Project Name:

Sentiment Analysis on IMDB Movie Reviews Using Naive Bayes

Project Summary:

This project focuses on building a binary text classification model to determine the sentiment (positive or negative) of IMDB movie reviews. Sentiment Analysis is crucial for businesses to understand customer feedback and improve services accordingly. Using the IMDB dataset with 50,000 labeled movie reviews, we preprocess the data, vectorize the text, train a Naive Bayes model, and evaluate its performance using accuracy, precision, recall, F1-score, and a confusion matrix.

Step 1: Import Libraries

- Load the necessary libraries for data manipulation, visualization, and machine learning tasks.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings as warning
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
warning.filterwarnings('ignore')
```

Step 2: Load the Dataset

- Load the dataset containing movie reviews and their corresponding sentiments.

```
df = pd.read_csv('/content/IMDB Dataset.csv', on_bad_lines='skip',
quoting=0)

print(df.columns.tolist())


['review', 'sentiment']

df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 50000,\n  \"fields\":
[\n    {\n      \"column\": \"review\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 49582,\n
\"samples\": [\n          \"\\\"Soul Plane\\\" is a horrible attempt
at comedy that only should appeal people with thick skulls, bloodshot
eyes and furry pawns. <br /><br />The plot is not only incoherent but
also non-existent, acting is mostly sub sub-par with a gang of highly
moronic and dreadful characters thrown in for bad measure, jokes are
often spotted miles ahead and almost never even a bit amusing. This
movie lacks any structure and is full of racial stereotypes that must
have seemed old even in the fifties, the only thing it really has
going for it is some pretty ladies, but really, if you want that you
can rent something from the \\\"Adult\\\" section. OK?<br /><br />I
can hardly see anything here to recommend since you'll probably have a
lot a better and productive time chasing rats with a sledgehammer or
inventing waterproof teabags or whatever.<br /><br />2/10\",\n
\"Guest from the Future tells a fascinating story of time travel,
friendship, battle of good and evil -- all with a small budget, child
actors, and few special effects. Something for Spielberg and Lucas to
learn from. ;) A sixth-grader Kolya \\\"Nick\\\" Gerasimov finds a
time machine in the basement of a decrepit building and travels 100
years into the future. He discovers a near-perfect, utopian society
where robots play guitars and write poetry, everyone is kind to each
other and people enjoy everything technology has to offer. Alice is
the daughter of a prominent scientist who invented a device called
Mielophone that allows to read minds of humans and animals. The device
can be put to both good and bad use, depending on whose hands it falls
into. When two evil space pirates from Saturn who want to rule the
universe attempt to steal Mielophone, it falls into the hands of 20th
century school boy Nick. With the pirates hot on his tracks, he
travels back to his time, followed by the pirates, and Alice. Chaos,
confusion and funny situations follow as the luckless pirates try to
blend in with the earthlings. Alice enrolls in the same school Nick
goes to and demonstrates superhuman abilities in PE class. The catch
is, Alice doesn't know what Nick looks like, while the pirates do.
Also, the pirates are able to change their appearance and turn
literally into anyone. (Hmm, I wonder if this is where James Cameron
got the idea for Terminator...) Who gets to Nick -- and Mielophone --
first? Excellent plot, non-stop adventures, and great soundtrack. I
wish Hollywood made kid movies like this one...\",\n
\"\\\"National Treasure\\\" (2004) is a thoroughly misguided hodge-
podge of plot entanglements that borrow from nearly every cloak and
dagger government conspiracy clich\\u00e9 that has ever been written.
The film stars Nicholas Cage as Benjamin Franklin Gates (how precious
is that, I ask you?); a seemingly normal fellow who, for no other
reason than being of a lineage of like-minded misguided fortune
hunters, decides to steal a 'national treasure' that has been hidden
by the United States founding fathers. After a bit of subtext and
background that plays laughably (unintentionally) like Indiana Jones
meets The Patriot, the film degenerates into one misguided whimsy
after another \\u0096 attempting to create a 'Stanley Goodspeed'
regurgitation of Nicholas Cage and launch the whole convoluted mess
forward with a series of high octane, but disconnected
misadventures.<br /><br />The relevancy and logic to having George
Washington and his motley crew of patriots burying a king's ransom
someplace on native soil, and then, going through the meticulous plan
of leaving clues scattered throughout U.S. currency art work, is
something that director Jon Turteltaub never quite gets around to
explaining. Couldn't Washington found better usage for such wealth
during the start up of the country? Hence, we are left with a mystery
built on top of an enigma that is already on shaky ground by the time
Ben appoints himself the new custodian of this untold wealth. Ben's
intentions are noble \\u0096 if confusing. He's set on protecting the
treasure. For who and when?\\u0085your guess is as good as mine.<br
/><br />But there are a few problems with Ben's crusade. First up, his
friend, Ian Holmes (Sean Bean) decides that he can't wait for Ben to
make up his mind about stealing the Declaration of Independence from
the National Archives (oh, yeah \\u0096 brilliant idea!). Presumably,
the back of that famous document holds the secret answer to the
ultimate fortune. So Ian tries to kill Ben. The assassination attempt
is, of course, unsuccessful, if overly melodramatic. It also affords
Ben the opportunity to pick up, and pick on, the very sultry curator
of the archives, Abigail Chase (Diane Kruger). She thinks Ben is
clearly a nut \\u0096 at least at the beginning. But true to
action/romance form, Abby's resolve melts quicker than you can say,
\\\"is that the Hope Diamond?\\\" The film moves into full X-File-ish
mode, as the FBI, mistakenly believing that Ben is behind the theft,
retaliate in various benign ways that lead to a multi-layering of
action sequences reminiscent of Mission Impossible meets The Fugitive.
Honestly, don't those guys ever get 'intelligence' information that is
correct? In the final analysis, \\\"National Treasure\\\" isn't great
film making, so much as it's a patchwork rehash of tired old bits from
other movies, woven together from scraps, the likes of which would
make IL' Betsy Ross blush.<br /><br />The Buena Vista DVD delivers a
far more generous treatment than this film is deserving of. The
anamorphic widescreen picture exhibits a very smooth and finely
detailed image with very rich colors, natural flesh tones, solid
blacks and clean whites. The stylized image is also free of blemishes
and digital enhancements. The audio is 5.1 and delivers a nice sonic
boom to your side and rear speakers with intensity and realism. Extras
include a host of promotional junket material that is rather deep and
over the top in its explanation of how and why this film was made. If
only, as an audience, we had had more clarification as to why Ben and
co. were chasing after an illusive treasure, this might have been one
good flick. Extras conclude with the theatrical trailer, audio
commentary and deleted scenes. Not for the faint-hearted \\u0096 just
the thick-headed.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
```

[\n   \"negative\",\n   \"positive\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.tail()
```

