Part A: IMDb Movie Review Sentiment Analysis

Name = Manoj Kumar

Batch = 1st September Batch

Course = Data Science Placement Guarantee Course

Email = manojkumarrajput@gmail.com

Part A Explaination Video link = https://drive.google.com/file/d/108EnNXRl3jvXNqGVWFZ1GSRvqusp=drivesdk

1. Overview

Sentiment analysis is a natural language processing (NLP) task that involves determining

whether a given text expresses a positive or negative sentiment. In this project, we will

analyze movie reviews from the IMDb dataset and predict the sentiment (positive or

negative) based on the text of the reviews. By leveraging various text preprocessing

techniques, feature extraction methods, and classification algorithms, this project will

develop a machine learning model capable of accurately predicting the sentiment of movie

reviews. The insights derived from this analysis can be useful for movie producers, critics,

and platforms like IMDb to understand public opinion and tailor marketing or content

strategies accordingly.

2. Problem Statement

The primary objective of this project is to build a machine learning classification model that

can predict the sentiment of IMDb movie reviews. The dataset contains a collection of movie

reviews, and each review is labeled as either positive or negative.

Using text preprocessing, feature extraction techniques (such as TF-IDF), and various

classification algorithms, the project will aim to develop a model that can effectively classify

the sentiment of movie reviews. The model's performance will be evaluated using standard

classification metrics, such as accuracy, precision, recall, and F1-score.

1. Data Exploration and Preprocessing

(a) Importing necessary libraries and loading the dataset for analysis.

In [3]:
```
import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
!pip install wordcloud
from wordcloud import WordCloud
from sklearn.metrics import roc_curve, auc
!pip install tensorflow
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages
(1.9.4)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.11/dist-packag
es (from wordcloud) (1.26.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (fr
om wordcloud) (11.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages
(from wordcloud) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pa
ckages (from matplotlib->wordcloud) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packag
es (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib->wordcloud) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib->wordcloud) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pac
kages (from matplotlib->wordcloud) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pa
ckages (from matplotlib->wordcloud) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dis
t-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil>=2.7->matplotlib->wordcloud) (1.17.0)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages
(2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-pack
ages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-p
ackages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/li
b/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-pa
ckages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-p
ackages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages
(from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.
4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from tenso
rflow) (4.25.6)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
(from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-package
s (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-pa
ckages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.1
1/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packa
```

```
ges (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (1.70.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packag
es (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packag
es (from tensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/li
b/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-
packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from
keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (fro
m keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (fr
om keras>=3.5.0->tensorflow) (0.14.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.1
1/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packag
es (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-
packages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-
packages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-pac
kages (from tensorboard<2.19,>=2.18->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/l
ib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-pac
kages (from tensorboard<2.19,>=2.18->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-p
ackages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/di
st-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/
dist-packages (from rich->keras>=3.5.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages
(from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
```

Loading IMDb Dataset

In [7]:
```
## data = pd.read_csv("C:\\Users\\msgme\\Downloads\\Imdb_Data.csv")
data = pd.read_csv("/content/Imdb_Data.csv", on_bad_lines="skip")
```

In [8]:
```
df = pd.DataFrame(data)
```

Checking total number of the rows.

In [9]:
```
print("Total number of rows:", df.shape[0])
```

```
Total number of rows: 50000
```

Checking number of unique reviews

In [10]:
```
print("Number of unique reviews:", df['review'].nunique())
```

```
Number of unique reviews: 49581
```

In [11]:
```
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
```

First five rows of the data

In [12]:
```
df.head()
```

Out[12]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. 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 [13]:
```
df.describe(include='all')
```

Out[13]:

	review	sentiment
count	50000	50000
unique	49581	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

Creating function to count words

In [14]:
```
word_counts = df['review'].apply(lambda x: len(x.split()))
```

In [15]:
```
print(word_counts)
```

```
0        307
1        162
2        166
3        138
4        230
        ...
49995    194
49996    112
49997    230
49998    212
49999    129
Name: review, Length: 50000, dtype: int64
```

Creating function to check length of review column

In [16]:
```
vocab = set()
for text in df['review']:
    words = text.split()
    vocab.update(words)
```

In [17]:
```
len(vocab)
```

Out[17]: 438711

Creating a new column for review length

In [18]:
```
df['review_length'] = df['review'].apply(lambda x: len(x.split()))
```

