I <>

step10:

The preprocess_tweet function performs various preprocessing steps on each tweet.

df['text'] refers to the column containing the original tweets, and df ['preprocessed_text'] is the column where preprocessed tweets will be stored.

# function to preprocess the text

def preprocess_text(text):

```
# Remove punctuations and numbers


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

# Single character removal

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

# Removing multiple spaces

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

# Converting to Lowercase

text = text.lower()

# Lemmatization

text =text.split()


WordNetLemmatizer()

#lemmatizer ', text)


#text = [lemmatizer.lemmatize (word) for word in text if not word in
set(stop

#text = return text '.join(text)
```

finally, after preprocessing the dataset,

```
# 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()
```

```
tweet_id airline_sentiment airline_sentiment_confidence negativereason
negativereason_confidence airline airline_sen

0 570306133677760513

neutral

1.0000
```

NaN

NaN

Virgin America

1 570301130888122368

positive

0.3486

NaN

0.0000

Virgin America

2 570301083672813571

neutral

0.6837

NaN

NaN

Virgin America

3 570301031407624196

negative

1.0000

Bad Flight

0.7033

Virgin America

4 570300817074462722

negative

1.0000

Can't Tell

1.0000

Viririgin America

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.

- 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:

```
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:

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.

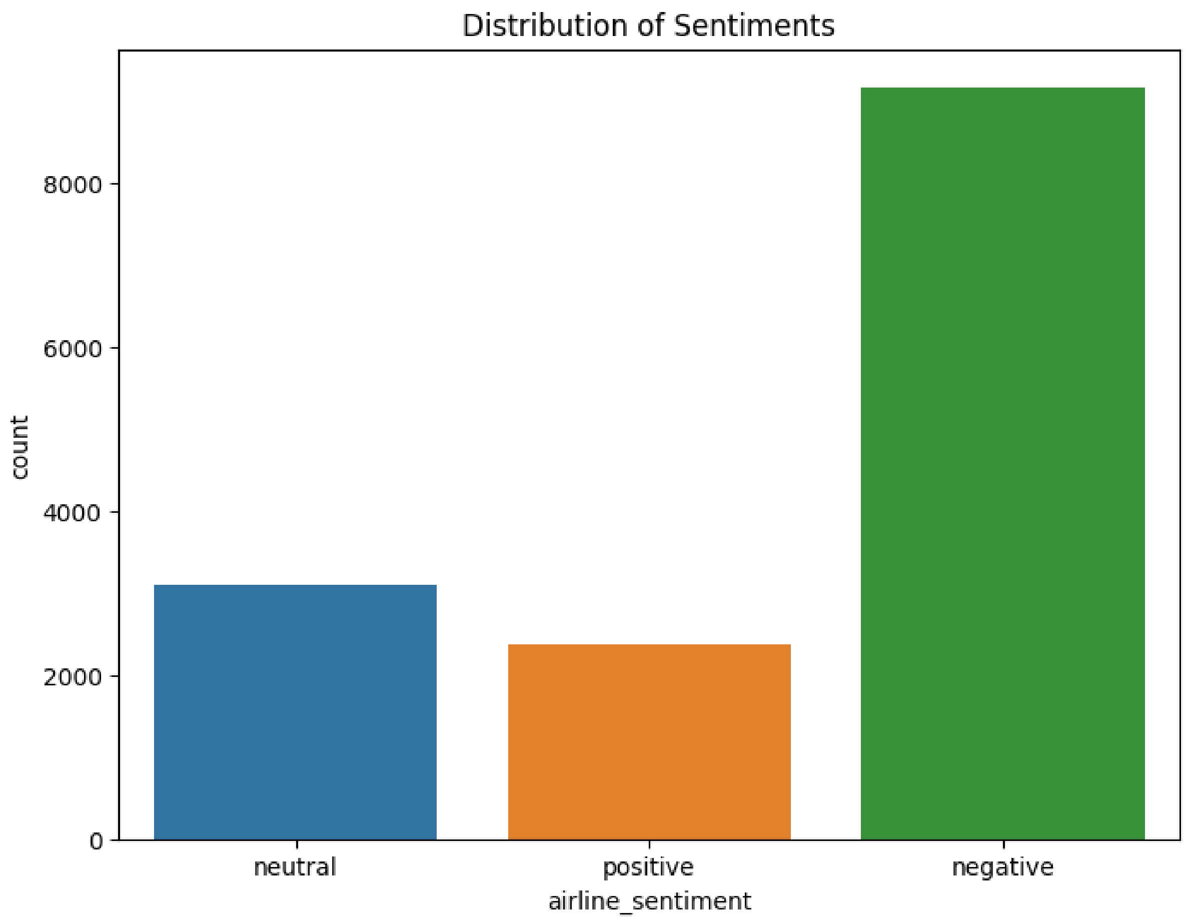
```
from sklearn.ensemble
import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
```

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.

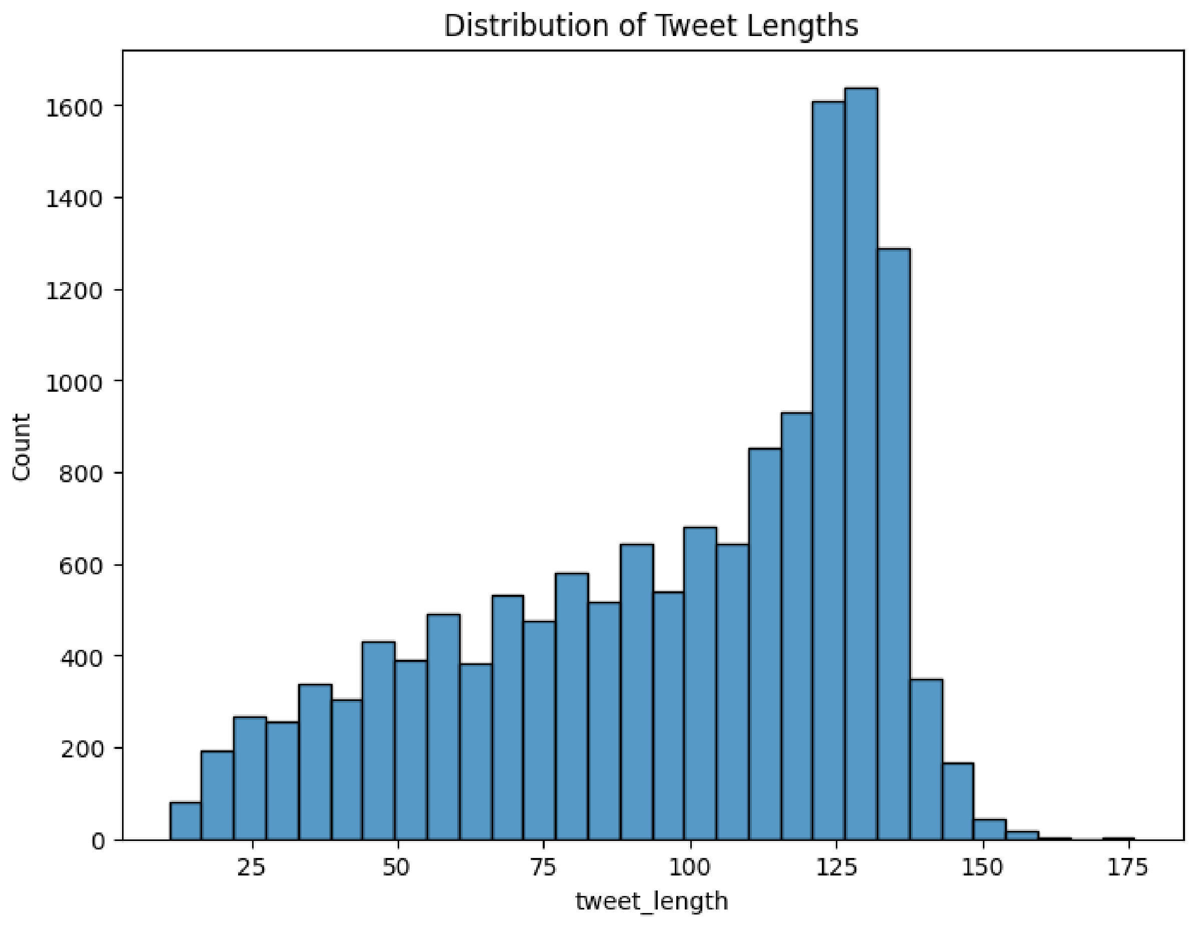
```
plt.figure(figsize=(8,6))
sns.countplot(x='airline_sentiment', data=df)
plt.title('Distribution of Sentiments')
plt.show()
```



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.

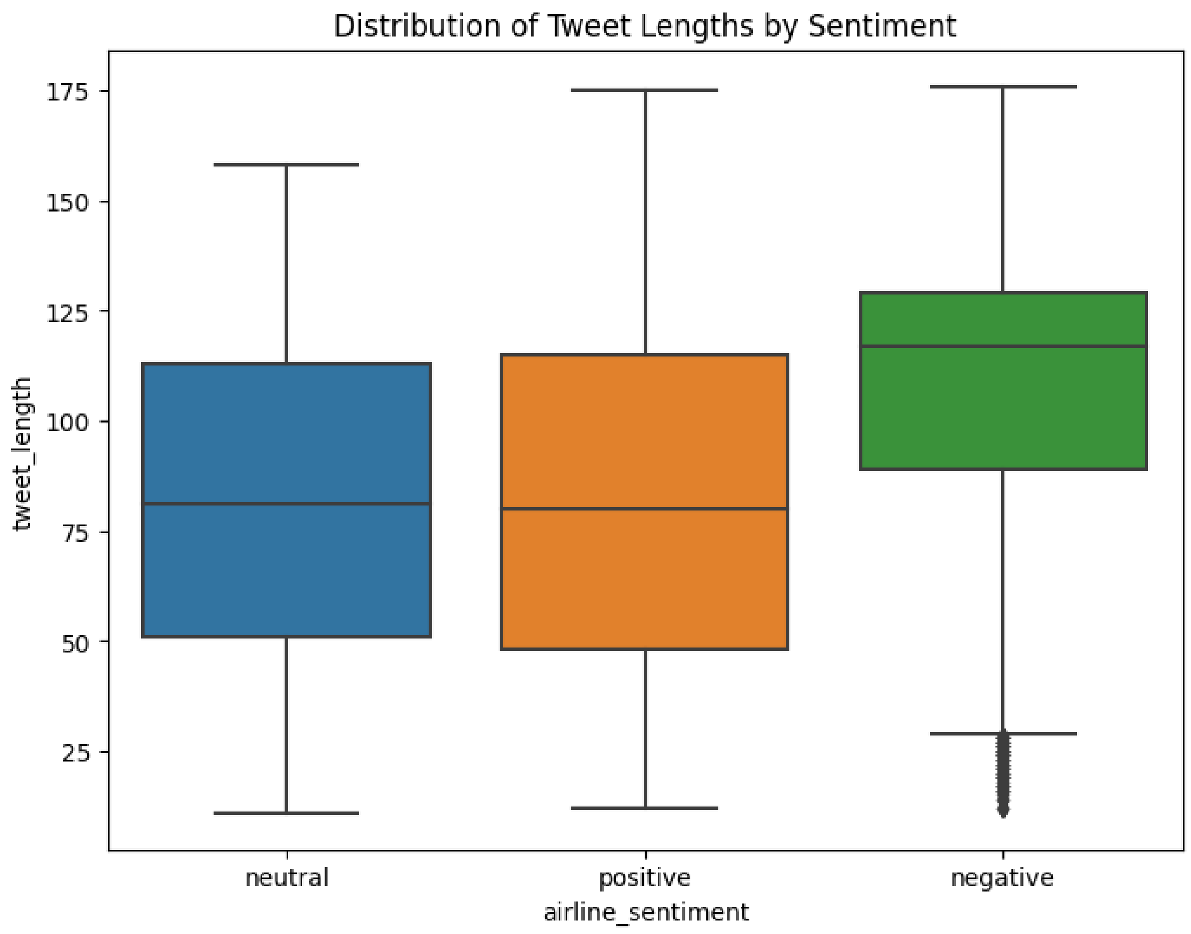
```
plt.figure(figsize=(8,6))
sns.histplot(df['tweet_length'], bins=30)
plt.title('Distribution of Tweet Lengths')
plt.show()
```



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))
```