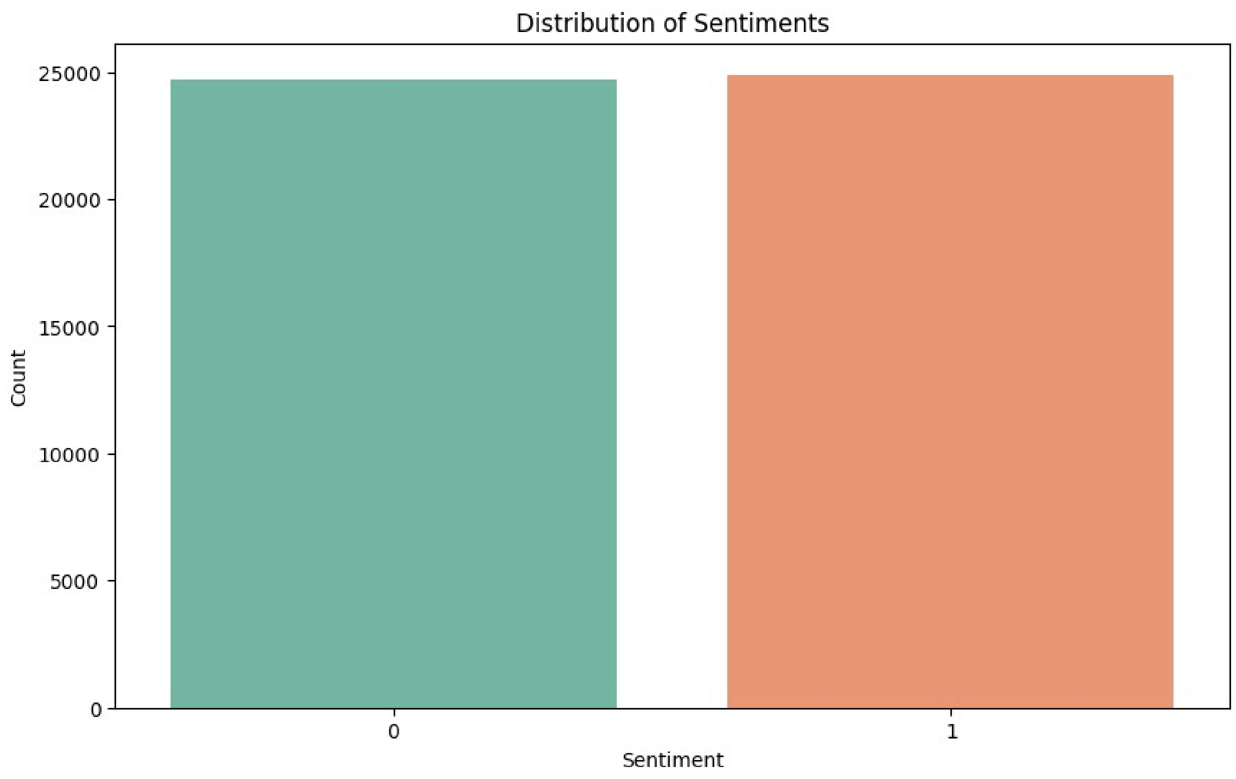
{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"review\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"Bad plot, bad dialogue, bad acting, idiotic directing, the annoying porn groove soundtrack that ran continually over the overacted script, and a crappy copy of the VHS cannot be redeemed by consuming liquor. Trust me, because I stuck this turkey out to the end. It was so pathetically bad all over that I had to figure it was a fourth-rate spoof of Springtime for Hitler.<br /><br />The girl who played Janis Joplin was the only faint spark of interest, and that was only because she could sing better than the original.<br /><br />If you want to watch something similar but a thousand times better, then watch Beyond The Valley of The Dolls.\",\n   \"No one expects the Star Trek movies to be high art, but the fans do expect a movie that is as good as some of the best episodes. Unfortunately, this movie had a muddled, implausible plot that just left me cringing - this is by far the worst of the nine (so far) movies. Even the chance to watch the well known characters interact in another movie can't save this movie - including the goofy scenes with Kirk, Spock and McCoy at Yosemite.<br /><br />I would say this movie is not worth a rental, and hardly worth watching, however for the True Fan who needs to see all the movies, renting this movie is about the only way you'll see it - even the cable channels avoid this movie.\",\n   \"I am a Catholic taught in parochial elementary schools by nuns, taught by Jesuit priests in high school & college. I am still a practicing Catholic but would not be considered a \\\"good Catholic\\\" in the church's eyes because I don't believe certain things or act certain ways just because the church tells me to.<br /><br />So back to the movie...its bad because two people are killed by this nun who is supposed to be a satire as the embodiment of a female religious figurehead. There is no comedy in that and the satire is not done well by the over acting of Diane Keaton. I never saw the play but if it was very different from this movies then it may be good.<br /><br />At first I thought the gun might be a fake and the first shooting all a plan by the female lead of the four former students as an attempt to demonstrate Sister Mary's emotional and intellectual bigotry of faith. But it turns out the bullets were real and the story has tragedy...the tragedy of loss of life (besides the two former students...the lives of the aborted babies, the life of the student's mom), the tragedy of dogmatic authority over love of people, the tragedy of organized religion replacing true faith in God. This is what is wrong with today's Islam, and yesterday's Judaism and Christianity.\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"negative\",\n          \"positive\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

