DATA MINING PROJECT

January 14, 2024

1 MOVIE REVIEW PROJECT

1.1 Introduction

The purpose of this code is to perform sentiment analysis on movie reviews using machine learning techniques. Sentiment analysis, also known as opinion mining, is a computational approach that aims to determine the sentiment or emotional tone expressed in a piece of text, such as a movie review.

Background Information:

In this analysis, we work with two datasets: the IMDB dataset and a test dataset. The IMDB dataset contains a collection of movie reviews along with their associated sentiments (positive or negative). The goal is to train a machine learning model using the IMDB dataset to understand the sentiments expressed in these reviews.

Once the model is trained, it can predict the sentiment of new, unseen movie reviews. The code includes data preprocessing steps, text cleaning, feature extraction using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, and the utilization of a Random Forest Classifier for sentiment prediction.

By analyzing movie reviews in this way, we can gain insights into the sentiments of viewers, which can be valuable for various applications, including recommendation systems, market research, and understanding audience preferences in the film industry.

[84]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\zahid\anaconda3\lib\site-
packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in
c:\users\zahid\anaconda3\lib\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\zahid\anaconda3\lib\site-
packages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\zahid\anaconda3\lib\site-
packages (from wordcloud) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\zahid\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\zahid\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[85]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier  # Using RandomForest for␣
 ↪comparison
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
from wordcloud import WordCloud
from sklearn.metrics import confusion_matrix
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\zahid\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\zahid\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

1.2 Dataset Loading:

[86]:
```
# Load the datasets
imdb_data = pd.read_csv('IMDB Dataset.csv',nrows=35000)
test_data = pd.read_csv('test.csv',nrows=14000)
```

1.3 sentiment analysis

[87]:
```
imdb_data.head()
```

[87]:

	review	sentiment
0	This movie really shows its age. The print I s…	negative

```
1  The title should have been the walker. The guy…  negative
2  Sequels are a capricious lot with most nowhere…  positive
3  I really wish Hollywood would come up with som…  negative
4  I wasn't quite sure if this was just going to …  positive
```