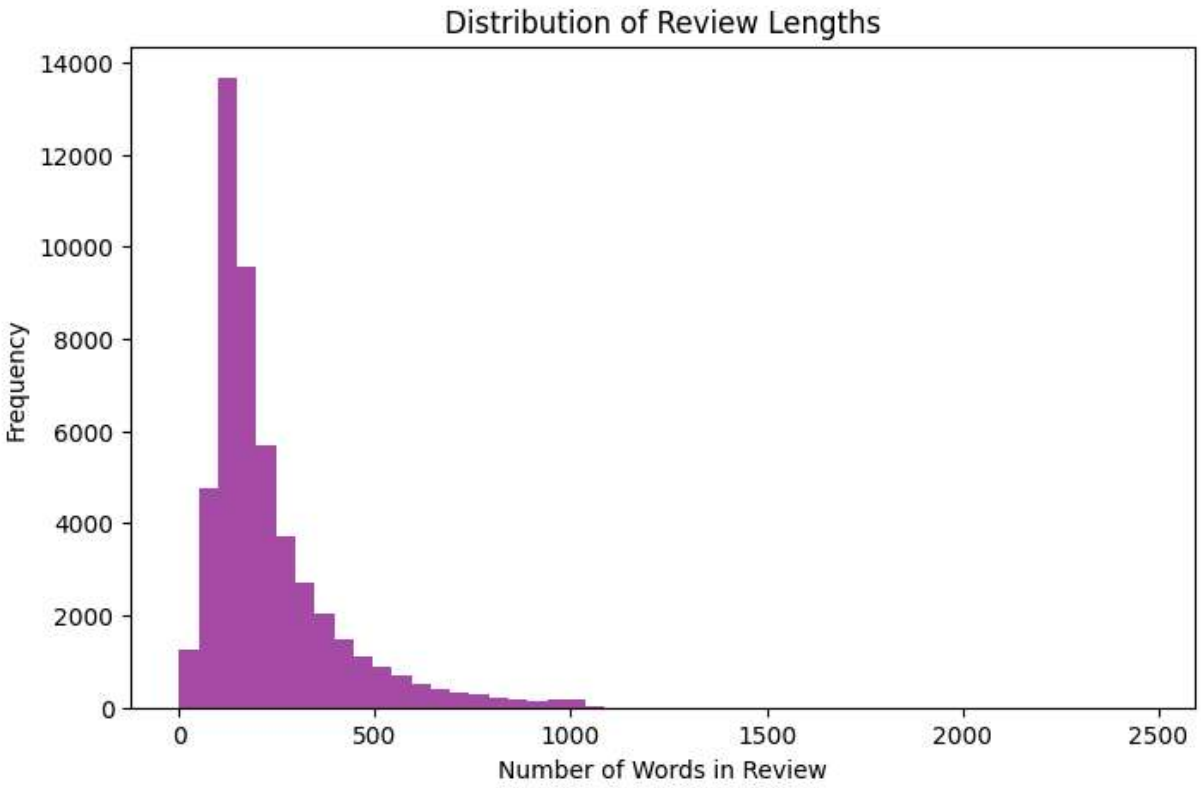
Summary statistics of review lengths

In [19]:
```
print(df['review_length'].describe())
```

```
count    50000.000000
mean       231.146580
std        171.349956
min          1.000000
25%        126.000000
50%        173.000000
75%        280.000000
max       2470.000000
Name: review_length, dtype: float64
```

Visualizing review length distribution

In [20]:
```
plt.figure(figsize=(8,5))
plt.hist(df['review_length'], bins=50, color='purple', alpha=0.7)
plt.title("Distribution of Review Lengths")
plt.xlabel("Number of Words in Review")
plt.ylabel("Frequency")
plt.show()
```



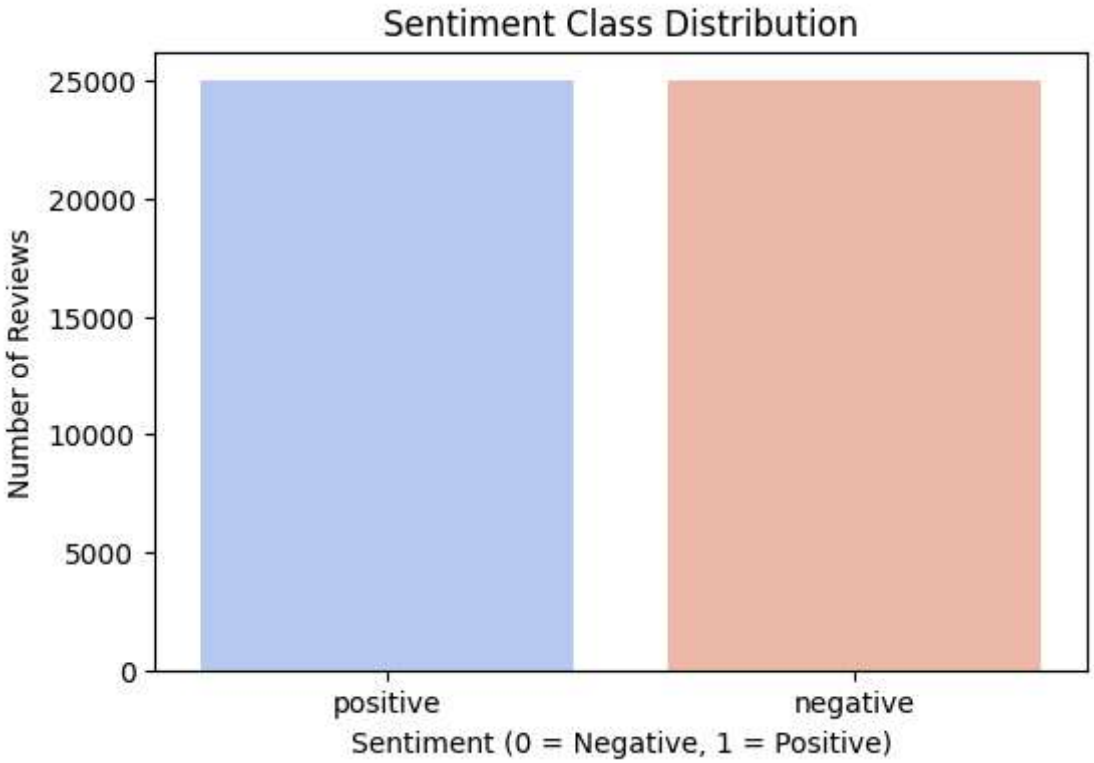
Counting the number of positive and negative reviews

In [21]:
```
class_counts = df['sentiment'].value_counts()
print(class_counts)
```

```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

Visualizing class distribution

In [22]:
```
plt.figure(figsize=(6,4))
sns.barplot(x=class_counts.index, y=class_counts.values, palette="coolwarm")
plt.title("Sentiment Class Distribution")
plt.xlabel("Sentiment (0 = Negative, 1 = Positive)")
plt.ylabel("Number of Reviews")
plt.show()
```



Performing data cleaning

Checking for Missing Data

Checking if any missing values in the columns.

In [23]:
```
df.isnull().sum()
```

Out[23]:

	0
review	0
sentiment	0
review_length	0

dtype: int64

There are no missing values in any column of the data.

Checking Duplicates

In [24]:
```
df.duplicated().sum()
```

Out[24]: 419

There are 419 duplicates out of 50000, so i am removing it.

In [25]:
```
df_cleaned = df.drop_duplicates(subset=["review"], keep="first")
```

In [26]:
```
df_cleaned.shape[0]
```

Out[26]: 49581

In [27]:
```
word_counts.sum()
```

Out[27]: 11557329

(c) Text Preprocessing (Removing stopwords, punctuation, tokenization, and lemmatization)

In [28]:
```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

Creating a function to convert text into lowercase than remove stopwords, punctuation, tokenization, and lemmatization.

In [29]:
```
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())
    tokens = word_tokenize(text)
    filtered_tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in
    return ' '.join(filtered_tokens)
```

Using Preprocess_text function into review column.

