YEAR : 2025

Project Report

Sentiment Analysis on Movie Review

Presented by:
VISHWA PARMAR
PRACHI JOSHI
KAVYA PATEL
PRACHI PRAJAPATI
PRANSHU SONI

Table of Contents

Introduction..1

Objective..3

Methodology..4

Code and Implementation Details..................................6

Results and Observations...15

Conclusions...19

INTRODUCTION

- This project focuseson developinga **Movie ReviewSentiment Analyzer**, a natural language processing (NLP) application designed to classify movie reviews as either positive or negative. The solution encompasses both the machine learning model development and a user-friendly web application for practical deployment.

- The core of the project involves building a sentiment analysis model using a dataset of movie reviews, specifically the "**IMDB Dataset.csv**", which contains 50,000 entries with an equal distribution of positive and negative sentiments. Key steps in the model development, as detailed in the **Sentiment Analysis on Movie Review.ipynb** notebook, include:

1. **Data Preprocessing:** Cleaning text data by converting it to lowercase, removing non-alphabetic characters, applying stemming using PorterStemmer, and eliminating common stopwords.
2. **Feature Extraction:** Converting text reviews into numerical features suitable for machine learning models, likely using techniques like CountVectorizer.
3. **Model Training:** Training a classification model, such as Multinomial Naive Bayes, to predict sentiment based on the processed text.
4. **Evaluation:** Assessing the model's performance using metrics like accuracy, classification reports, and confusion matrices.

INTRODUCTION

- The trained model and vectorizerarethensavedusing**pickle** for easy integration into the web application. The user-facing component of this project is a **Streamlit** web application (**app.py**). This application provides an intuitive interface for users to enter a movie review and receive an instant sentiment prediction (positive or negative). The application handles the necessary text preprocessing before feeding the review to the loaded sentiment analysis model. The interface also includes custom styling for an enhanced user experience. This project demonstrates a complete workflow for a sentiment analysis task, from data exploration and model training to deployment as an interactive web application.

OBJECTIVE

The primary objective of this project is to develop an effective and user-friendly Movie Review Sentiment Analyzer. This involves:

- **Building a robust sentiment classification model:** The goal is to accurately classify movie reviews as either positive or negative using machine learning techniques on a comprehensive dataset.
- **Implementing a full-stack solution:** To provide a practical application, the project aims to integrate the trained sentiment model into an interactive web application, allowing users to input reviews and instantly receive sentiment predictions.
- **Demonstrating proficiency in NLP and machine learning:** The project showcases the application of various NLP techniques such as text preprocessing (stemming, stopword removal) and machine learning algorithms (e.g., Multinomial Naive Bayes) for text classification.
- **Creating a well-structured and functional application:** The objective is to ensure the web application is intuitive, visually appealing, and performs its intended function reliably.

METHODOLOGY

The developmentof the MovieReview SentimentAnalyzer followed a structured methodology, encompassing data acquisition, preprocessing, model training, and application deployment.

1. Data Acquisition and Exploration:

⁎ Theprojectutilizesthe "IMDB Dataset.csv", adataset comprising50,000movie reviews, equally balanced between positive and negative sentiments.

⁎ Initial Exploratory Data Analysis (EDA) was performed to understand the dataset's structure, check for missing values, and analyze the distribution of sentiment labels.

2. Text Preprocessing:

⁎ **Noise Removal:** Regular expressions (re) were used to remove non-alphabetic characters from the movie reviews.

⁎ **Lowercasing:** All text was converted to lowercase to ensure uniformity and prevent the model from treating the same word with different cases as distinct entities.

⁎ **Tokenization and Stopword Removal:** Reviews were tokenized into individual words, and common English stopwords (e.g., "the", "a", "is") were removed using nltk.corpus.stopwords to reduce noise and focus on meaningful terms.

⁎ **Stemming:** The PorterStemmer from nltk.stem was applied to reduce words to their root form (e.g., "running" to "run") to further reduce dimensionality and improve generalization.

3. Feature Extraction:

⁎ Count Vectorization:The preprocessed text data was transformed into numerical feature vectors using CountVectorizer from sklearn.feature_extraction.text. This technique converts a collection of text documents to a matrix of token counts.

METHODOLOGY

4. Model Training:

⁕ **Data Splitting:** Thedatasetwassplit intotraining andtesting sets using

train_test_split from sklearn.model_selection to evaluate the model's performance on unseen data.

⁕ **Model Selection:** A Multinomial Naive Bayes classifier from sklearn.naivebayes was chosen for sentiment classification, a common and effective algorithm for text classification tasks.

⁕ **Training:** The selected model was trained on the preprocessed training data.

5. Model Persistence:

⁕ Both the trained MultinomialNB model and the CountVectorizer were serialized using pickle to enable their efficient loading and use in the Streamlit web application without retraining.

6. Web Application Development:

⁕ **Framework:** Streamlit was used to develop the interactive web application, providing a rapid and efficient way to create a user interface for the sentiment analyzer.

⁕ **User Interface:** The application features a text area for users to input movie reviews, an "Analyze Sentiment" button, and displays the predicted sentiment along with corresponding visual feedback (e.g., GIFs).

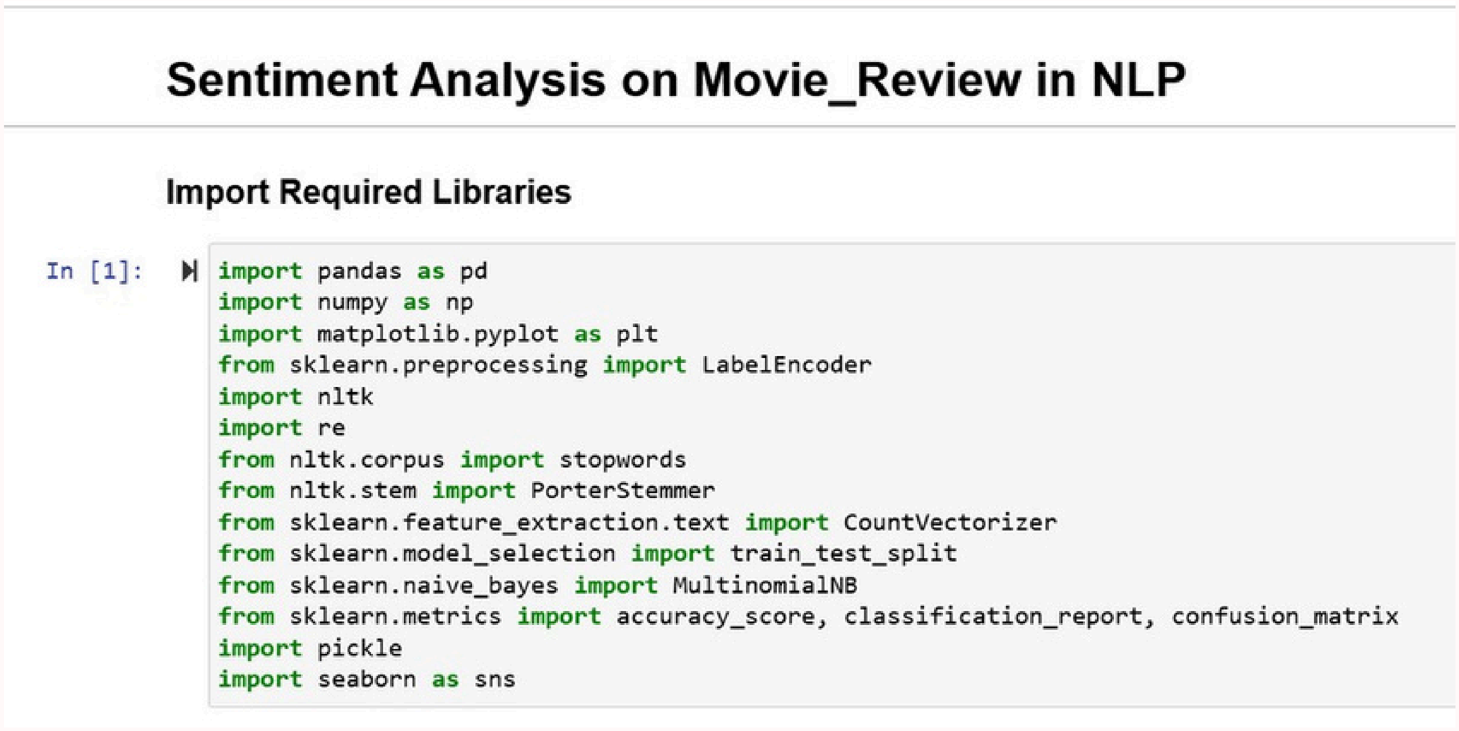
⁕ **Integration:** The app.py script loads the pre-trained model and vectorizer, preprocesses user input, and then uses the model to predict sentiment.

⁕ **Styling:** Custom CSS was incorporated to enhance the application's visual appeal, including a background image and styled elements for titles, buttons, and input areas.

CODE AND IMPLEMENTATION DETAILS

Theproject'simplementation is primarilycontainedwithin twokeyfiles:

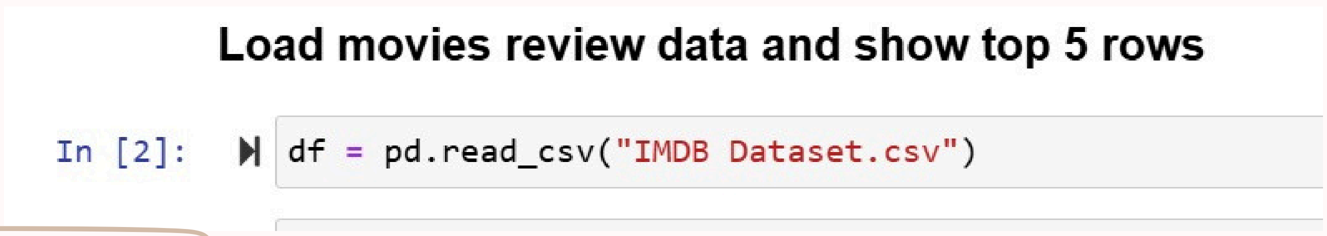
app.py for the Streamlit web application and **Sentiment Analysis on Movie Review.ipynb** for the machine learning model development and analysis. 1. **Sentiment Analysis on Movie Review.ipynb** (Jupyter Notebook for Model Development) This notebook contains the detailed steps for data loading, preprocessing, model training, and evaluation. Key sections and their functionalities include:



Sentiment Analysis on Movie_Review in NLP

Import Required Libraries

```
In [1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pickle
import seaborn as sns
```



Load movies review data and show top 5 rows

```
In [2]:
df = pd.read_csv("IMDB Dataset.csv")
```

```
In [3]:  df.head()
```