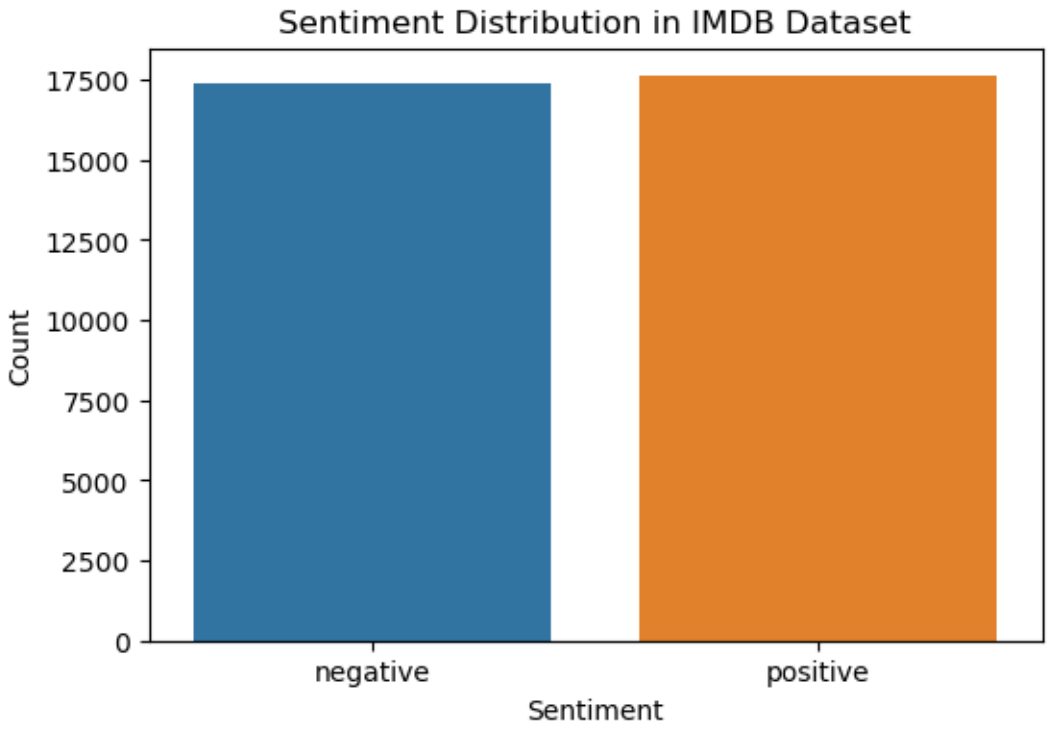
[88]:
```
imdb_data.shape
```

[88]:
```
(35000, 2)
```

[89]:
```
imdb_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35000 entries, 0 to 34999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     35000 non-null  object
 1   sentiment  35000 non-null  object
dtypes: object(2)
memory usage: 547.0+ KB
```

[90]:
```
# Plotting sentiment distribution in the IMDB dataset
plt.figure(figsize=(6, 4))
sns.countplot(x=imdb_data['sentiment'])
plt.title('Sentiment Distribution in IMDB Dataset')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```



[91]:
```
for i in range(5):
    print("Review: ", [i+1])
    print(imdb_data['review'].iloc[i], "\n")
    print("Sentiment: ", imdb_data['sentiment'].iloc[i], "\n\n")
```