In [30]:
```
import nltk
nltk.download('punkt')
nltk.download('all')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_eng to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_rus to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_rus.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package bcp47 to /root/nltk_data...
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/europarl_raw.zip.
[nltk_data]    | Downloading package extended_omw to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package floresta to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/floresta.zip.
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v15.zip.
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v17.zip.
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ieer.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package indian to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/indian.zip.
[nltk_data]    | Downloading package jeita to /root/nltk_data...
[nltk_data]    | Downloading package kimmo to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/kimmo.zip.
[nltk_data]    | Downloading package knbc to /root/nltk_data...
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/large_grammars.zip.
[nltk_data]    | Downloading package lin_thesaurus to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/lin_thesaurus.zip.
[nltk_data]    | Downloading package mac_morpho to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/mac_morpho.zip.
[nltk_data]    | Downloading package machado to /root/nltk_data...
[nltk_data]    | Downloading package masc_tagged to /root/nltk_data...
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package maxent_ne_chunker_tab to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker_tab.zip.
```

```
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger_tab to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/maxent_treebank_pos_tagger_tab.zip.
[nltk_data]    | Downloading package moses_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/moses_sample.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package mte_teip5 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/mte_teip5.zip.
[nltk_data]    | Downloading package mwa_ppdb to /root/nltk_data...
[nltk_data]    |   Unzipping misc/mwa_ppdb.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package nombank.1.0 to /root/nltk_data...
[nltk_data]    | Downloading package nonbreaking_prefixes to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nonbreaking_prefixes.zip.
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]    | Downloading package opinion_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/opinion_lexicon.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package paradigms to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/paradigms.zip.
[nltk_data]    | Downloading package pe08 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pe08.zip.
[nltk_data]    | Downloading package perluniprops to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping misc/perluniprops.zip.
[nltk_data]    | Downloading package pil to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pil.zip.
[nltk_data]    | Downloading package pl196x to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pl196x.zip.
[nltk_data]    | Downloading package porter_test to /root/nltk_data...
[nltk_data]    |   Unzipping stemmers/porter_test.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package problem_reports to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/problem_reports.zip.
[nltk_data]    | Downloading package product_reviews_1 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/product_reviews_1.zip.
[nltk_data]    | Downloading package product_reviews_2 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/product_reviews_2.zip.
```

```
[nltk_data]    | Downloading package propbank to /root/nltk_data...
[nltk_data]    | Downloading package pros_cons to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/pros_cons.zip.
[nltk_data]    | Downloading package ptb to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ptb.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package punkt_tab to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt_tab.zip.
[nltk_data]    | Downloading package qc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/qc.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package rslp to /root/nltk_data...
[nltk_data]    |   Unzipping stemmers/rslp.zip.
[nltk_data]    | Downloading package rte to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/rte.zip.
[nltk_data]    | Downloading package sample_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/sample_grammars.zip.
[nltk_data]    | Downloading package semcor to /root/nltk_data...
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package sentence_polarity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sentence_polarity.zip.
[nltk_data]    | Downloading package sentiwordnet to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sentiwordnet.zip.
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/shakespeare.zip.
[nltk_data]    | Downloading package sinica_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/sinica_treebank.zip.
[nltk_data]    | Downloading package smultron to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/smultron.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package spanish_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/spanish_grammars.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/subjectivity.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package switchboard to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/switchboard.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package tagsets_json to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets_json.zip.
```

```
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet.zip.
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]    |   Unzipping models/wmt15_eval.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/word2vec_sample.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet2022 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet2022.zip.
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

Out[30]: True

In [31]:
```
df_cleaned['cleaned_review'] =df_cleaned ['review'].apply(preprocess_text)
```

In [32]:
```
df_cleaned.head()
```

Out[32]:

	review	sentiment	review_length	cleaned_review
0	One of the other reviewers has mentioned that ...	positive	307	one reviewer mentioned watching 1 oz episode y...
1	A wonderful little production. The...	positive	162	wonderful little production br br filming tech...
2	I thought this was a wonderful way to spend ti...	positive	166	thought wonderful way spend time hot summer we...
3	Basically there's a family where a little boy ...	negative	138	basically there family little boy jake think t...
4	Petter Mattei's "Love in the Time of Money" is...	positive	230	petter matteis love time money visually stunni...

2. Feature Engineering

(a) Vectorize the Text Data (TF-IDF)

Converting the text data into a numerical format using TF-IDF (Term Frequency - Inverse Document Frequency).

In [33]:
```
tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df_cleaned['cleaned_review']).toarray()
```

Converting sparse matrix to DataFrame for better readability

In [34]:
```
tfidf_df = pd.DataFrame(X, columns=tfidf.get_feature_names_out())
print(tfidf_df.head())
```

```
         10  100  1000  1010   11  110   12   13  13th   14  ...  youd  \
0  0.000000  0.0   0.0   0.0  0.0  0.0  0.0  0.0   0.0  0.0  ...   0.0
1  0.000000  0.0   0.0   0.0  0.0  0.0  0.0  0.0   0.0  0.0  ...   0.0
2  0.000000  0.0   0.0   0.0  0.0  0.0  0.0  0.0   0.0  0.0  ...   0.0
3  0.078044  0.0   0.0   0.0  0.0  0.0  0.0  0.0   0.0  0.0  ...   0.0
4  0.000000  0.0   0.0   0.0  0.0  0.0  0.0  0.0   0.0  0.0  ...   0.0

      youll    young  younger     youre  youth  youve  zero    zombie  zone
0  0.057785  0.00000      0.0  0.000000    0.0    0.0   0.0  0.000000   0.0
1  0.000000  0.00000      0.0  0.000000    0.0    0.0   0.0  0.000000   0.0
2  0.000000  0.08008      0.0  0.000000    0.0    0.0   0.0  0.000000   0.0
3  0.000000  0.00000      0.0  0.080777    0.0    0.0   0.0  0.112778   0.0
4  0.000000  0.00000      0.0  0.000000    0.0    0.0   0.0  0.000000   0.0

[5 rows x 5000 columns]
```

In [35]:
```
tfidf_df.head()
```

Out[35]:

	10	100	1000	1010	11	110	12	13	13th	14	...	youd	youll	young
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.057785	0.00000
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.00000
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.08008
3	0.078044	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.00000
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.000000	0.00000

5 rows × 5000 columns

Word count (Number of words in a review)

In [36]:
```
df_cleaned['word_count'] = df_cleaned['cleaned_review'].apply(lambda x: len(x.split
```

Character count (Total number of characters in a review)

In [37]:
```
df_cleaned['char_count'] = df_cleaned['cleaned_review'].apply(lambda x: len(x))
```

Average word length

In [38]:
```
df_cleaned['avg_word_length'] = df_cleaned['char_count'] / df_cleaned['word_count']
```

Displaying first few rows

In [39]:
```
print(df_cleaned[['word_count', 'char_count', 'avg_word_length']].head())
```