Out[3]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
In [4]:  df.tail()
```

Out[4]:

	review	sentiment
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

Perform EDA

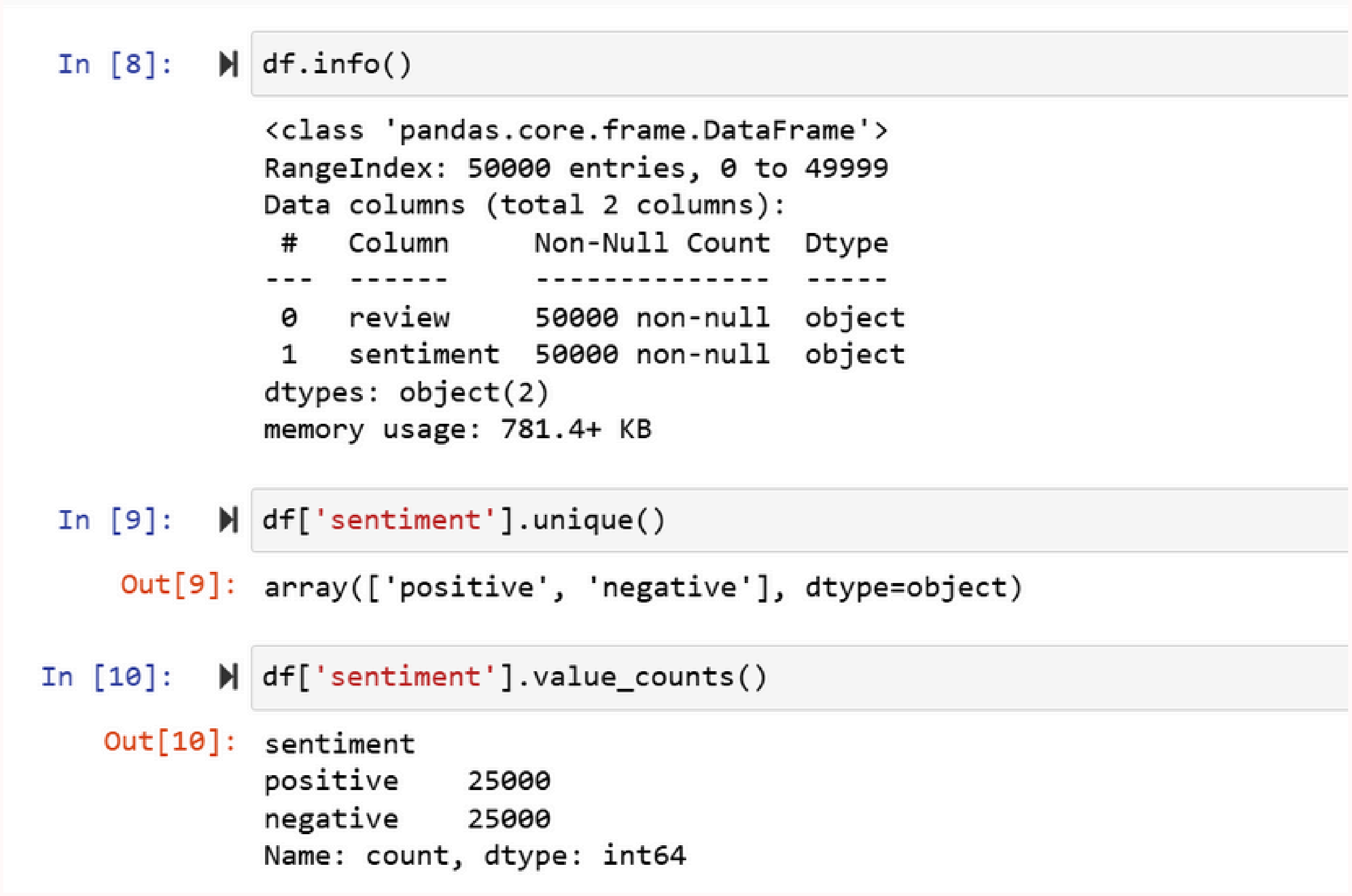
```
In [5]:  df.shape

Out[5]:  (50000, 2)
```

```
In [6]:  df.isnull().sum()

Out[6]:  review       0
         sentiment    0
         dtype: int64
```

```
In [7]:  df.describe()
```

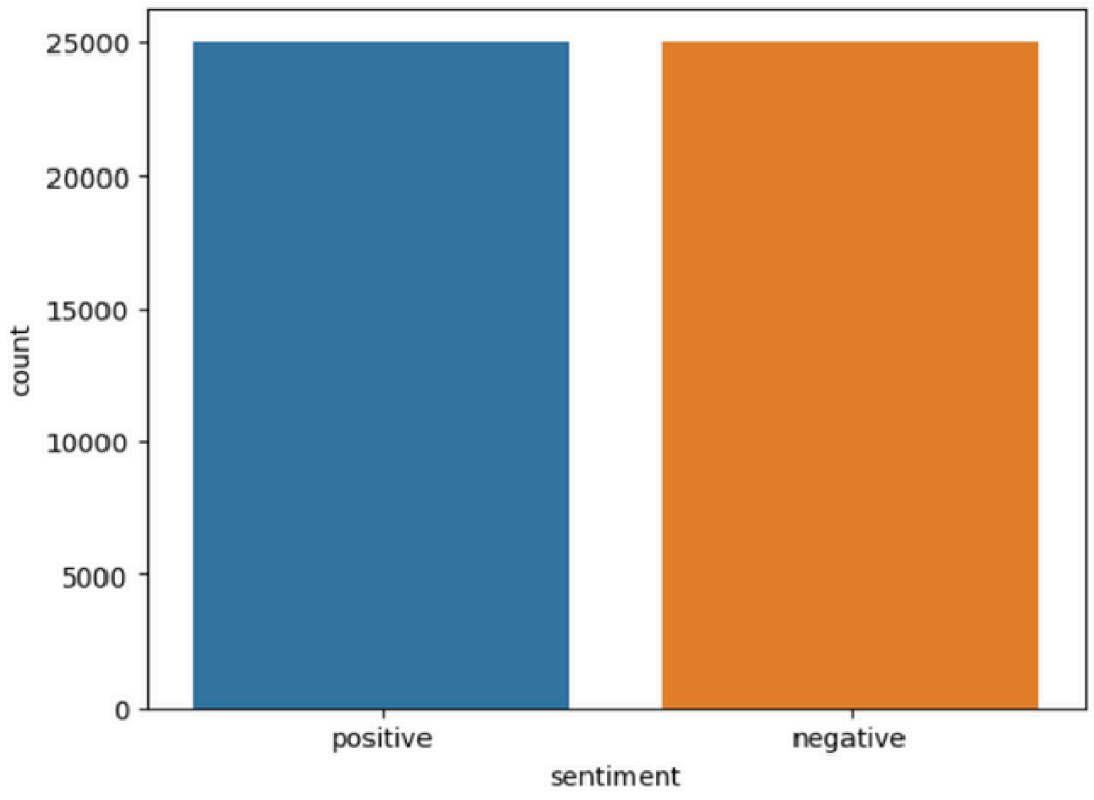
Out[7]:

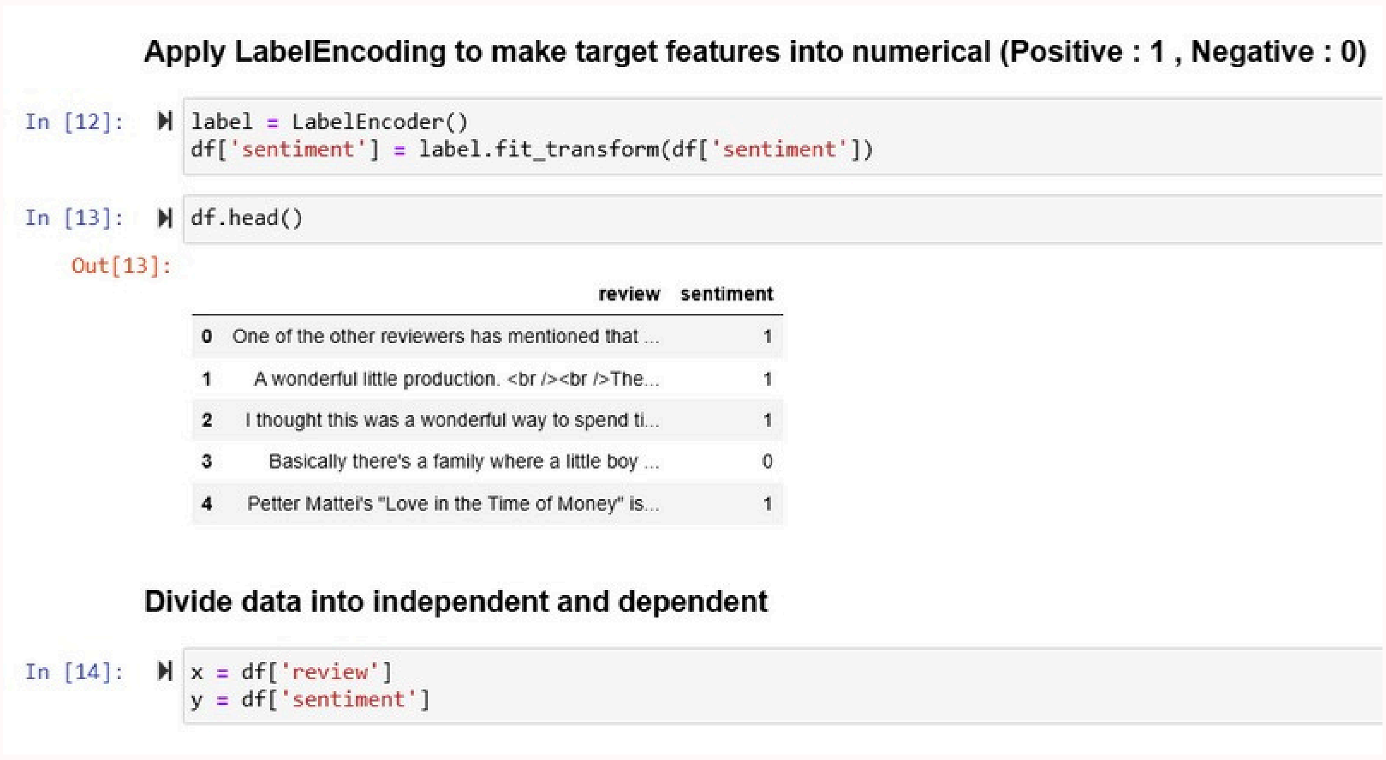
	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000



```
In [8]:   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

In [9]:   df['sentiment'].unique()

Out[9]:   array(['positive', 'negative'], dtype=object)

In [10]:  df['sentiment'].value_counts()

Out[10]:  sentiment
          positive    25000
          negative    25000
          Name: count, dtype: int64
```

```
In [11]:  sns.countplot(x='sentiment', data=df)

Out[11]:  <Axes: xlabel='sentiment', ylabel='count'>
```





Apply LabelEncoding to make target features into numerical (Positive : 1 , Negative : 0)