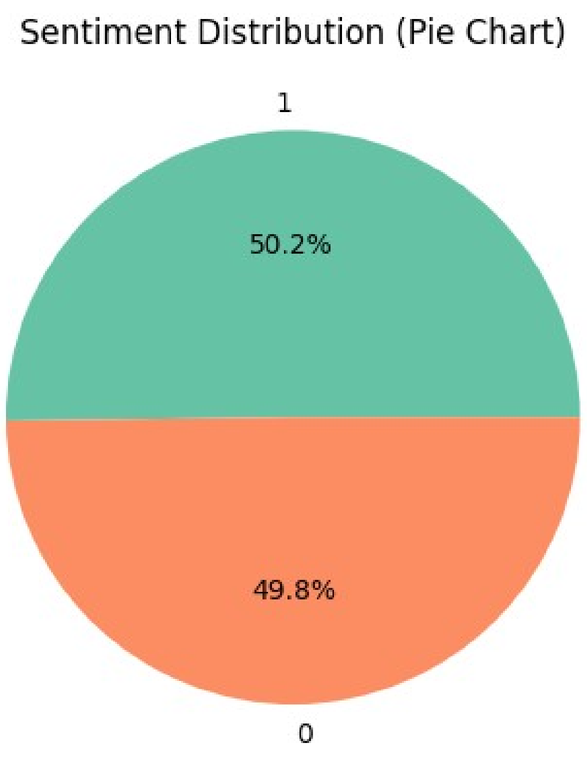
Step 3: Exploratory Data Analysis (EDA)