```
   word_count  char_count  avg_word_length
0         171        1136         6.643275
1          90         658         7.311111
2          88         584         6.636364
3          72         464         6.444444
4         130         864         6.646154
```

(b) Encoding Sentiment Labels (Positive/Negative)

Converting the textual sentiment labels ('positive' or 'negative') into numeric format for model training.

In [40]:
```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df_cleaned['sentiment'])
```

3. Model Development

(a) Split Data into Training and Testing Sets

Split the data into training and testing sets with an 80-20 ratio.

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

(b) Training Logistic Regression Model

Training a Logistic Regression model to classify the sentiment based on the reviews.

In [42]:
```
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
```

Out[42]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Predictions

In [43]:
```
y_pred_log_reg = log_reg.predict(X_test)
```

Accuracy

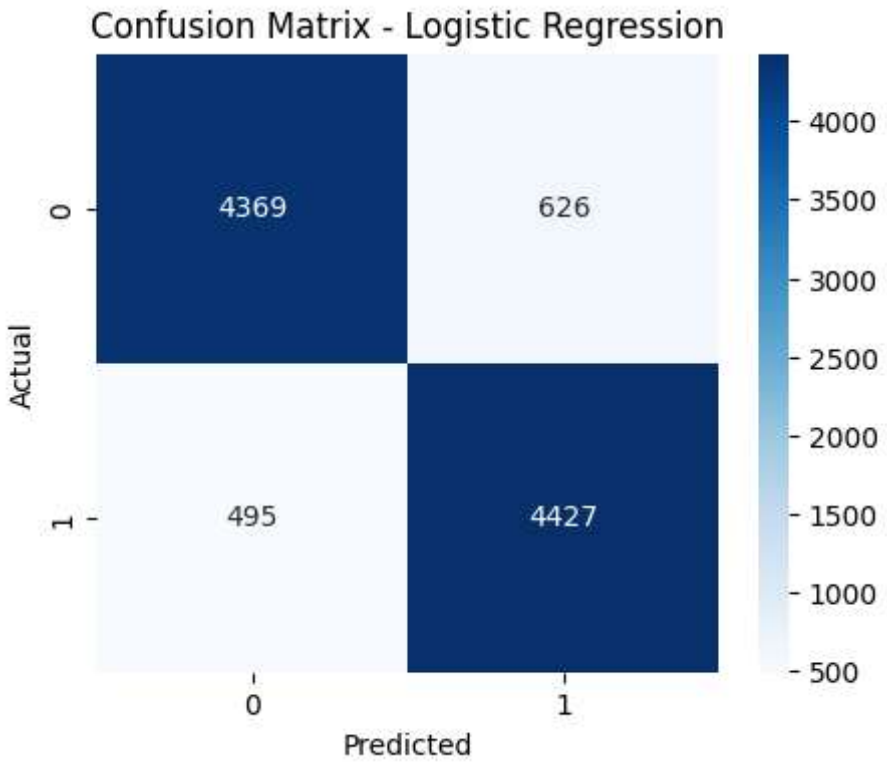
In [44]:
```
print("Classification Report - Logistic Regression:\n", classification_report(y_tes
```

```
Classification Report - Logistic Regression:
               precision    recall  f1-score   support

           0       0.90      0.87      0.89      4995
           1       0.88      0.90      0.89      4922

    accuracy                           0.89      9917
   macro avg       0.89      0.89      0.89      9917
weighted avg       0.89      0.89      0.89      9917
```

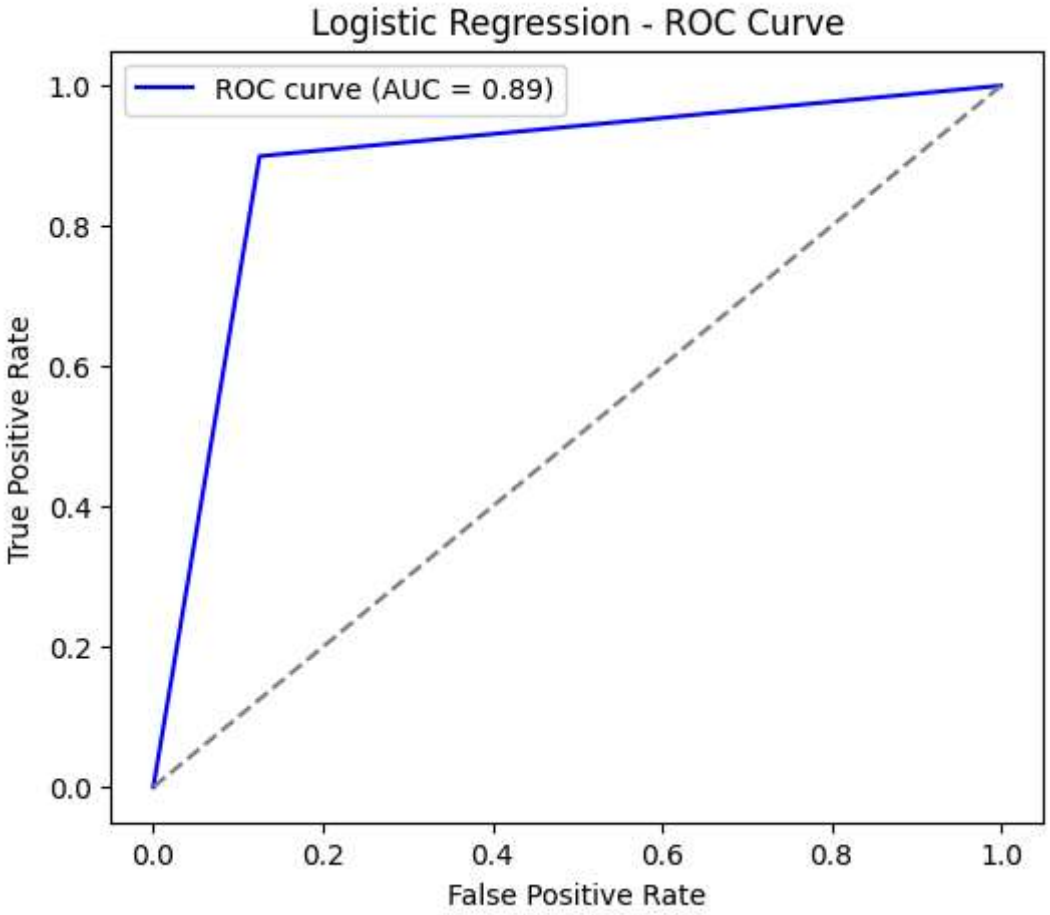
In [45]:
```
cm = confusion_matrix(y_test, y_pred_log_reg)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```