```
In [12]:  label = LabelEncoder()
          df['sentiment'] = label.fit_transform(df['sentiment'])
```

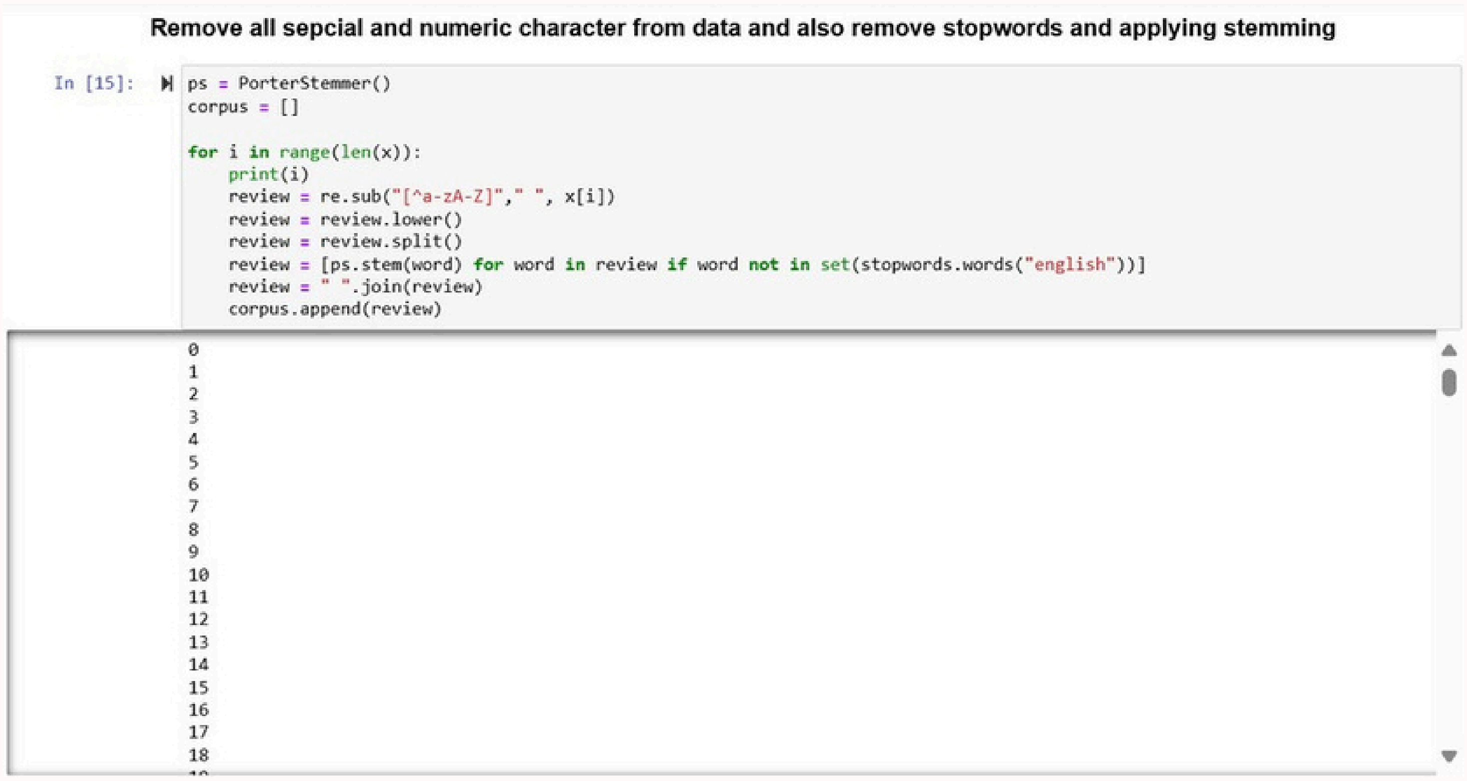
```
In [13]:  df.head()
```

Out[13]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. <br /><br />The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

Divide data into independent and dependent

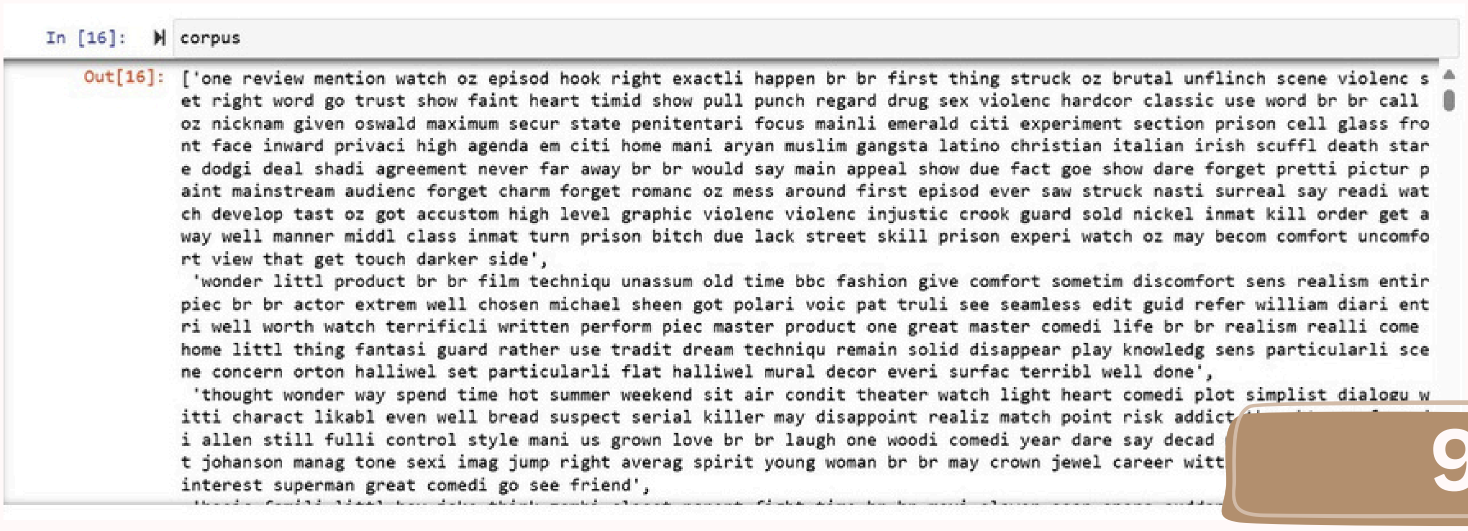
```
In [14]:  x = df['review']
          y = df['sentiment']
```



Remove all sepcial and numeric character from data and also remove stopwords and applying stemming

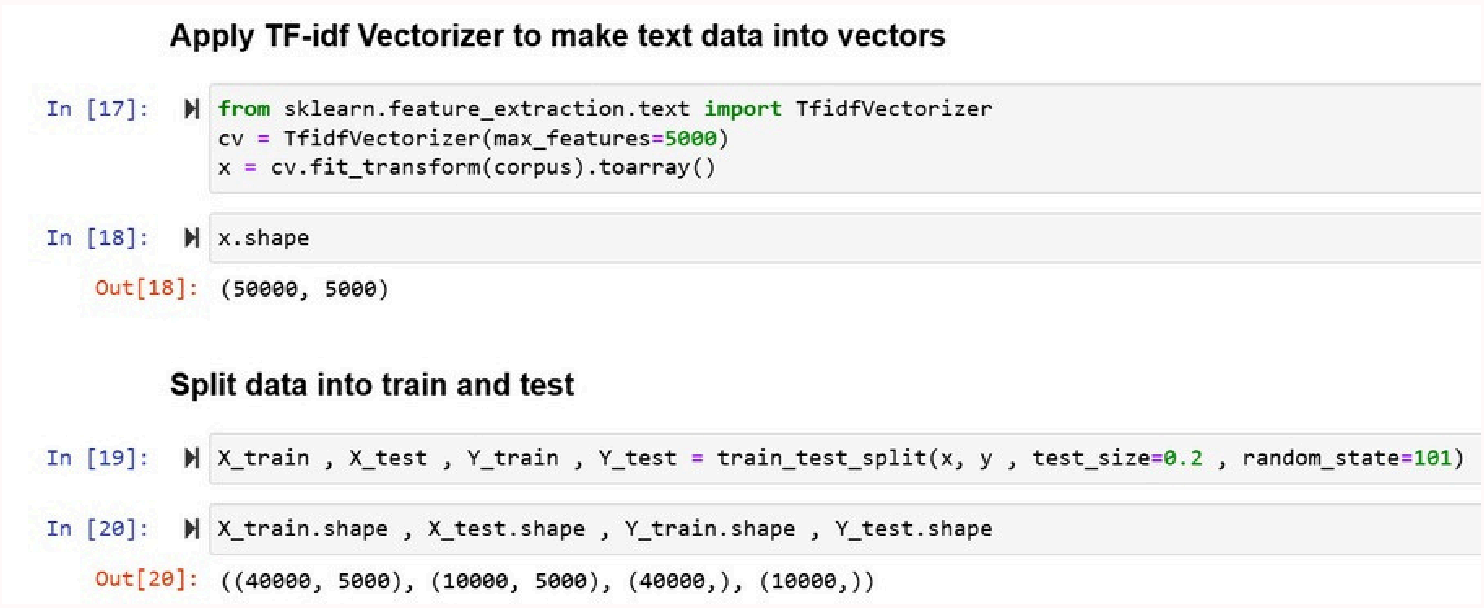
```
In [15]:  ps = PorterStemmer()
          corpus = []

          for i in range(len(x)):
              print(i)
              review = re.sub("[^a-zA-Z]"," ", x[i])
              review = review.lower()
              review = review.split()
              review = [ps.stem(word) for word in review if word not in set(stopwords.words("english"))]
              review = " ".join(review)
              corpus.append(review)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```



```
In [16]:  corpus
```

```
Out[16]: ['one review mention watch oz episod hook right exactli happen br br first thing struck oz brutal unflinch scene violenc s
         et right word go trust show faint heart timid show pull punch regard drug sex violenc hardcor classic use word br br call
         oz nicknam given oswald maximum secur state penitentari focus mainli emerald citi experiment section prison cell glass fro
         nt face inward privaci high agenda em citi home mani aryan muslim gangsta latino christian italian irish scuffl death star
         e dodgi deal shadi agreement never far away br br would say main appeal show due fact goe show dare forget pretti pictur p
         aint mainstream audienc forget charm forget romanc oz mess around first episod ever saw struck nasti surreal say readi wat
         ch develop tast oz got accustom high level graphic violenc violenc injustic crook guard sold nickel inmat kill order get a
         way well manner middl class inmat turn prison bitch due lack street skill prison experi watch oz may becom comfort uncomfo
         rt view that get touch darker side',
          'wonder littl product br br film techniqu unassum old time bbc fashion give comfort sometim discomfort sens realism entir
         piec br br actor extrem well chosen michael sheen got polari voic pat truli see seamless edit guid refer william diari ent
         ri well worth watch terrificli written perform piec master product one great master comedi life br br realism realli come
         home littl thing fantasi guard rather use tradit dream techniqu remain solid disappear play knowledg sens particularli sce
         ne concern orton halliwel set particularli flat halliwel mural decor everi surfac terribl well done',
          'thought wonder way spend time hot summer weekend sit air condit theater watch light heart comedi plot simplist dialogu w
         itti charact likabl even well bread suspect serial killer may disappoint realiz match point risk addict
         i allen still fulli control style mani us grown love br br laugh one woodi comedi year dare say decad
         t johanson manag tone sexi imag jump right averag spirit young woman br br may crown jewel career witt
         interest superman great comedi go see friend',
```