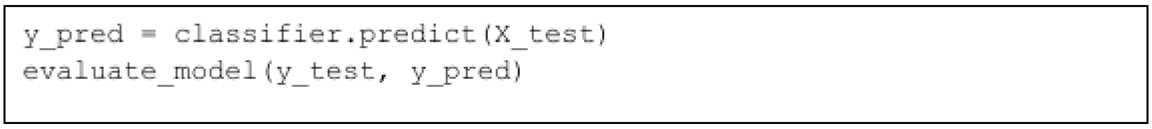
Output:

```
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:

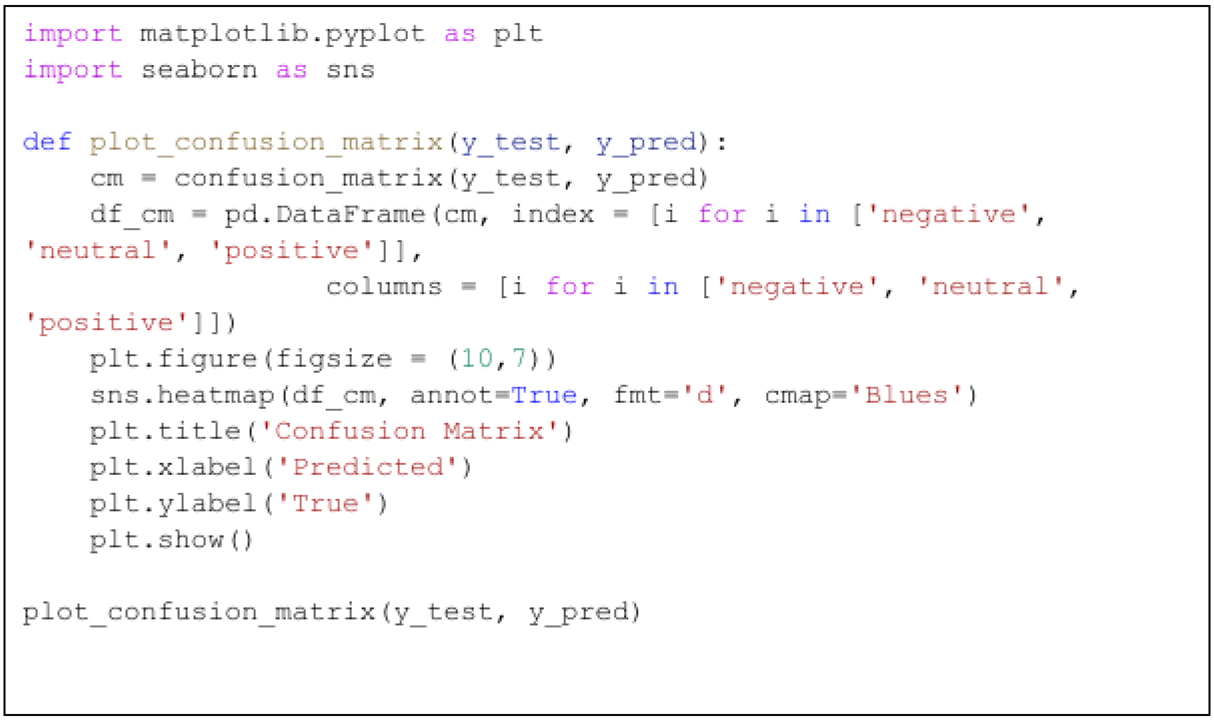
```
Accuracy Score:
0.773224043715847
```

Step 16:

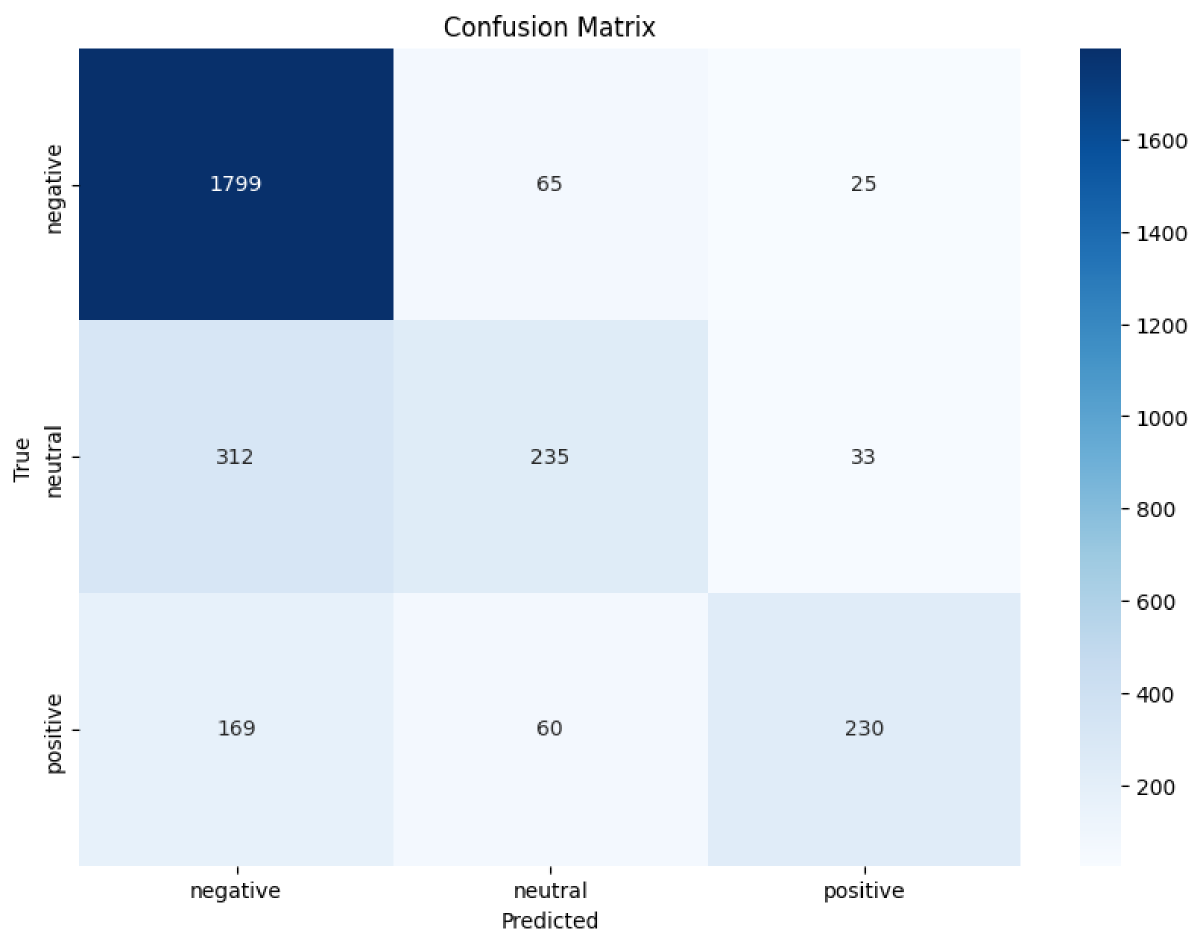
Confusion matrix:

- 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)
```


Confusion Matrix
True
Predicted
negative
neutral
positive
1799
65
25
312
235
33
169
60
230
1600
1400
1200
1000
800
600
400
200