- Analyze the dataset to check for missing values and visualize the distribution of sentiments.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB

df.duplicated().sum()

418

df.drop_duplicates(inplace=True)

df.isnull().sum()

review       0
sentiment    0
dtype: int64

plt.figure(figsize=(10,6))
sns.countplot(x='sentiment', data=df, palette='Set2')
plt.title('Distribution of Sentiments')
plt.xlabel('Sentiment')
plt.ylabel('Count');
```



```
# Pie chart for sentiment distribution
sentiment_counts = df['sentiment'].value_counts()
sentiment_counts.plot(kind='pie', autopct='%1.1f%%',
colors=sns.color_palette('Set2', len(sentiment_counts)))
plt.title('Sentiment Distribution (Pie Chart)')
plt.ylabel('')  # Hides the y-axis label
plt.show()
```



Step 4: Data Preprocessing

- Prepare the dataset for training by encoding the target variable and cleaning the text data.

```
# Encode Sentiments (Binary Values):

df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
print(df['sentiment'].value_counts())  # Count of positive and
negative sentiments

Series([], Name: count, dtype: int64)

# Use regular expressions and NLP techniques to clean the text.

import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
def preprocess_text(review):
  review = re.sub(r'/W', ' ', review)
  review = review.lower()
  words = review.split()
  words = [word for word in words if word not in stop_words]
  return ' '.join(words)

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.

df['review'] = df['review'].apply(preprocess_text)
```

Step 5: Split Data into Training and Testing Sets

- Divide the dataset into training (80%) and testing (20%) sets.