Apply TF-idf Vectorizer to make text data into vectors

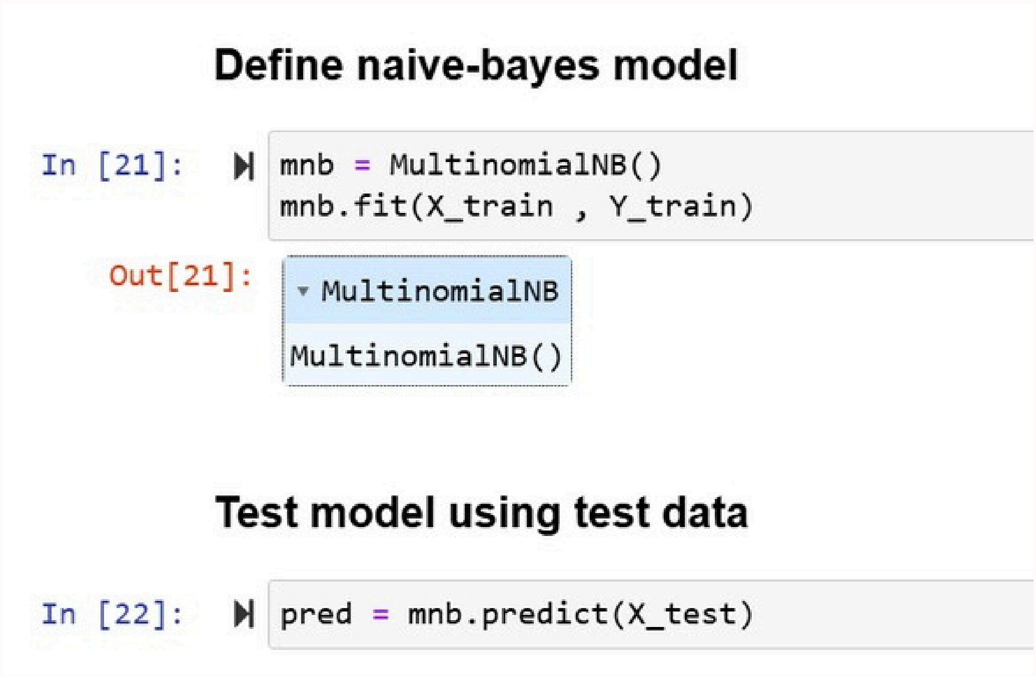
```
In [17]:  from sklearn.feature_extraction.text import TfidfVectorizer
          cv = TfidfVectorizer(max_features=5000)
          x = cv.fit_transform(corpus).toarray()
```

```
In [18]:  x.shape

Out[18]:  (50000, 5000)
```

Split data into train and test

```
In [19]:  X_train , X_test , Y_train , Y_test = train_test_split(x, y , test_size=0.2 , random_state=101)
```

```
In [20]:  X_train.shape , X_test.shape , Y_train.shape , Y_test.shape

Out[20]:  ((40000, 5000), (10000, 5000), (40000,), (10000,))
```

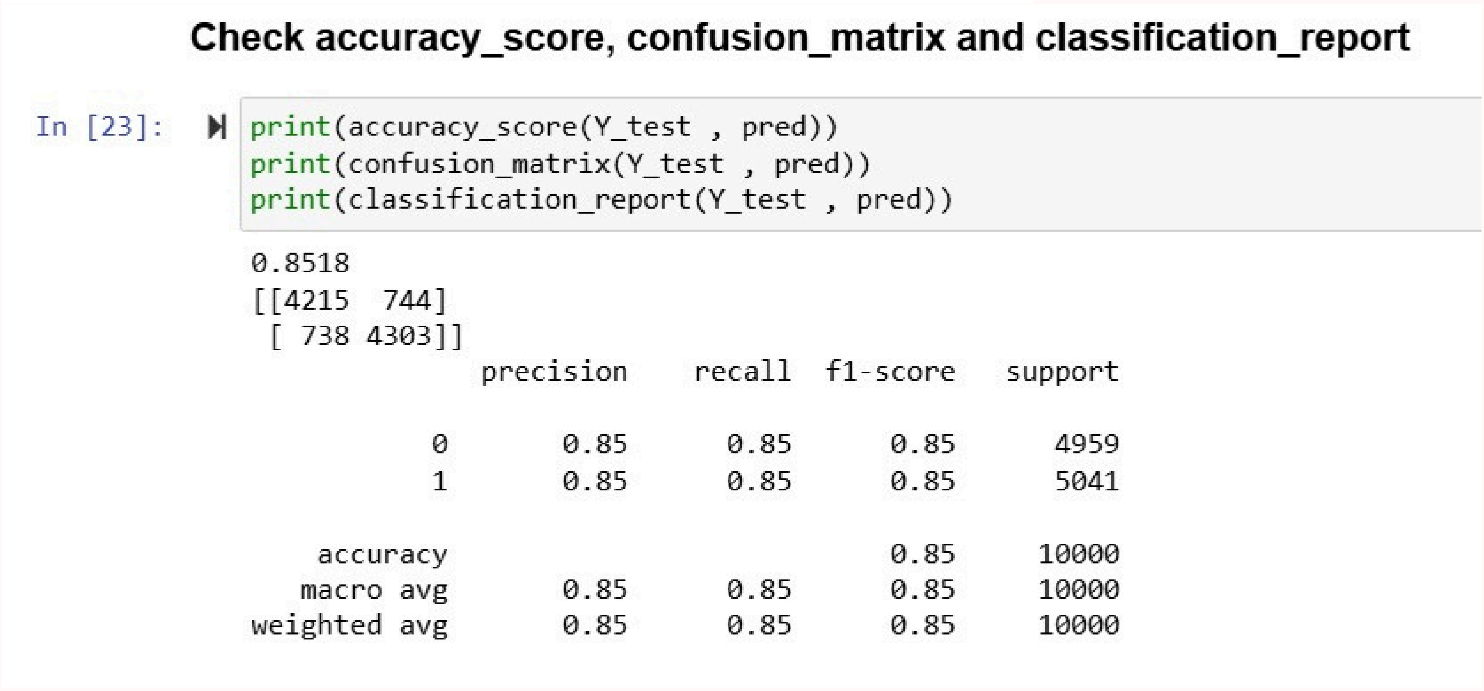


Define naive-bayes model

```
In [21]:  mnb = MultinomialNB()
          mnb.fit(X_train , Y_train)

Out[21]:  ▾ MultinomialNB
          MultinomialNB()
```

Test model using test data

```
In [22]:  pred = mnb.predict(X_test)
```



Check accuracy_score, confusion_matrix and classification_report

```
In [23]:  print(accuracy_score(Y_test , pred))
          print(confusion_matrix(Y_test , pred))
          print(classification_report(Y_test , pred))

          0.8518
          [[4215  744]
           [ 738 4303]]
                        precision    recall  f1-score   support

                     0       0.85      0.85      0.85      4959
                     1       0.85      0.85      0.85      5041

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

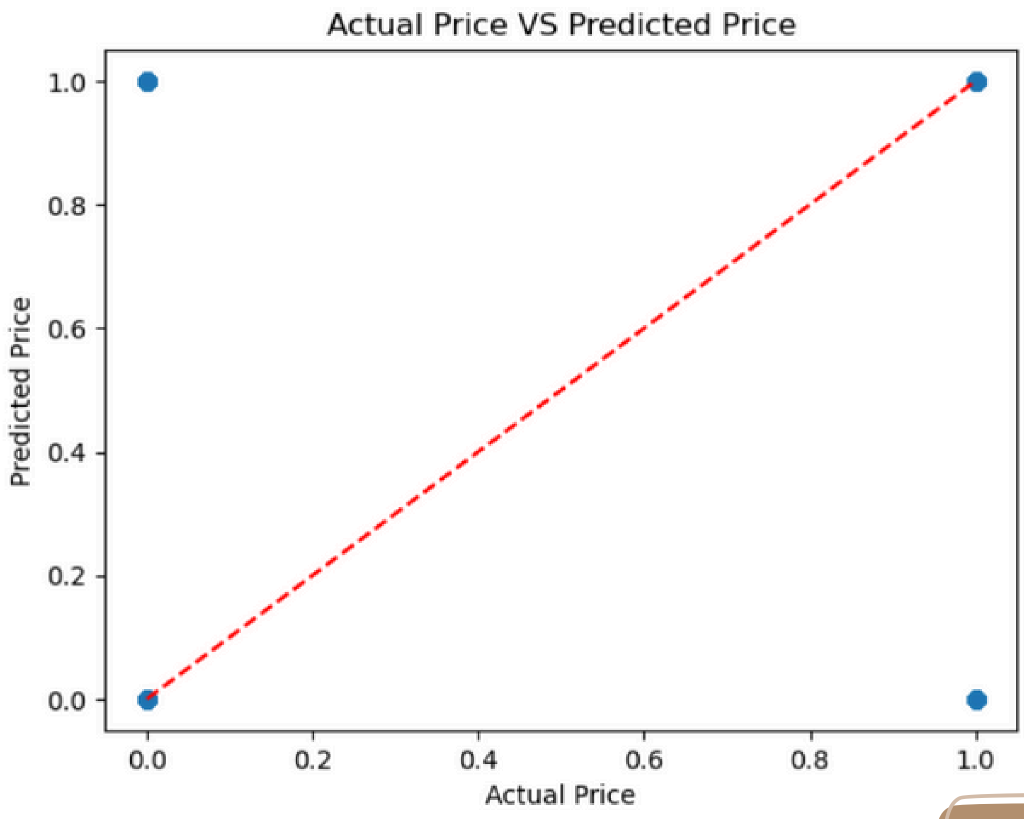
Difference Between Actual and Predict Data

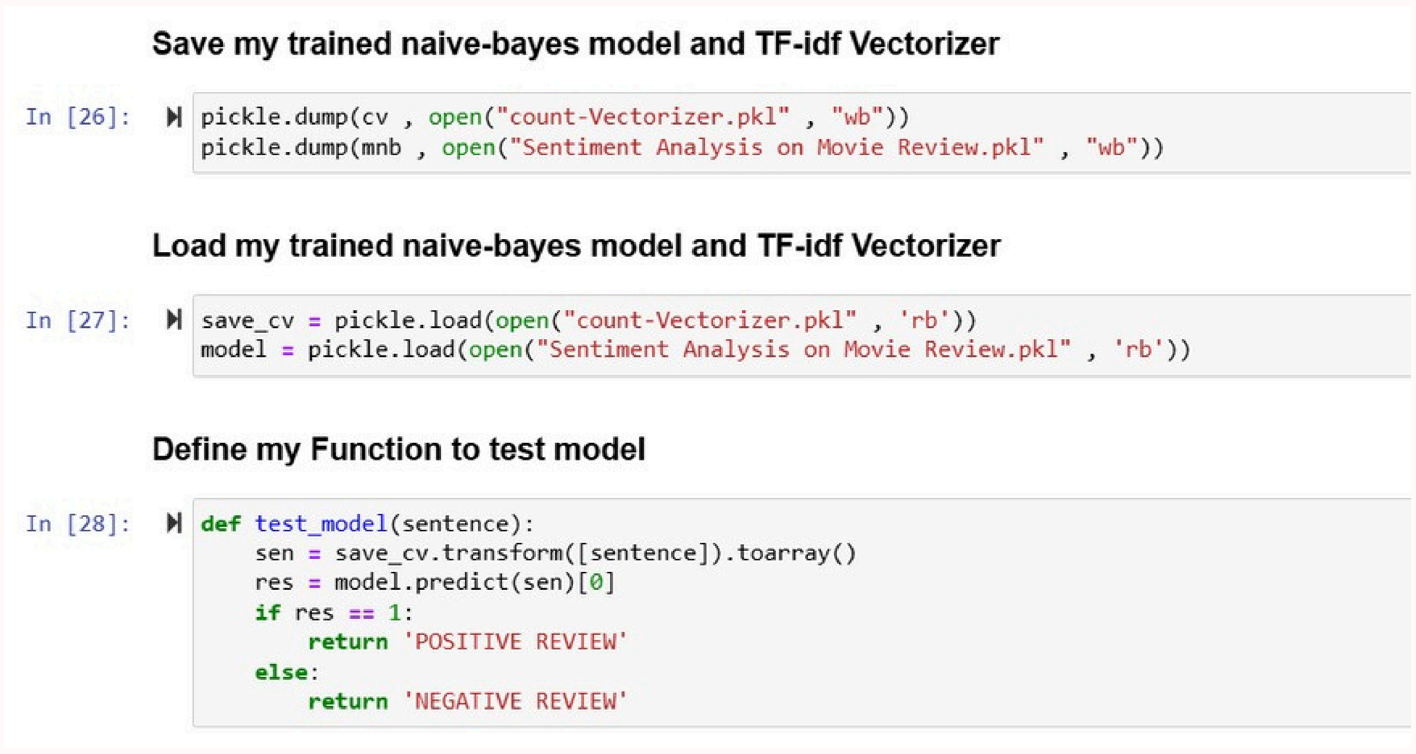
```
In [24]:  pd.DataFrame(np.c_[Y_test , pred] , columns=["Actual" , "Predicted"])
Out[24]:
```

	Actual	Predicted
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1
...	...	...
9995	1	1
9996	0	0
9997	0	0
9998	1	1
9999	1	1

10000 rows × 2 columns