```
Review:  [1]
This movie really shows its age. The print I saw was terrible due to age, but it
is possible that there are better prints out there. However, this was not the
major problem with the movie. The problem was that although the film was made in
1933, it was essentially a silent film with only the barest of dialog scattered
(only a few sentences) in the film in the most amateur fashion. Sometimes the
characters' backs were turned or they were talking with their hands over their
faces--all in a pathetic attempt to obscure their lips and "cleaverly" (?) hide
the fact that the film was dubbed. Well, its true that this Czech film would
need to be dubbed into many languages but to do it this way was really stupid
and obvious. It just looked cheap.<br /><br />Overall, the film looked low
budget and silly. It's really a shame though, because there was a grain of a
good story--a young woman who marries an older man who is either gay and/or has
no interest in women. But in the 21st century, few people would really be
willing to sit through this archaic mess. EVEN with a few glimpses of the naked
(and somewhat chunky) Hedy Lamarr, it isn't worth all the fuss that accompanied
the film when it debuted. Even by 1933 standards, this film was a poorly made
dud. About the only interesting thing about the film is to see how different
Lamarr looked in 1933 compared with the glamorous image Hollywood created when
```

she came to America--she looks like 2 completely different people.

It's such an incomplete looking and technically inferior film, I don't see how it has gotten such rave reviews. For technical problems alone, the movie can't rate a 10 or anything near it.

Sentiment: negative

Review: [2]
The title should have been the walker. The guy expend 90% of the movie walking. He doesn't know what he wants, or what he is. Go through life stealing peoples identity for nothing. He gets no benefit, no money, nothing pretending to be another person.

No body was able to understand why he was pretending to be somebody else.

The only thing that was clear in this movie is that he love his father and was a good son. But the rest was crap.

May be director is a looser that would like to be somebody else. But what he really should do is to get a real job, because after his movie, I don't think he has a chance to make as a movie producer.

Sentiment: negative

Review: [3]
Sequels are a capricious lot with most nowhere near the stature of the original. Sometimes you find a sequel that is considered better than the original, some critics (such as John Charles) have stated that Project A2 is better than the original, I disagree somewhat but this movie is still a worthwhile follow-up and fits well in the output of brilliant Hong Kong action cinema in the 1980s as well as Jackie's own oeuvre. I do wonder how with such an awesome release of great films that his later films were not as good. He only has directed two films in the 1990s and none past that, but he has had much clout in many of the films where he is not officially the director.

Earlier in 1987 Jackie had brain surgery following a disastrous fall in the filming of Armour of God. This encouraged him to work on his next film close to home. This did not encourage him to stop risking his life and his stunt team for our amusement. What resulted is a smash hit at home that eclipsed the original in box office tallies (31 million HK dollars compared to 19 million for the original).

Jackie Chan is once again police officer extraordinaire Dragon Ma and he is ordered to work with "Three Wan" Superintendent Chun (Lam Wai, Royal Warriors) who is the only Chinese police officer allowed to have a gun yet is thought to be staging arrests to make himself look better and ignoring the crimes of a triad lord named Tiger Au (Michael Chan Wai-Man, Dragon Lord). Apparently Chun has too much power to be taken down directly, but he is relieved of the Sai Wan district (now he is "Two Wan") which Dragon Ma takes over. This inefficient and corrupt office will soon get a makeover and there is a great scene where three officers, who do not know who they are dealing with, attempt to assault Ma to teach him a lesson about complaining about police officers. He soon has that district ship-shape and Tiger Au taken care of. The fight

choreography and stunts with Tiger and his men are quite awesome. My favorite stunt was a beautifully brutal fall from the second floor into a large vase and that vase did not appear to be soft.

Meanwhile a couple of subplots are happening. There are pirates who have survived from the first film who are looking for revenge and food. Then there are revolutionaries including Maggie (Maggie Cheung, In The Mood For Love) and (Rosamund Kwan, Casino Raiders) who are trying to raise funds for Dr. Sun Yat-sen to overthrow the Qing Government as well as government operatives who are trying to find these rebels. Throw in a mixture of corrupt Hong Kong and British Cops as well as legitimate ones and you have a stew that is getting a bit too many ingredients, but yet still seems to coalesce. This works well when there is a Marx Brothers influenced scene (the Marx Brothers have done this type of scene a few times with The Cocoanuts (1929) being the first) at Maggie's place where everyone is looking for someone while hiding from someone else. Many weeks were spent on this scene alone and the effort certainly shows.

There are several faults with the film. There is a certain didactic nature that creeps in the film that seems a bit out-of-place especially one small speech towards the end that Jackie gives when dealing with the Mainland revolutionaries and the extremely easy conversion of the pirates that survived from the first film. Female characters are once again underused and under-appreciated, especially Maggie Cheung. I was not as satisfied with the continuance of the plot as much as the first film either. The individual scenes dominate my feelings for the film instead of thinking of this movie as a cohesive whole. I do not fault the film for not being able to have Sammo Hung and Yuen Biao like the first though (I have heard the main reason behind this was that those two were filming Eastern Condors, but I do wonder if Jackie could have waited a small while to get them to perform in this they would work together for the last time the following year in Dragons Forever), but they are missed.

I found this to be quite an enjoyable and well-made film and it is rightfully regarded as one of the better comedic action films of the 1980s. This film is also quite good in a few unexpected places. The art direction is superb (Eddie Ma Poon-chiu), the costumes are exquisite, the cinematography is good and the movie looks quite authentic. But the stunts, comedy and the action is what I remember this film for. There is a chase involving a handcuffed Dragon and Chun that is superb (part of the axe throwing scene would be used in Shanghai Noon). The last twenty minutes is full of awe-inspiring hits, falls, chili-peppers as a mouth-mace (Jackie writes in his autobiography about how he used real peppers in this scene; you can see him in a lot of mouth pain during the outtakes at the end) and is a worthy conclusion to this movie. The most famous stunt from this sequence is his homage to Buster Keaton from Steamboat Bill Jr. (1928) with the exception that there is no hole and only a weak section where his head pops through.

Fans of Jackie and/or Hong Kong action cinema should consider this a must own and watch. I certainly do.

Sentiment: positive

Review: [4]

I really wish Hollywood would come up with some new ideas and quick. Instead they go around and recreate and mess up a perfectly good movie with a re-make. This movie is awful from the DeMille version. All the way through this movie I was saying to myself, Huh??? - What???? - I don't remember that part. The only exciting thing in this movie so far was the parting of the Red Sea. And in Heston's version - it was a heck of a lot better than this version. Did anyone else see an atomic or nuclear bomb cloud fade in and out when the Red Sea was being parted? I think I did. Anyway, I Might - Might - watch the last part tonight.

I wish Hollywood would tackle different ideas and subject matters when they are making new movies. Instead of re-hashing old films.

They should of left well enough alone.

UPDATE:

Well I watched the last part. Did Moses make up the second copy of the ten commandments with his own hand - or was I seeing things - please someone - email me and let me know. HORRIBLE

Sentiment: negative

Review: [5]
I wasn't quite sure if this was just going to be another one of those idiotic nighttime soap operas that seem to clutter prime time but, as it turns out, this is a pretty good show (no small thanks to talented casting). Four female friends with diverse backgrounds get together and share the weekly goings-on of their love-lives. The hour long program follows each of them separately through their often screwed up quests to find love and it does it without being boring or trite. Sharon Small's "Trudi" is the homemaker one (allegedly widowed after September 11th) who gets a little preachy and annoying with her friends (who tend to be a little looser and more creative in their endeavors). It's great to see Small back on t.v., as she was great in the "Inspector Lynley Mysteries". The chick can act. Orla Brady's character (Siobhan, a lawyer) is perhaps the most damaged but still very sympathetic of the women, as she wrestles with her kind but self-absorbed husband Hari (Jaffrey, formerly of "Spooks") in his driven desire to have a child with her, regardless of her needs. The final two members of the cast are the effervescent Jess (Shellie Conn), an events planner who's a wild child who sleeps with anyone and everyone, gender not specific, and Katie, (Sarah Parrish) a somber doctor who's affair with a patient AND his son have sent her career and love life spiraling out of control. That being said, I'm hooked now and hope that the BBC continues cranking this series out because it's good, it's different and it's got a great cast.

Sentiment: positive

1.4 Data Preprocessing

1. Lowercasing
2. Removing HTML Tags

3. Removing URLs
4. Removing Special Characters
5. Stopword Removal
6. Lemmatization
7. Negation Handling

[92]:
```
# Compile the negation pattern outside the function
negation_pattern = re.compile(r'\b(not|no)\b[\s]+([a-z]+)')