```
X = df['review']
y = df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Step 6: Text Vectorization (TF-IDF)

- Convert the text reviews into numerical vectors using TF-IDF Vectorizer.

```
tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train).toarray()
X_test_tfidf = tfidf.transform(X_test).toarray()

print(X_train_tfidf.shape, X_test_tfidf.shape)

(39665, 5000) (9917, 5000)
```

Step 7: Train the Naive Bayes Model

- Train a Multinomial Naive Bayes classifier on the vectorized data.

```
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)

# Predictions
y_pred = model.predict(X_test_tfidf)
```

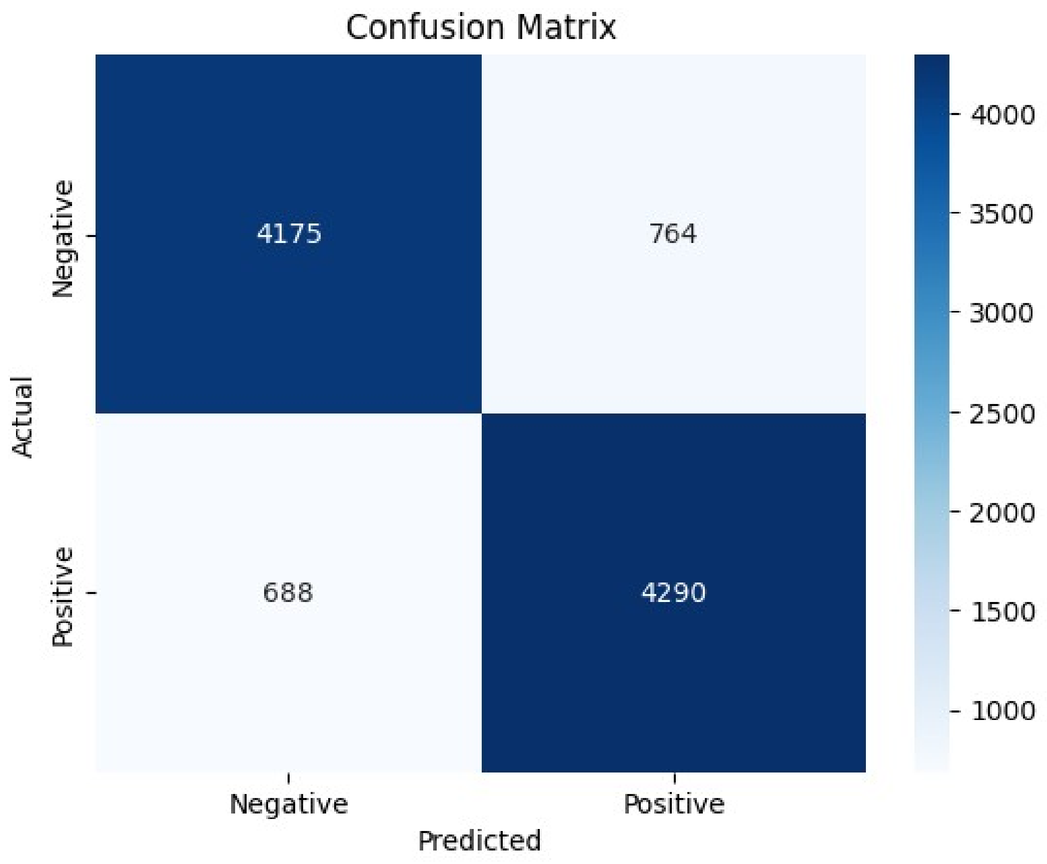
Step 8: Model Evaluation

- Evaluate the model using accuracy, classification report, and confusion matrix.

```
# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification report
print(classification_report(y_test, y_pred))
```

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

           0       0.86      0.85      0.85      4939
           1       0.85      0.86      0.86      4978

    accuracy                           0.85      9917
   macro avg       0.85      0.85      0.85      9917
weighted avg       0.85      0.85      0.85      9917
```

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Negative', 'Positive'], yticklabels=['Negative',
'Positive'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Step 9: Visualize Important Words

- Identify words that strongly contribute to positive and negative sentiments.