```
In [25]:  plt.scatter(Y_test , pred)
          plt.plot([Y_test.min() , Y_test.max()] , [Y_test.min() , Y_test.max()] , 'r--')
          plt.xlabel('Actual Price')
          plt.ylabel('Predicted Price')
          plt.title('Actual Price VS Predicted Price')
          plt.show()
```





Save my trained naive-bayes model and TF-idf Vectorizer

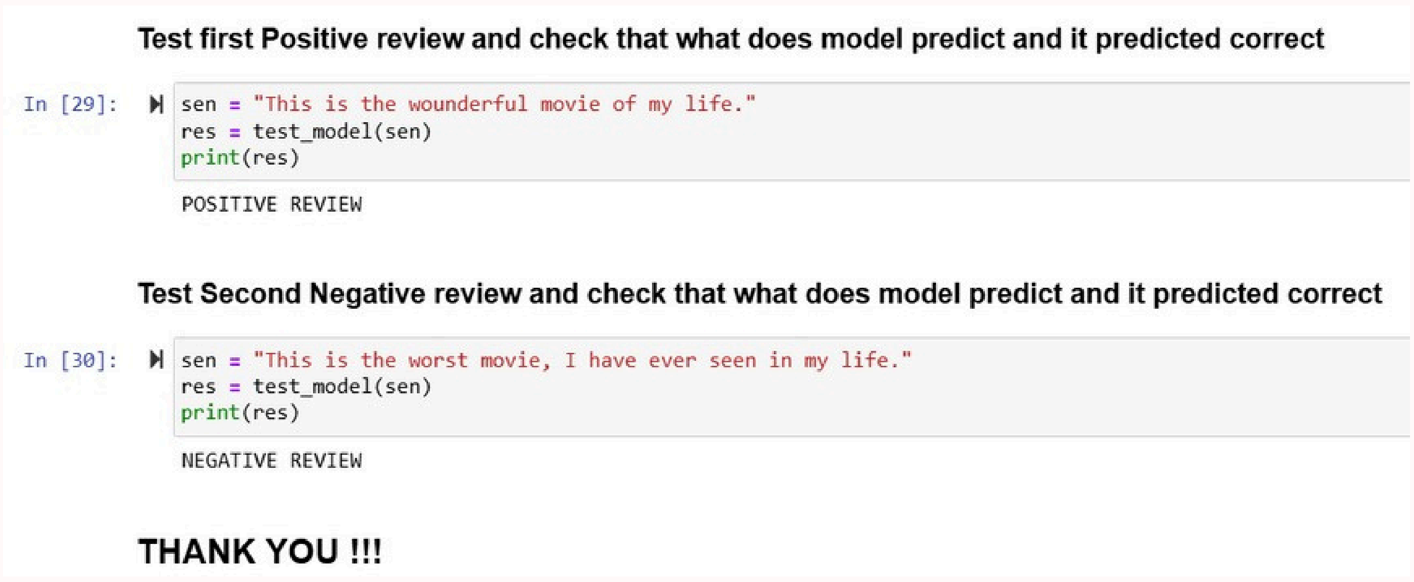
In [26]:
```python
pickle.dump(cv , open("count-Vectorizer.pkl" , "wb"))
pickle.dump(mnb , open("Sentiment Analysis on Movie Review.pkl" , "wb"))
```

Load my trained naive-bayes model and TF-idf Vectorizer

In [27]:
```python
save_cv = pickle.load(open("count-Vectorizer.pkl" , 'rb'))
model = pickle.load(open("Sentiment Analysis on Movie Review.pkl" , 'rb'))
```

Define my Function to test model

In [28]:
```python
def test_model(sentence):
    sen = save_cv.transform([sentence]).toarray()
    res = model.predict(sen)[0]
    if res == 1:
        return 'POSITIVE REVIEW'
    else:
        return 'NEGATIVE REVIEW'
```



Test first Positive review and check that what does model predict and it predicted correct

In [29]:
```python
sen = "This is the wounderful movie of my life."
res = test_model(sen)
print(res)
```