# Improved function for text cleaning and lemmatization
def clean_and_lemmatize_text(text):
    # Lowercasing and removing HTML tags
    text = text.lower()
    text = re.sub('<br />', '', text)

    # Removing URLs and special characters
    text = re.sub(r"https?\S+|www\S+|http\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#', '', text)
    text = re.sub(r'[^\w\s]', '', text)

    # Improved negation handling
    text = negation_pattern.sub(r'\1_\2', text)

    # Removing stopwords
    stop_words = set(stopwords.words('english'))
    words = text.split()
    filtered_words = [word for word in words if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]
    return ' '.join(lemmatized_words)
```

[93]:
```
# Apply improved cleaning to datasets
imdb_data['cleaned_review'] = imdb_data['review'].
 ↪apply(clean_and_lemmatize_text)
test_data['cleaned_review'] = test_data['review'].
 ↪apply(clean_and_lemmatize_text)
```

1.5 WORLD CLOUD

[94]:
```
# Assuming `imdb_data` is your DataFrame with 'cleaned_review' and 'sentiment'␣
 ↪columns
positive_reviews = imdb_data[imdb_data['sentiment'] == 'positive']
negative_reviews = imdb_data[imdb_data['sentiment'] == 'negative']

# Function to generate word cloud and plot
```

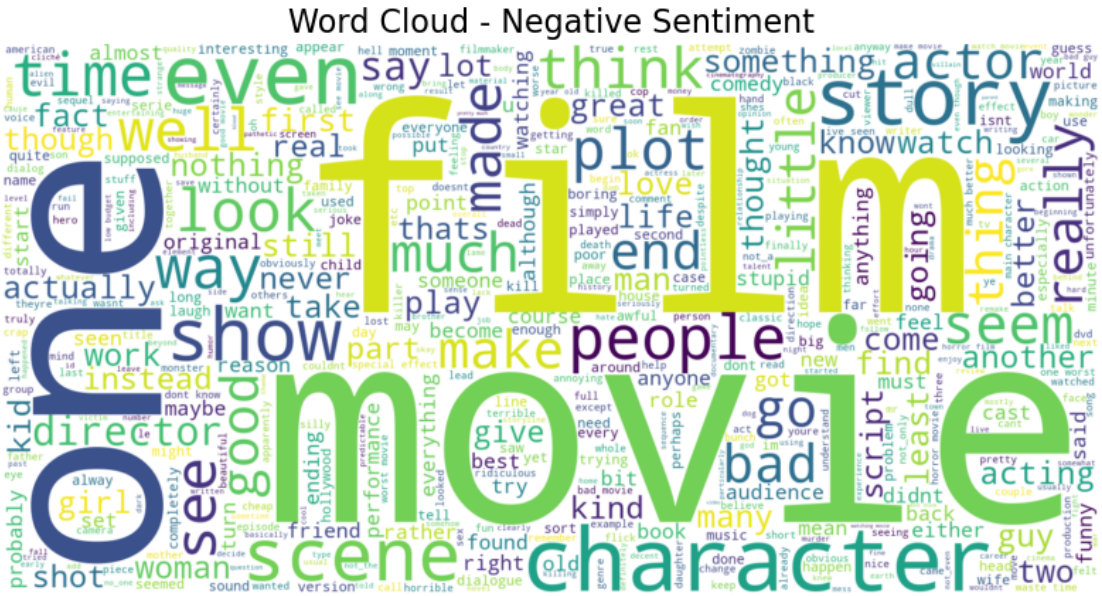
```
def generate_wordcloud(data, title):
    text = ' '.join([word for word in data['cleaned_review']])
    wordcloud = WordCloud(max_words=500, width=1600, height=800, background_color='white').generate(text)