```
feature_names = tfidf.get_feature_names_out()
# Access feature_log_prob_ instead of coef_
sorted_indices = model.feature_log_prob_[0].argsort()

top_positive = [(feature_names[i], model.feature_log_prob_[0][i]) for
i in sorted_indices[-10:]]
top_negative = [(feature_names[i], model.feature_log_prob_[0][i]) for
i in sorted_indices[:10]]

print("Top Positive Words:", top_positive)
print("Top Negative Words:", top_negative)

Top Positive Words: [('would', -5.869157353971577), ('it', -
5.845140431949791), ('good', -5.8373286875111985), ('even', -
5.794457221628031), ('like', -5.59673558344928), ('bad', -
5.576128437429326), ('one', -5.544297013572176), ('film', -
5.176496580580668), ('movie', -4.8470593484104185), ('br', -
4.273098655878022)]
Top Negative Words: [('custer', -11.663612456136724), ('felix', -
11.246701063895532), ('miyazaki', -11.097637915304782), ('matthau', -
10.978671290505133), ('superbly', -10.913075122498736), ('flawless', -
10.814344788077275), ('understated', -10.812514504499433),
('loneliness', -10.778702590134257), ('uplifting', -
10.705577096567543), ('cassavetes', -10.694063522954673)]
```

Step 10: Save the Model and Vectorizer

- Save the model and vectorizer for future use.

```python
import pickle

# Save vectorizer
with open('tfidf_vectorizer.pkl', 'wb') as f:
    pickle.dump(tfidf, f)

# Save the model
with open('sentiment_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

Model Summary:

The model achieved an **accuracy** of **85%**, indicating that it correctly predicted the sentiment of 85% of the movie reviews in the test set.

Here’s a breakdown of the model's performance based on the classification report:

Precision:

- **Precision** for class **0** (Negative sentiment) is **0.86**, meaning that when the model predicts a review as negative, it is correct 86% of the time.
- **Precision** for class **1** (Positive sentiment) is **0.85**, meaning that when the model predicts a review as positive, it is correct 85% of the time.

Overall Precision: The average precision across both classes is **0.85** (macro average), indicating a balanced performance for both positive and negative sentiments.

Recall:

- **Recall** for class **0** (Negative sentiment) is **0.85**, meaning that the model correctly identifies 85% of all actual negative reviews.
- **Recall** for class **1** (Positive sentiment) is **0.86**, meaning the model correctly identifies 86% of all actual positive reviews.

Overall Recall: The recall for both classes is quite close, showing that the model performs similarly well in identifying both types of sentiment.

F1-Score:

- The **F1-score** for class **0** (Negative sentiment) is **0.85**, and for class **1** (Positive sentiment), it is **0.86**.
- The **F1-score** is the harmonic mean of precision and recall. An F1-score of **0.85** for negative and **0.86** for positive suggests that the model has a good balance between precision and recall for both classes.

Support:

- **Support** represents the number of actual occurrences of each class in the dataset:
  - Class **0** (Negative) has **4939** samples.
  - Class **1** (Positive) has **4978** samples.

The support values show a relatively balanced dataset in terms of the number of positive and negative reviews.

Macro Average and Weighted Average:

- **Macro Average**: The average of precision, recall, and F1-score calculated independently for each class (without considering the class distribution). In this case, it is **0.85**, indicating that the model performs consistently across both classes.

- **Weighted Average**: This takes the support (the number of samples in each class) into account, giving more weight to the class with more samples. Here, the weighted average of **0.85** shows that the model performs equally well for both positive and negative classes.

Conclusion:

The **Naive Bayes Sentiment Analysis model** has a strong performance with an accuracy of **85%**. The precision, recall, and F1-scores for both positive and negative classes are very close, indicating that the model is well-balanced and performs fairly for both sentiments. The model is effective at understanding and predicting the sentiment of IMDB movie reviews and is a reliable tool for classifying text data into positive or negative categories.

We could improve performance further by trying different models, tuning hyperparameters, or adding more advanced techniques such as deep learning if we need better accuracy or more nuanced classifications.