```
POSITIVE REVIEW
```

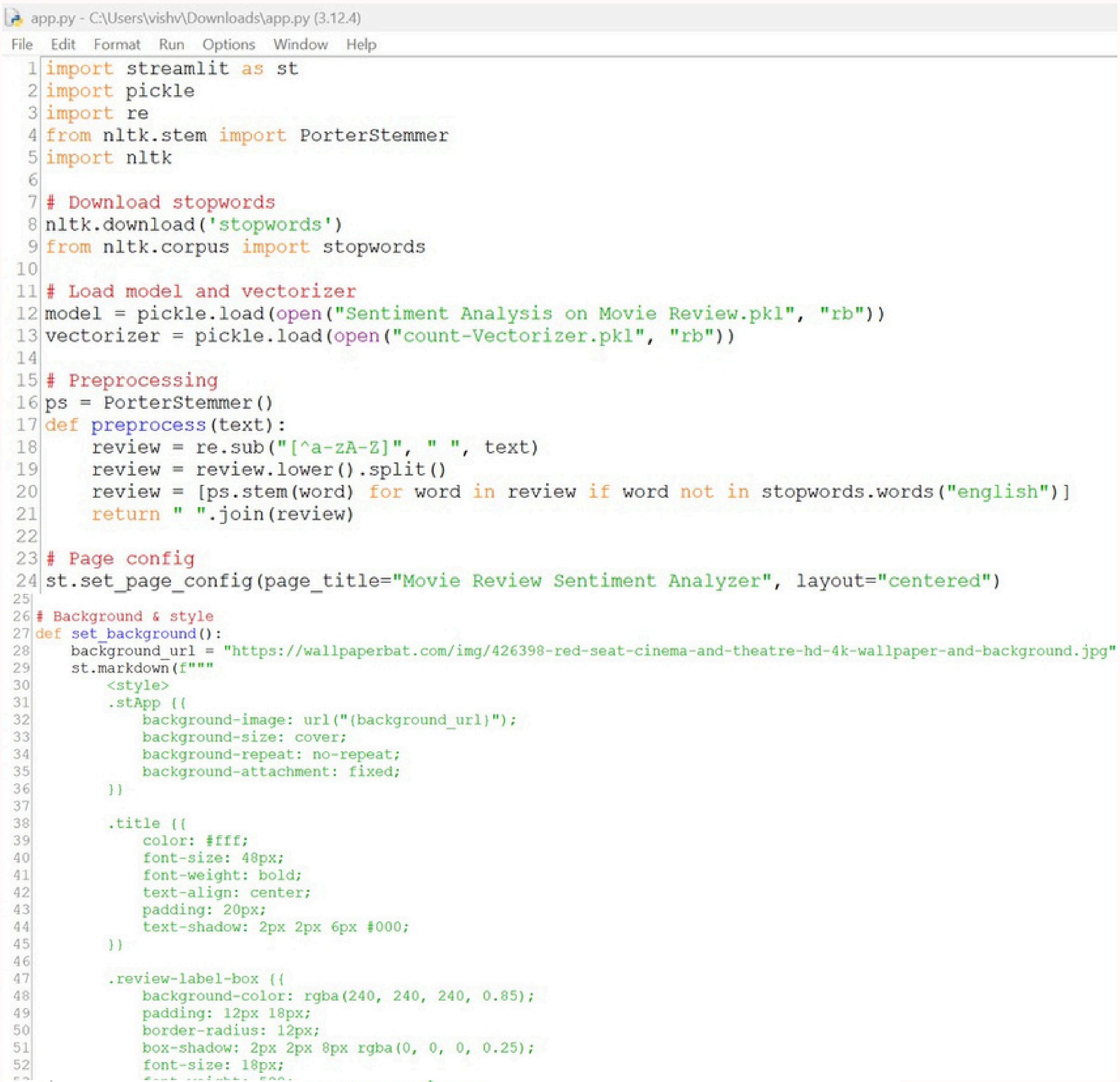
Test Second Negative review and check that what does model predict and it predicted correct

In [30]:
```python
sen = "This is the worst movie, I have ever seen in my life."
res = test_model(sen)
print(res)
```

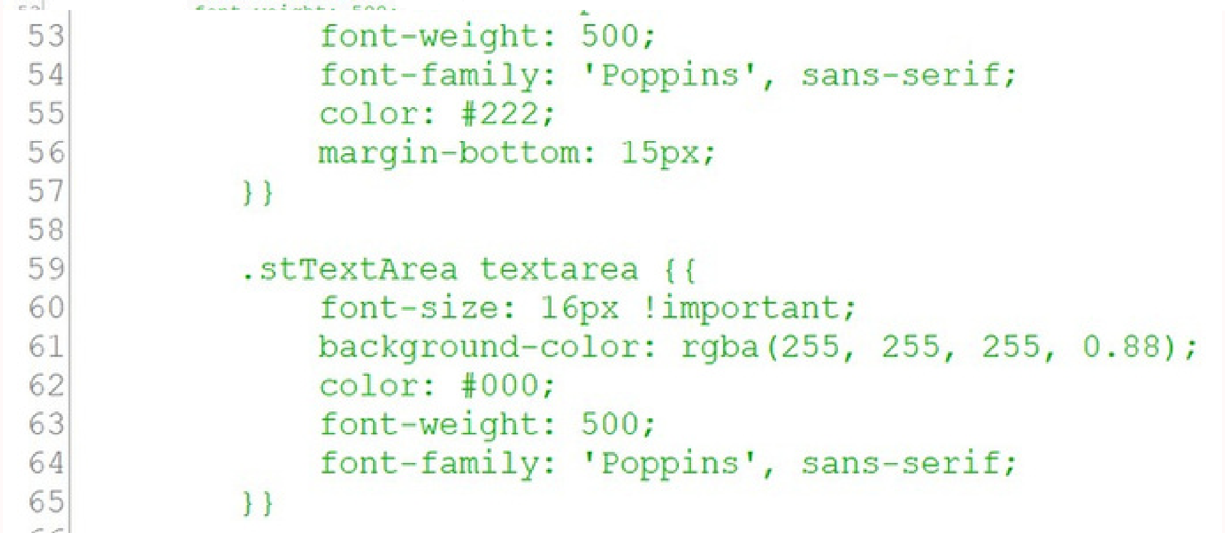
```
NEGATIVE REVIEW
```

THANK YOU !!!

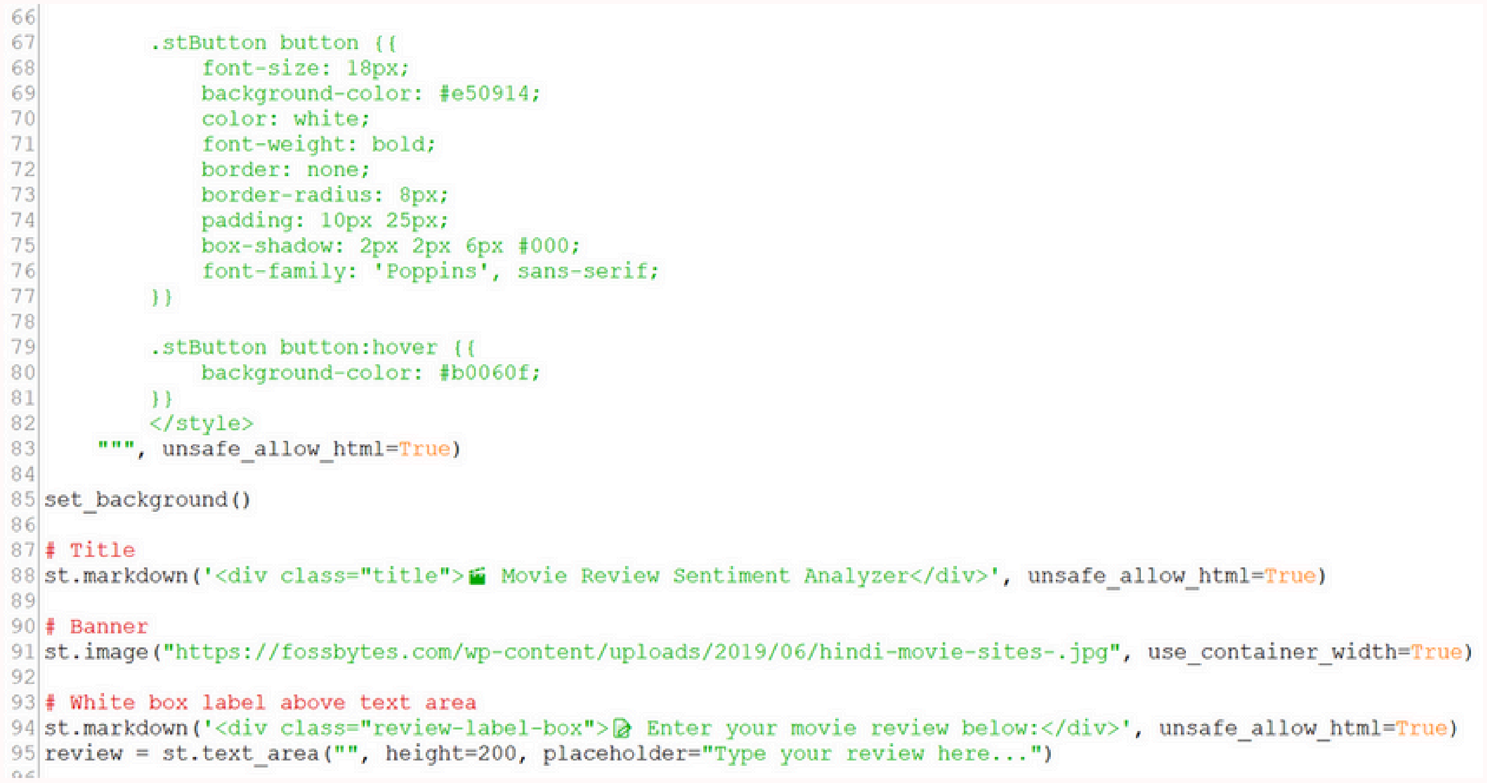
2. app.py (Streamlit Web Application) This file constitutes the user-facing part of the project.


app.py - C:\Users\vishv\Downloads\app.py (3.12.4)
File Edit Format Run Options Window Help

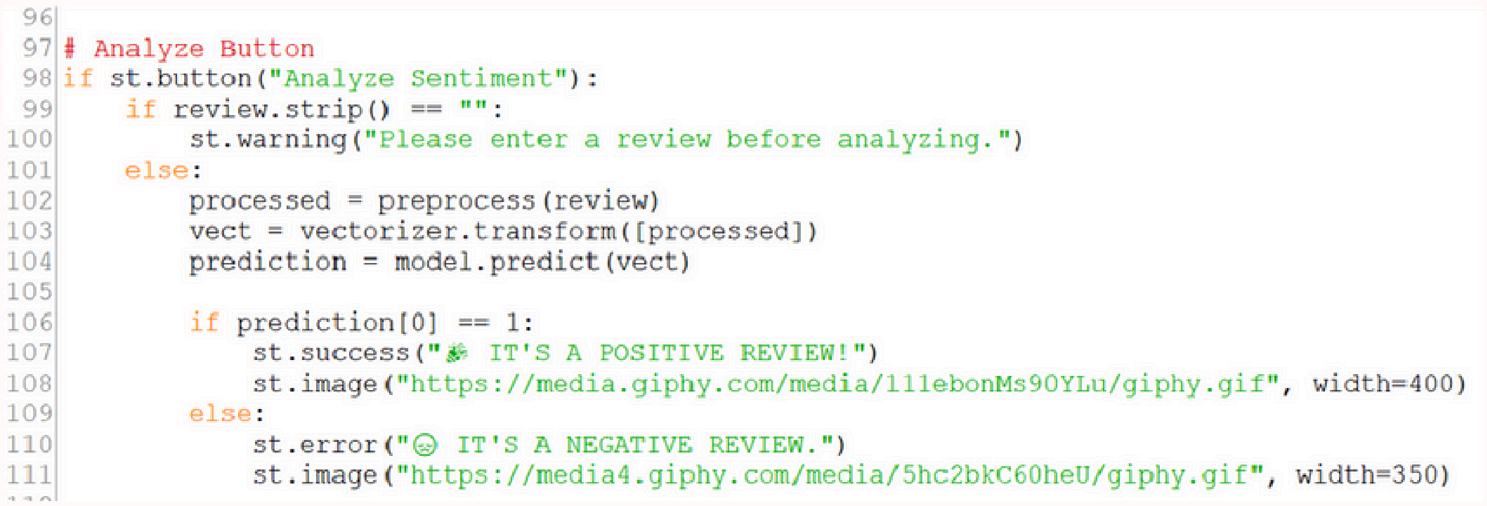
```python
import streamlit as st
import pickle
import re
from nltk.stem import PorterStemmer
import nltk

# Download stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords

# Load model and vectorizer
model = pickle.load(open("Sentiment Analysis on Movie Review.pkl", "rb"))
vectorizer = pickle.load(open("count-Vectorizer.pkl", "rb"))

# Preprocessing
ps = PorterStemmer()
def preprocess(text):
    review = re.sub("[^a-zA-Z]", " ", text)
    review = review.lower().split()
    review = [ps.stem(word) for word in review if word not in stopwords.words("english")]
    return " ".join(review)

# Page config
st.set_page_config(page_title="Movie Review Sentiment Analyzer", layout="centered")

# Background & style
def set_background():
    background_url = "https://wallpaperbat.com/img/426398-red-seat-cinema-and-theatre-hd-4k-wallpaper-and-background.jpg"
    st.markdown(f"""
        <style>
        .stApp {{
            background-image: url("{background_url}");
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
        }}

        .title {{
            color: #fff;
            font-size: 48px;
            font-weight: bold;
            text-align: center;
            padding: 20px;
            text-shadow: 2px 2px 6px #000;
        }}

        .review-label-box {{
            background-color: rgba(240, 240, 240, 0.85);
            padding: 12px 18px;
            border-radius: 12px;
            box-shadow: 2px 2px 8px rgba(0, 0, 0, 0.25);
            font-size: 18px;
```



```python
            font-weight: 500;
            font-family: 'Poppins', sans-serif;
            color: #222;
            margin-bottom: 15px;
        }}

        .stTextArea textarea {{
            font-size: 16px !important;
            background-color: rgba(255, 255, 255, 0.88);
            color: #000;
            font-weight: 500;
            font-family: 'Poppins', sans-serif;
        }}
```



```
        .stButton button {{
            font-size: 18px;
            background-color: #e50914;
            color: white;
            font-weight: bold;
            border: none;
            border-radius: 8px;
            padding: 10px 25px;
            box-shadow: 2px 2px 6px #000;
            font-family: 'Poppins', sans-serif;
        }}

        .stButton button:hover {{
            background-color: #b0060f;
        }}
        </style>
    """, unsafe_allow_html=True)

set_background()

# Title
st.markdown('<div class="title">🎬 Movie Review Sentiment Analyzer</div>', unsafe_allow_html=True)

# Banner
st.image("https://fossbytes.com/wp-content/uploads/2019/06/hindi-movie-sites-.jpg", use_container_width=True)

# White box label above text area
st.markdown('<div class="review-label-box">📝 Enter your movie review below:</div>', unsafe_allow_html=True)
review = st.text_area("", height=200, placeholder="Type your review here...")
```

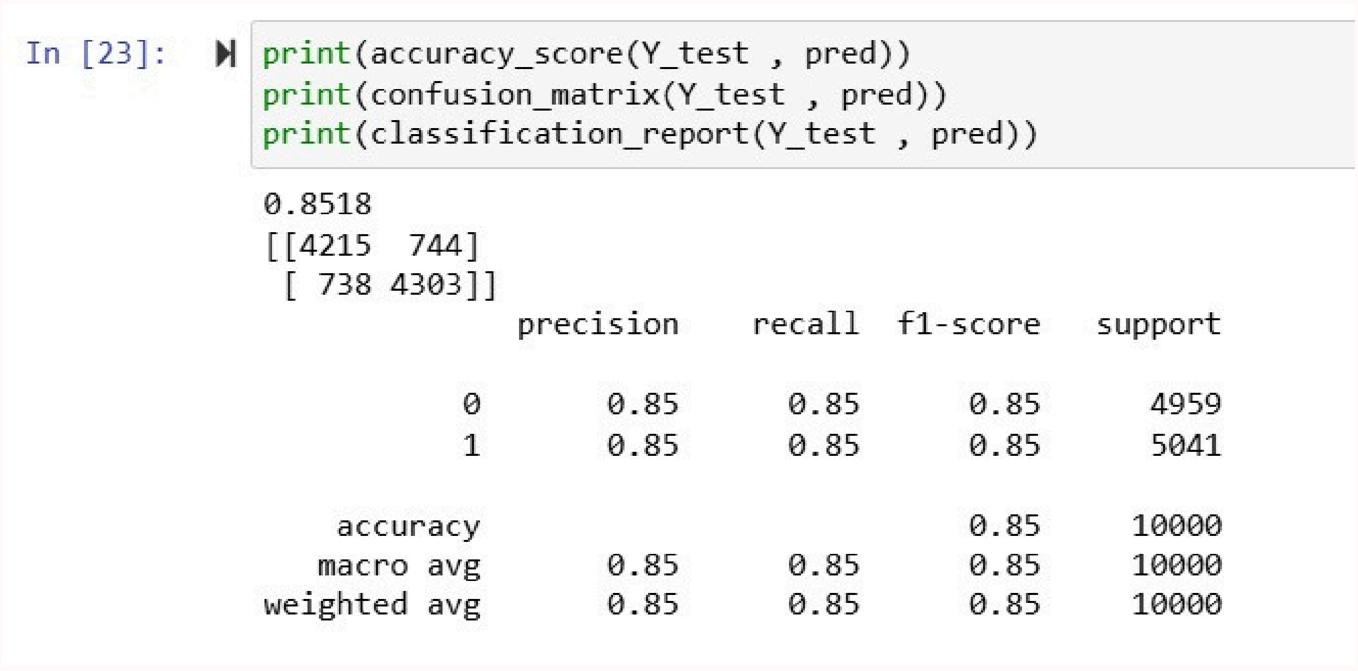