    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, fontsize=16)
    plt.show()

# Plot positive sentiment word cloud
generate_wordcloud(positive_reviews, 'Word Cloud - Positive Sentiment')

# Plot negative sentiment word cloud
generate_wordcloud(negative_reviews, 'Word Cloud - Negative Sentiment')
```

Word Cloud - Positive Sentiment



[95]:
```
imdb_data.head()
```

[95]:
```
                                              review sentiment  \
0  This movie really shows its age. The print I s…  negative
1  The title should have been the walker. The guy…  negative
2  Sequels are a capricious lot with most nowhere…  positive
3  I really wish Hollywood would come up with som…  negative
4  I wasn't quite sure if this was just going to …  positive

                                      cleaned_review
0  movie really show age print saw terrible due a…
1  title walker guy expend 90 movie walking doesn…
2  sequel capricious lot nowhere near stature ori…
3  really wish hollywood would come new idea quic…
4  wasnt quite sure going another one idiotic nig…
```

[96]:
```
# Convert sentiments to binary
imdb_data['sentiment'] = imdb_data['sentiment'].astype('category')
test_data['sentiment'] = test_data['sentiment'].astype('category')
y_imdb = imdb_data['sentiment'].map({'positive': 1, 'negative': 0}).values
y_test = test_data['sentiment'].map({'positive': 1, 'negative': 0}).values
```

1.6 Model Building

Algorithm: Random Forest Classifier

[97]:
```
# Fit TF-IDF Vectorizer once outside the pipeline
tfidf_vectorizer = TfidfVectorizer(max_features=35000, min_df=5, max_df=0.7,␣
 ↪ngram_range=(1, 2))
X_train_tfidf = tfidf_vectorizer.fit_transform(imdb_data['cleaned_review'])
X_test_tfidf = tfidf_vectorizer.transform(test_data['cleaned_review'])