ROC Curve & AUC for Logistic Regression

In [46]:
```
fpr, tpr, _ = roc_curve(y_test, y_pred_log_reg)
roc_auc = auc(fpr, tpr)
```

In [47]:
```
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Regression - ROC Curve")
plt.legend()
plt.show()
```



Training Naive Bayes Model

In [48]:
```
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
```

Out[48]:
```
MultinomialNB
MultinomialNB()
```

Predictions

In [49]:
```
y_pred_nb = nb_model.predict(X_test)
```

Accuracy

In [50]:
```
print("Naïve Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))
```

```
Naïve Bayes Accuracy: 0.8543914490269234
```

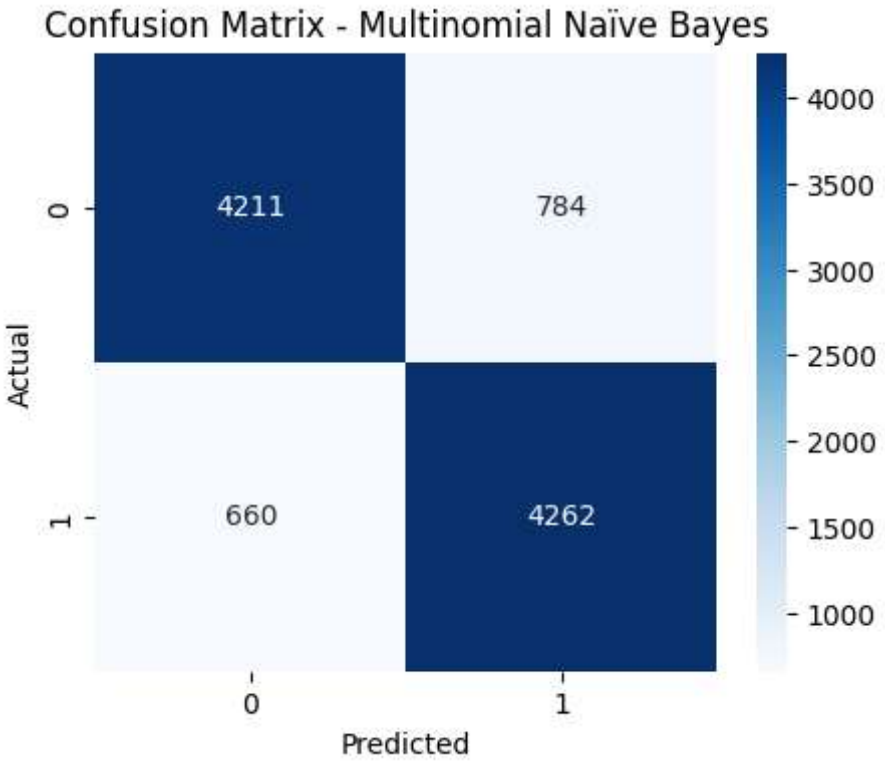
In [51]:
```
print("Classification Report - Multinomial Naïve Bayes:\n", classification_report(y
```

```
Classification Report - Multinomial Naïve Bayes:
              precision    recall  f1-score   support

           0       0.86      0.84      0.85      4995
           1       0.84      0.87      0.86      4922

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

In [52]:
```
cm = confusion_matrix(y_test, y_pred_nb)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Multinomial Naïve Bayes")
plt.show()
```

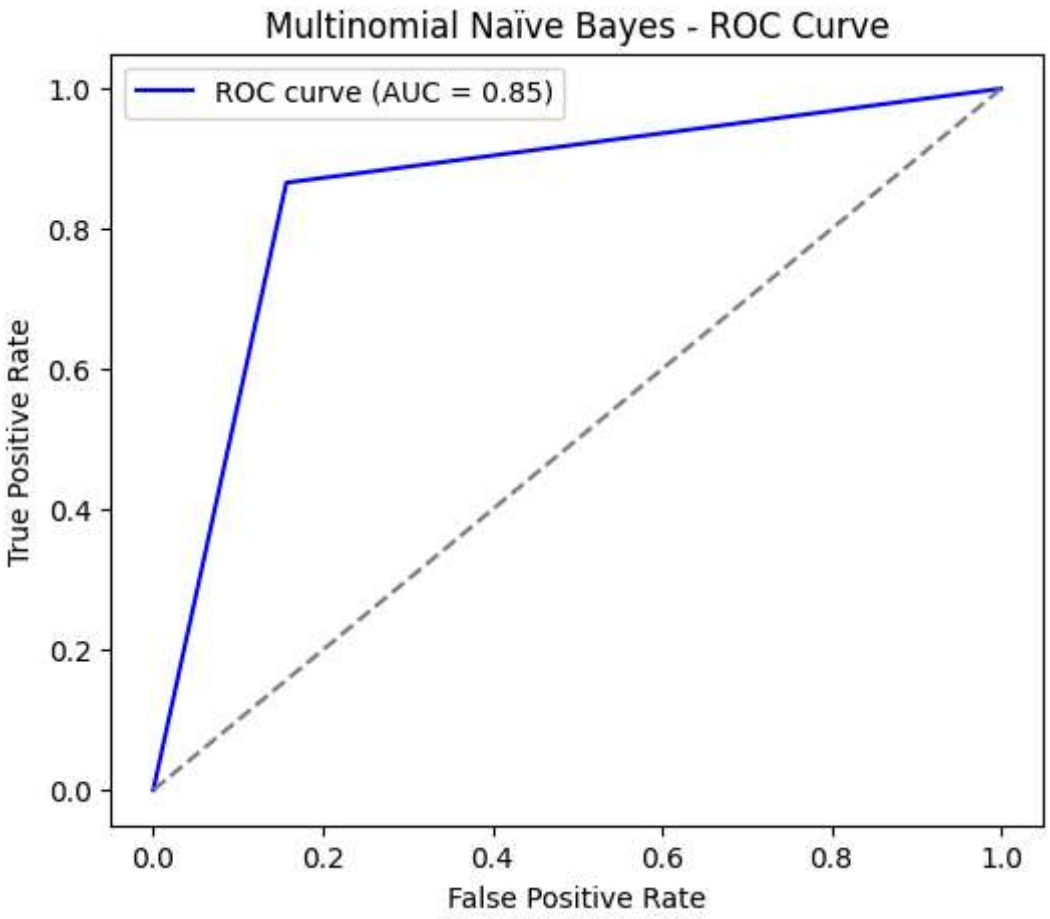


ROC Curve & AUC

In [53]:
```
fpr, tpr, _ = roc_curve(y_test, y_pred_nb)
roc_auc = auc(fpr, tpr)
```

In [54]:
```
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Multinomial Naïve Bayes - ROC Curve")
```