```
# Analyze Button
if st.button("Analyze Sentiment"):
    if review.strip() == "":
        st.warning("Please enter a review before analyzing.")
    else:
        processed = preprocess(review)
        vect = vectorizer.transform([processed])
        prediction = model.predict(vect)

        if prediction[0] == 1:
            st.success("🎉 IT'S A POSITIVE REVIEW!")
            st.image("https://media.giphy.com/media/111ebonMs90YLu/giphy.gif", width=400)
        else:
            st.error("😞 IT'S A NEGATIVE REVIEW.")
            st.image("https://media4.giphy.com/media/5hc2bkC60heU/giphy.gif", width=350)
```

Results and Observations

The sentiment analysis model achieved a high accuracy in classifying movie reviews, indicating its effectiveness in distinguishing between positive and negative sentiments.

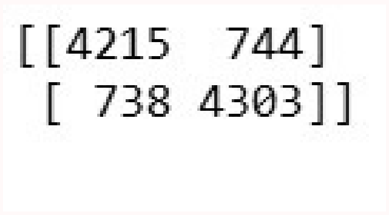
- **Accuracy:**



```
In [23]:  print(accuracy_score(Y_test , pred))
          print(confusion_matrix(Y_test , pred))
          print(classification_report(Y_test , pred))

0.8518
[[4215  744]
 [ 738 4303]]
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      4959
           1       0.85      0.85      0.85      5041

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

- **Classification Report:** The classification report showed strong precision, recall, and F1-scores for both positive and negative classes, suggesting the model performs well across both categories.

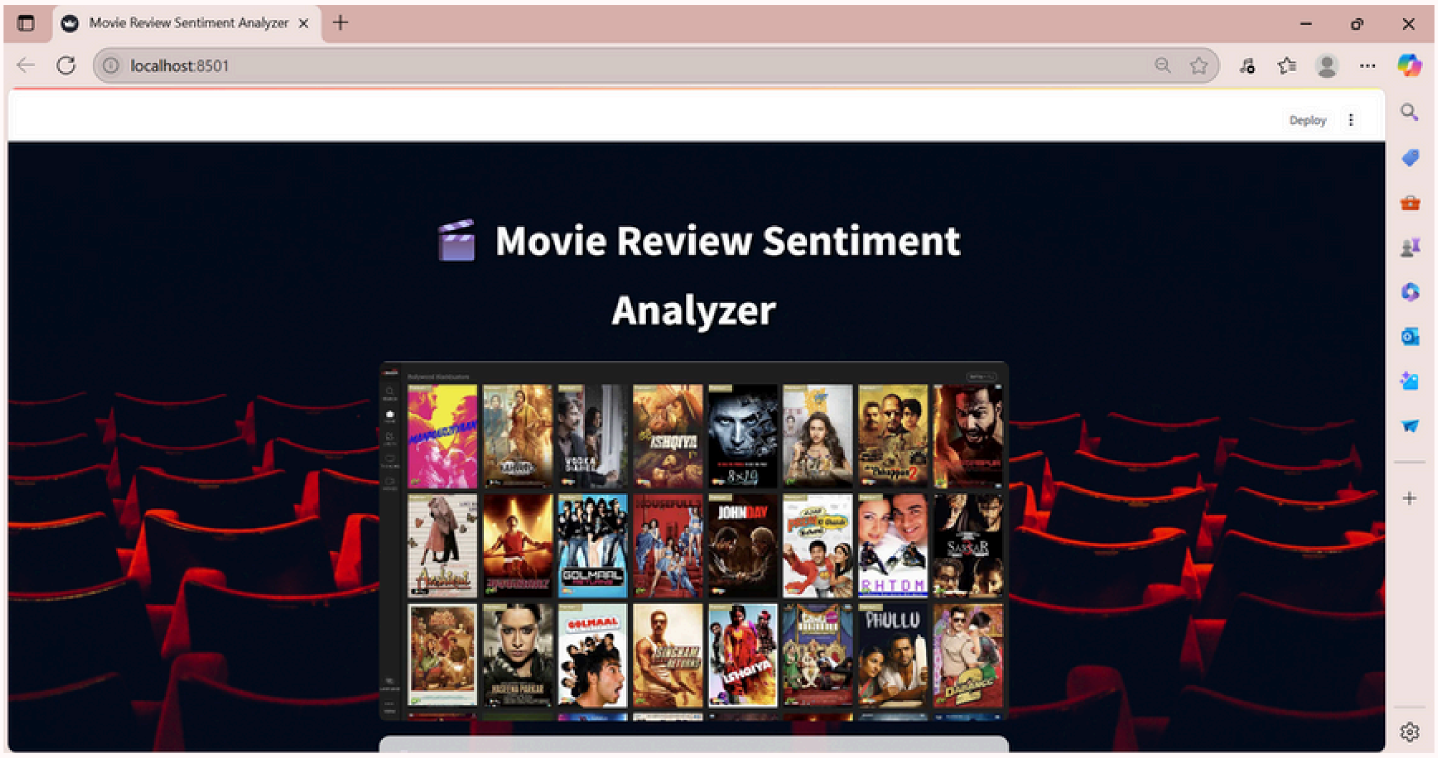
- **Confusion Matrix:** The confusion matrix indicated a low number of false positives and false negatives, further reinforcing the model's ability to correctly classify reviews.