# Setting up the pipeline with RandomForestClassifier
pipeline = Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('classifier', RandomForestClassifier(n_estimators=100,random_state=42))
])
```

1.7 Training and Evaluation

1.7.1 Model Training

The sentiment analysis model is trained on the IMDb dataset using a Random Forest Classifier.

1.7.2 Evaluation Metrics

- **Accuracy Score**: Measures overall model accuracy.
- **Precision**: Proportion of true positives among positive predictions.
- **Recall**: Proportion of true positives among actual positives.
- **F1-Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Provides detailed true positives, true negatives, false positives, and false negatives.

Interpretation of Metrics: - **Accuracy**: Higher is better overall. - **Precision**: Fewer false positives. - **Recall**: Better identification of positive cases. - **F1-Score**: Balances precision and recall. - **Confusion Matrix**: Detailed error analysis.

[98]:
```
pipeline.fit(imdb_data['cleaned_review'], y_imdb)
```

[98]:
```
Pipeline(steps=[('tfidf',
                 TfidfVectorizer(max_df=0.7, max_features=35000, min_df=5,
                                 ngram_range=(1, 2))),
                ('classifier', RandomForestClassifier(random_state=42))])
```

1.8 Results:

[99]:
```
# Evaluating the model on the test set
y_pred_test = pipeline.predict(test_data['cleaned_review'])
print("Accuracy on Test Set:", accuracy_score(y_test, y_pred_test))
print("Classification Report on Test Set:\n", classification_report(y_test,␣
 ↪y_pred_test))
```

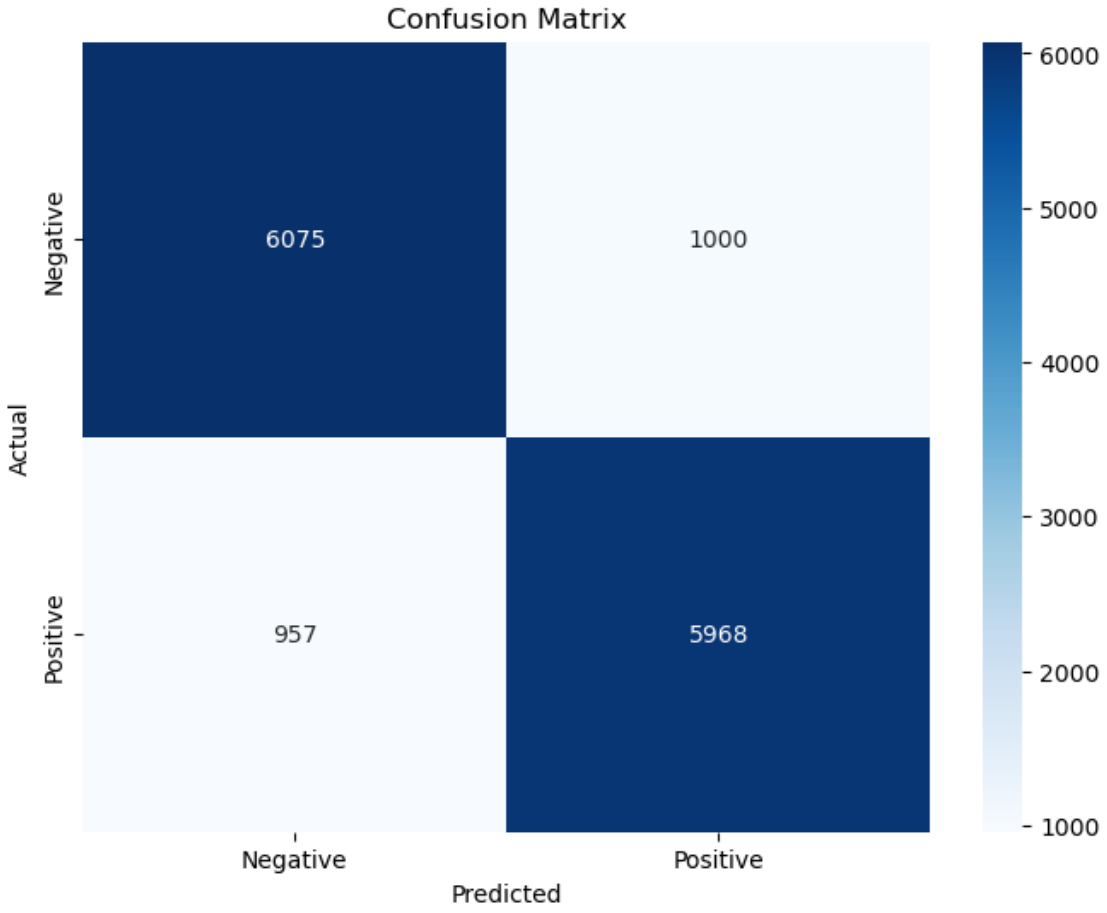
```
Accuracy on Test Set: 0.8602142857142857
Classification Report on Test Set:
              precision    recall  f1-score   support
```

```
              0       0.86      0.86      0.86      7075
              1       0.86      0.86      0.86      6925

    accuracy                            0.86     14000
   macro avg        0.86      0.86      0.86     14000
weighted avg        0.86      0.86      0.86     14000
```

[100]:
```
conf_matrix = confusion_matrix(y_test, y_pred_test)

# Display the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



1.9 FOR FUTURE PREDICTION

[101]:
```
# Function to predict new reviews
def predict_new_review(new_review, pipeline):
    cleaned_review = clean_and_lemmatize_text(new_review)
    return 'positive' if pipeline.predict([cleaned_review])[0] == 1 else␣
 ↪'negative'
```

[102]:
```
# Example usage
new_review = "i found it very heart touching movies of the year"
predicted_sentiment = predict_new_review(new_review, pipeline)
print(f"The predicted sentiment for the new review is: {predicted_sentiment}")
```

```
The predicted sentiment for the new review is: positive
```