```
plt.legend()
plt.show()
```



Training Random Forest model

In [55]:
```
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)
```

Out[55]:
```
RandomForestClassifier
RandomForestClassifier()
```

Predictions

In [56]:
```
y_pred_rf = rf_model.predict(X_test)
```

Accuracy

In [57]:
```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.844307754361198
```

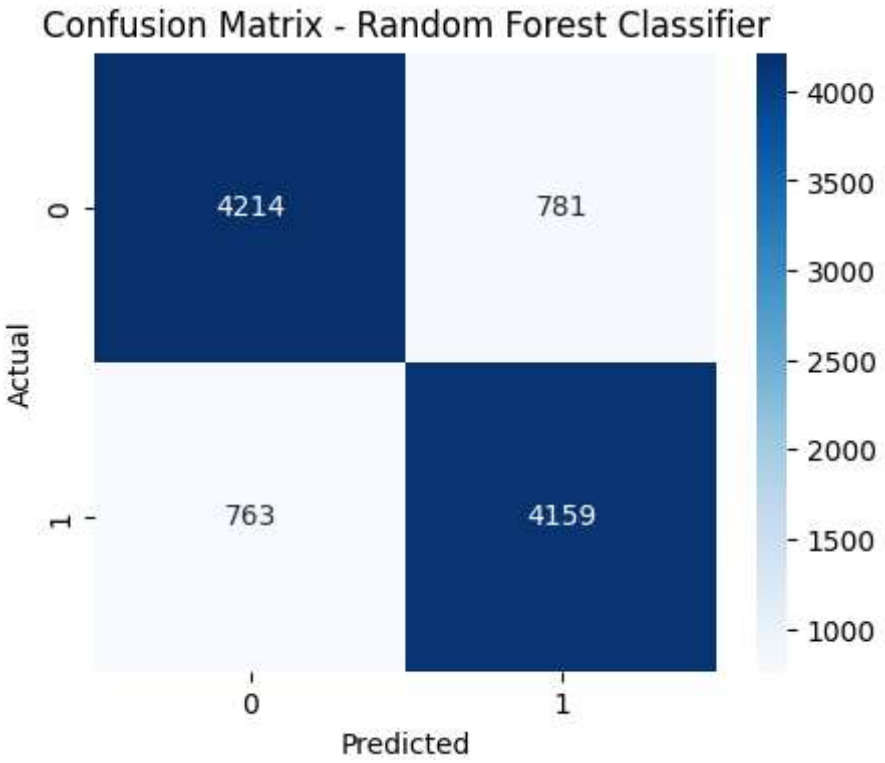
In [58]:
```
print("Classification Report - Random Forest Classifier:\n", classification_report(
```

```
Classification Report - Random Forest Classifier:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85      4995
           1       0.84      0.84      0.84      4922

    accuracy                           0.84      9917
   macro avg       0.84      0.84      0.84      9917
weighted avg       0.84      0.84      0.84      9917
```

In [59]:
```
cm = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Random Forest Classifier")
plt.show()
```

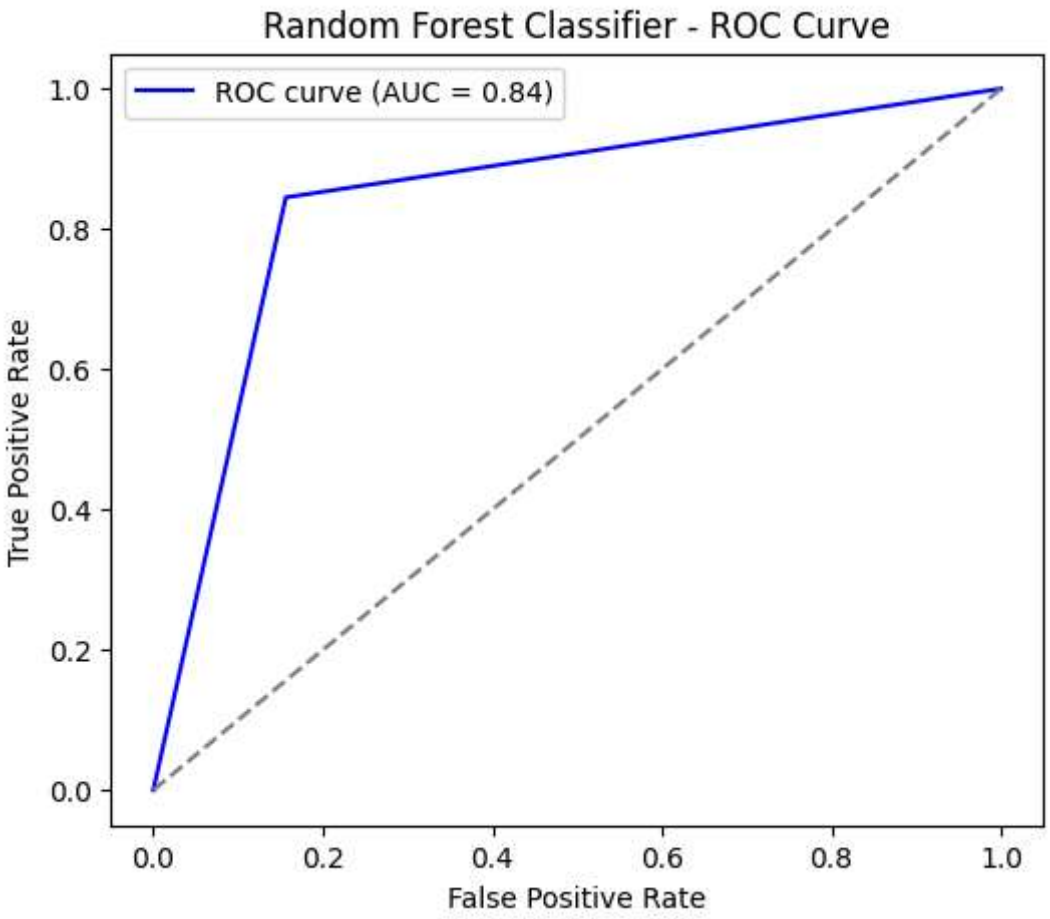


ROC Curve & AUC

In [60]:
```
fpr, tpr, _ = roc_curve(y_test, y_pred_rf)
roc_auc = auc(fpr, tpr)
```

In [61]:
```
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Random Forest Classifier - ROC Curve")
```