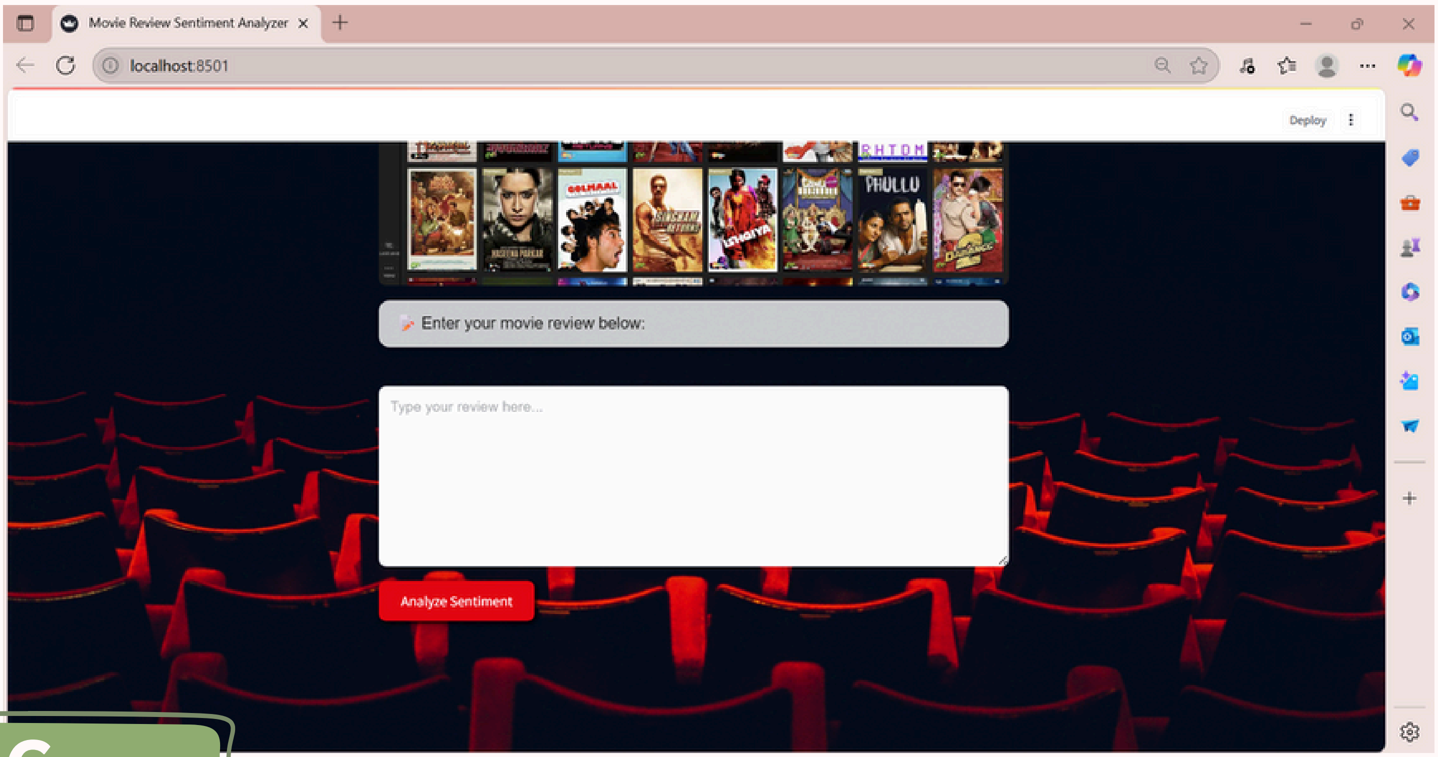


```
[[4215  744]
 [ 738 4303]]
```

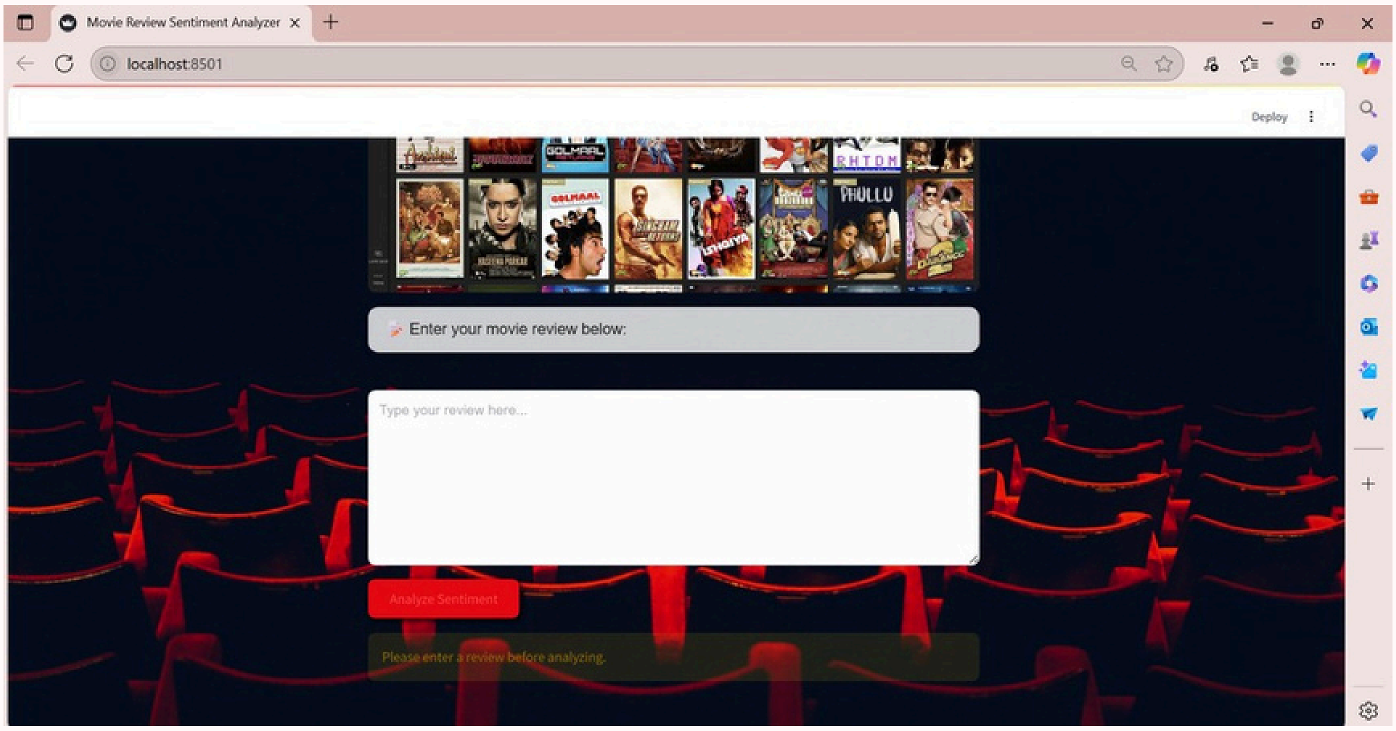
- Application Functionality: The Streamlit application successfully integrates the trained model, providing a real-time sentiment analysis tool. Users can input any movie review, and the application promptly returns an accurate sentiment prediction.

USER INTERFACE


Movie Review Sentiment Analyzer
localhost:8501
Deploy
Movie Review Sentiment Analyzer


Movie Review Sentiment Analyzer
localhost:8501
Deploy
Enter your movie review below:
Type your review here...
Analyze Sentiment

"The application prompts the user to enter a movie review before performing sentiment analysis, ensuring input validation."

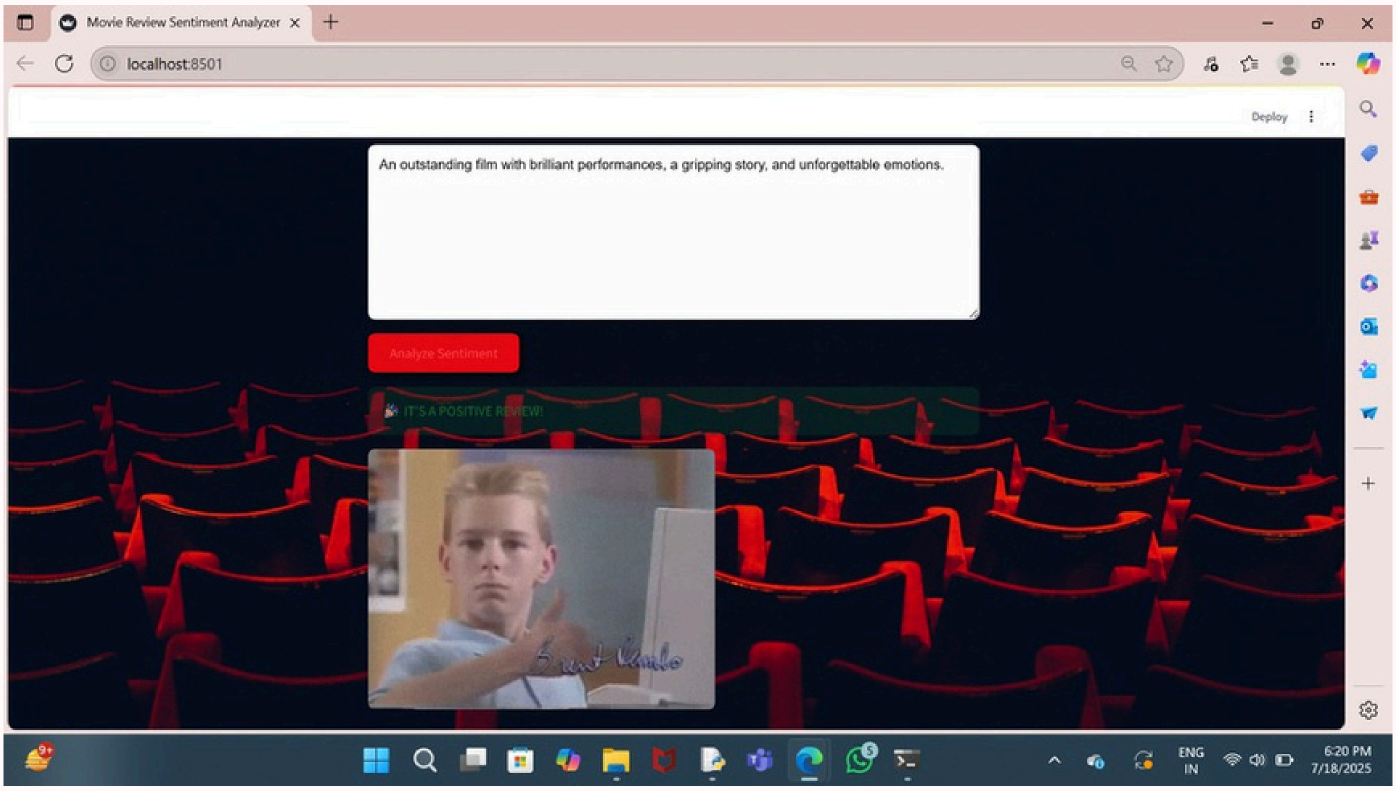


Enter your movie review below:

Type your review here...

Please enter a review before analyzing.

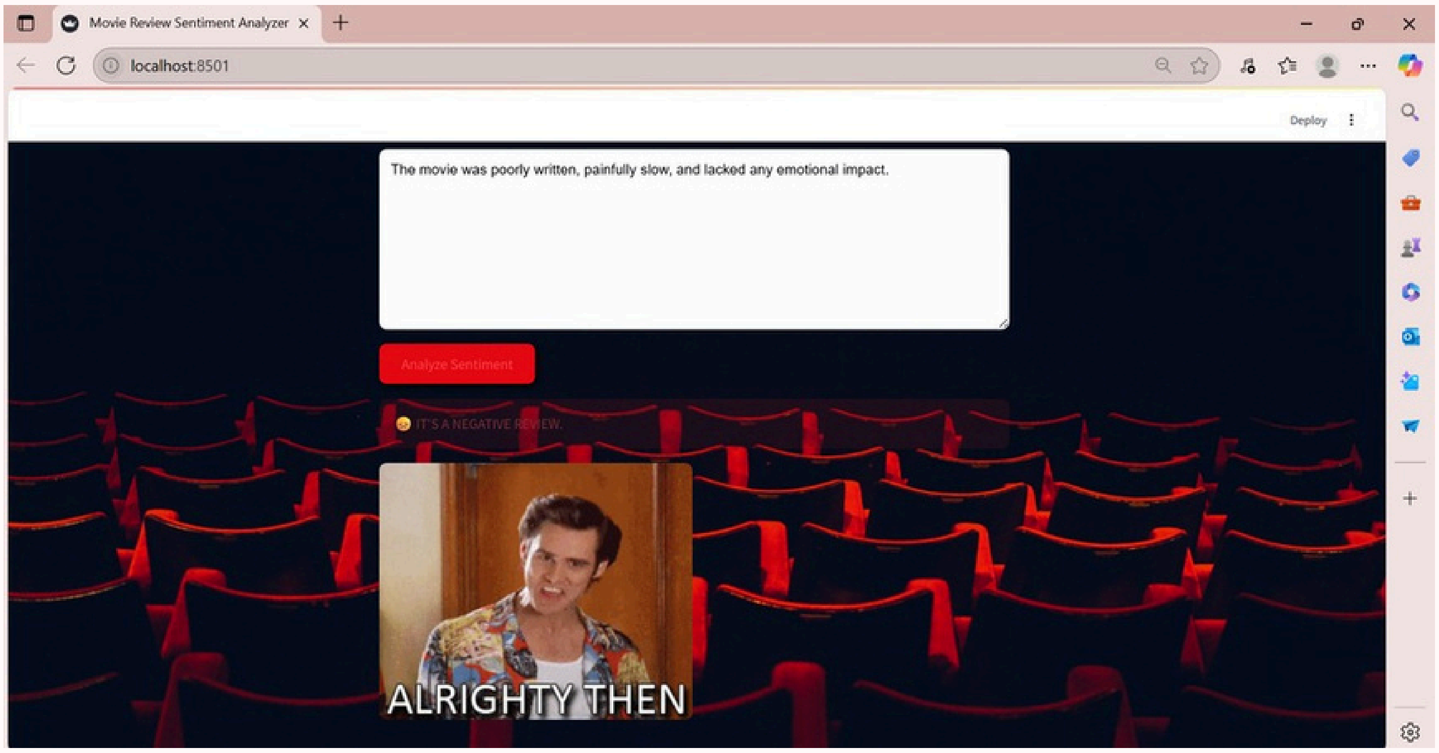
"The system correctly identifies the input as a positive movie review, displaying a confirmation message along with a supportive visual GIF."



An outstanding film with brilliant performances, a gripping story, and unforgettable emotions.

IT'S A POSITIVE REVIEW!

"The system accurately classifies the input as a negative movie review, displaying an appropriate message and a reinforcing visual indicator."



The movie was poorly written, painfully slow, and lacked any emotional impact.

😞 IT'S A NEGATIVE REVIEW.

- **User Experience:** The custom styling and intuitive layout of the Streamlit app enhance the user experience, making the sentiment analysis accessible and engaging.
- **Visual Feedback:** The use of relevant GIFs for positive and negative predictions provides immediate and clear visual feedback to the user.

CONCLUSION

- This project successfully developed and implemented a Movie Review Sentiment Analyzer, aligning with all stated objectives. We effectively built a machine learning model capable of accurately classifying movie review sentiments and deployed it as a functional and user-friendly web application.
- The methodology adopted, from detailed text preprocessing using NLTK to employing a robust classification algorithm like Multinomial Naive Bayes, proved highly effective. The use of Streamlit for the front-end ensured a quick and intuitive deployment, making the powerful sentiment analysis model accessible to end-users.
- Key learnings from this project include the importance of thorough text preprocessing in NLP tasks, the seamless integration of machine learning models into web applications, and the power of libraries like Streamlit for rapid prototyping and deployment. The project successfully demonstrates a comprehensive understanding of sentiment analysis principles and their practical application, providing a valuable tool for understanding public opinion from textual data.

THANKYOU