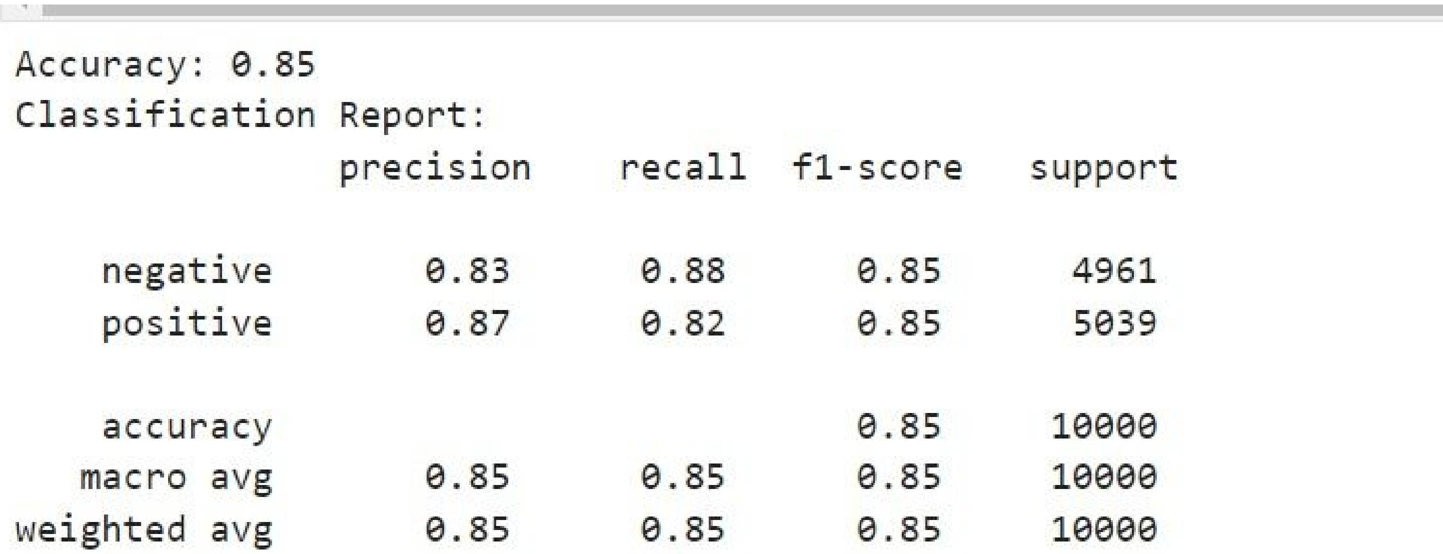

```
Accuracy: 0.85
Classification Report:
              precision    recall  f1-score   support

    negative       0.83      0.88      0.85      4961
    positive       0.87      0.82      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Sample dataset with labeled documents (positive or negative)
data = pd.read_csv('G:/Games/preseed/sample.csv')

# List of polarity words
positive_words = ["good", "happy", "excellent", "awesome", "amazing"]
negative_words = ["bad", "sad", "terrible", "awful", "disappointing"]

# Function to classify documents based on polarity words
def classify_document(document):
    pos_count = sum(1 for word in document.split() if word in positive_words)
    neg_count = sum(1 for word in document.split() if word in negative_words)
    if pos_count > neg_count:
        return 'positive'
    elif neg_count > pos_count:
        return 'negative'
    else:
        return 'neutral'

# Apply classification to the dataset
data['predicted_sentiment'] = data['review'].apply(classify_document)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['review'], data['sentiment'], test_size=0.2, random_state=42)
```

```
# Create a CountVectorizer to convert text data to numerical features
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# Train a Multinomial Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train_counts, y_train)

# Predict document sentiment
y_pred = classifier.predict(X_test_counts)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy:.2f}')
print('Classification Report:\n', report)
```