```
plt.legend()
plt.show()
```



Training SVM model

In [63]:
```
svm_model = SVC(kernel='linear', C=0.1)
svm_model.fit(X_train, y_train)
```

Out[63]:
```
SVC
SVC(C=0.1, kernel='linear')
```

Predictions

In [64]:
```
y_pred_svm = svm_model.predict(X_test)
```

Accuracy

In [65]:
```
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.8798023595845518
```

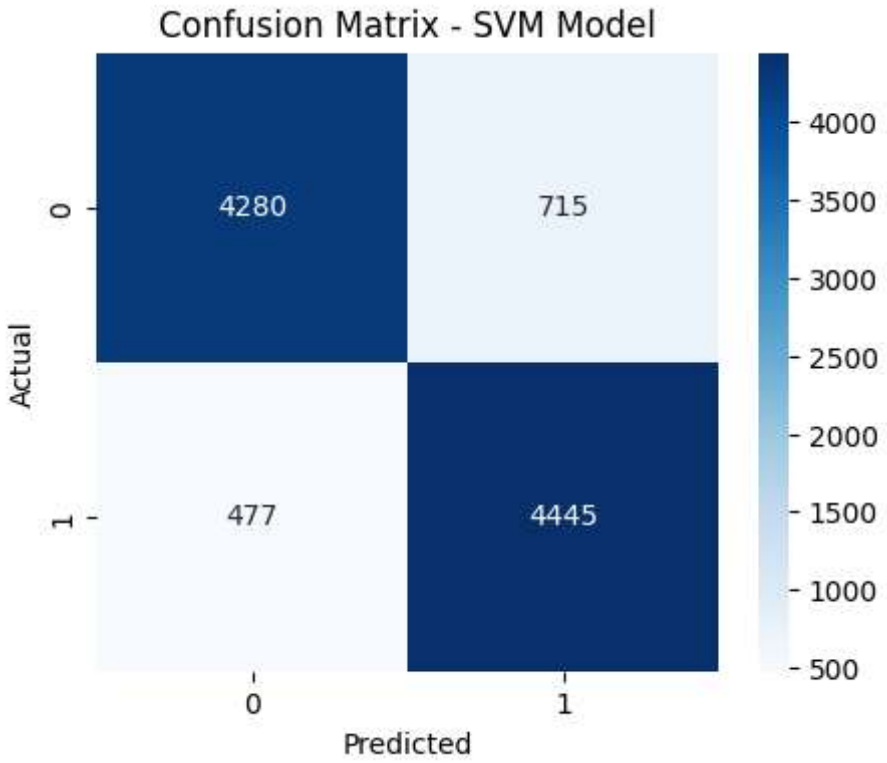
In [66]:
```
print("Classification Report - SVM Model:\n", classification_report(y_test, y_pred_
```

```
Classification Report - SVM Model:
              precision    recall  f1-score   support

           0       0.90      0.86      0.88      4995
           1       0.86      0.90      0.88      4922

    accuracy                           0.88      9917
   macro avg       0.88      0.88      0.88      9917
weighted avg       0.88      0.88      0.88      9917
```

In [67]:
```
cm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - SVM Model")
plt.show()
```

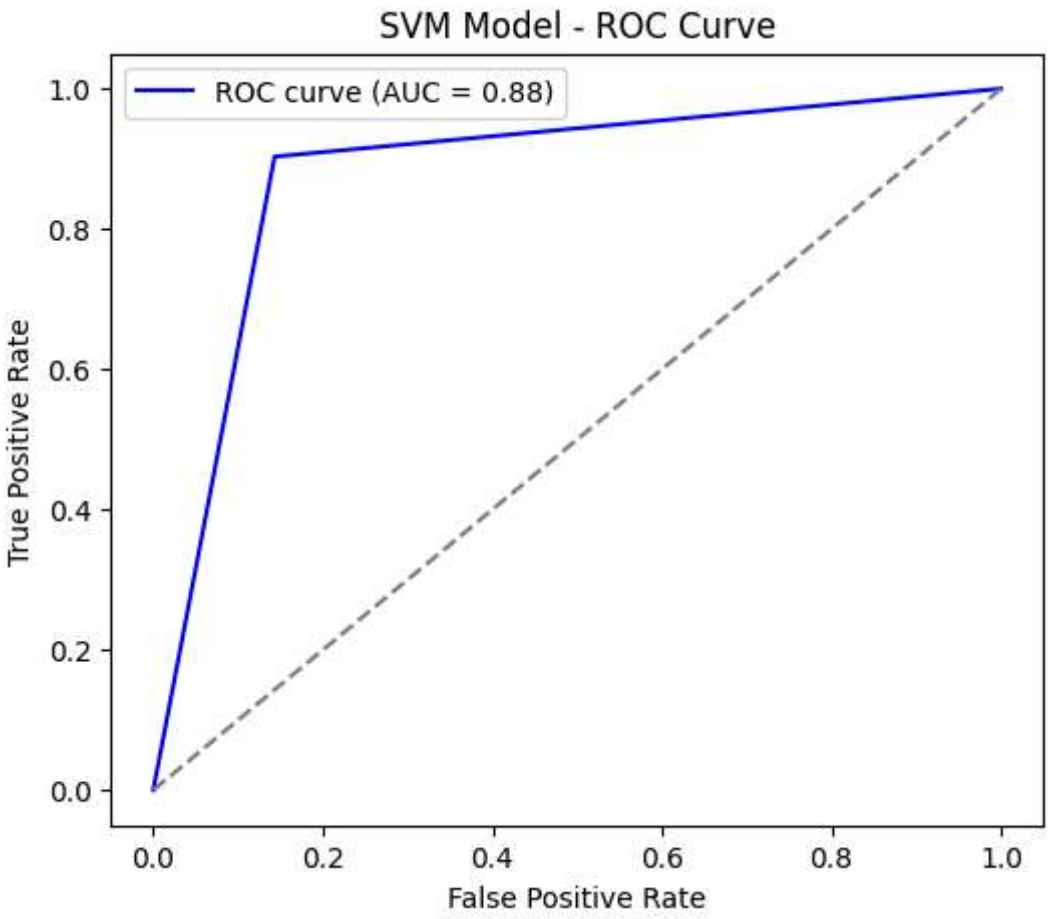


ROC Curve & AUC

In [68]:
```
fpr, tpr, _ = roc_curve(y_test, y_pred_svm)
roc_auc = auc(fpr, tpr)
```

In [69]:
```
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("SVM Model - ROC Curve")
```

```
plt.legend()
plt.show()
```



Definin LSTM for Sentiment Analysis

In [70]:
```
lstm_model = Sequential([
    Embedding(input_dim=5000, output_dim=128, input_length=X_train.shape[1]),LSTM(6
```

Compile model

In [71]:
```
lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy
```

Training model

lstm_model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))

In [73]:
```
lstm_model.fit(X_train, y_train, batch_size=32, epochs=3, validation_data=(X_test,
```

```
Epoch 1/3
1240/1240 ━━━━━━━━━━━━━━━━━━━━ 352s 280ms/step - accuracy: 0.5009 - loss: 0.6939 - v
al_accuracy: 0.4963 - val_loss: 0.6935
Epoch 2/3
1240/1240 ━━━━━━━━━━━━━━━━━━━━ 377s 278ms/step - accuracy: 0.5010 - loss: 0.6932 - v
al_accuracy: 0.4963 - val_loss: 0.6933
Epoch 3/3
1240/1240 ━━━━━━━━━━━━━━━━━━━━ 372s 271ms/step - accuracy: 0.5044 - loss: 0.6932 - v
al_accuracy: 0.4963 - val_loss: 0.6932
```

Out[73]: <keras.src.callbacks.history.History at 0x7b24b00c3510>

Evaluating the model

In [74]:
```
loss, accuracy = lstm_model.evaluate(X_test, y_test)
print(f"LSTM Test Accuracy: {accuracy:.4f}")
```

```
310/310 ━━━━━━━━━━━━━━━━━━━━ 35s 112ms/step - accuracy: 0.4992 - loss: 0.6932
LSTM Test Accuracy: 0.4963
```

Predictions

In [75]:
```
y_probs = lstm_model.predict(X_test)
y_pred = (y_probs > 0.5).astype("int32")
```

```
310/310 ━━━━━━━━━━━━━━━━━━━━ 34s 109ms/step
```

Classification Report

In [76]:
```
print("LSTM Classification Report:\n", classification_report(y_test, y_pred, target
```

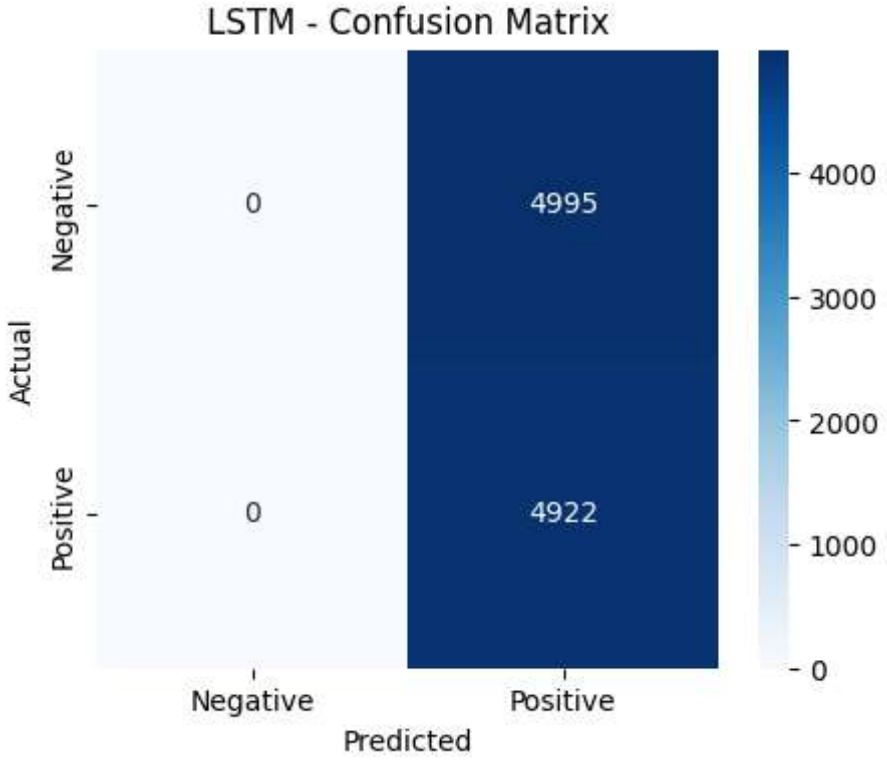
```
LSTM Classification Report:
               precision    recall  f1-score   support

    Negative       0.00      0.00      0.00      4995
    Positive       0.50      1.00      0.66      4922

    accuracy                           0.50      9917
   macro avg       0.25      0.50      0.33      9917
weighted avg       0.25      0.50      0.33      9917
```

Confusion Matrix

In [77]:
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positi
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("LSTM - Confusion Matrix")
plt.show()
```



LSTM model is not working properly because it is predicting only one type of result (Positive) for all inputs.

This is why the Negative class has 0 precision, recall, and F1-score, meaning the model is completely ignoring it.

Logistic Regression gives the best results with an accuracy of 88.68%, so it's the best choice for making

predictions. The other models, like SVM (87.97%), MultinomialNB (85.46%), and Random Forest (84.42%), are

not as accurate as Logistic Regression, which means they might not be as reliable for predicting new data.

Logistic Regression is also simpler to understand compared to more complex models like Random Forest or

SVM, so it's often a good option to use.

Making Predictions on New Reviews

Transforming new reviews using trained TF-IDF and Predicting sentiment

In [78]:
```
new_reviews = ["The movie was fantastic! I really enjoyed watching it from start to
new_reviews_tfidf = tfidf.transform(new_reviews)
new_predictions = log_reg.predict(new_reviews_tfidf)
```

Converting back to labels

In [79]:
```
new_predictions_labels = label_encoder.inverse_transform(new_predictions)
for review, sentiment in zip(new_reviews, new_predictions_labels):
    print(f"Review: {review} → Sentiment: {sentiment}")
```

```
Review: The movie was fantastic! I really enjoyed watching it from start to finish.
→ Sentiment: positive
Review: This film was disappointing. I expected more, but it didn’t meet my expectat
ions. → Sentiment: negative
```

Positive Reviews Word Cloud

In [80]:
```
positive_reviews = " ".join(df_cleaned[df_cleaned['sentiment'] == 'positive']['clea
wordcloud_pos = WordCloud(width=600, height=400, background_color="white").generate
```

Negative Reviews Word Cloud

In [81]:
```
negative_reviews = " ".join(df_cleaned[df_cleaned['sentiment'] == 'negative']['clea
wordcloud_neg = WordCloud(width=600, height=400, background_color="black", colormap
```

Ploting Word Clouds

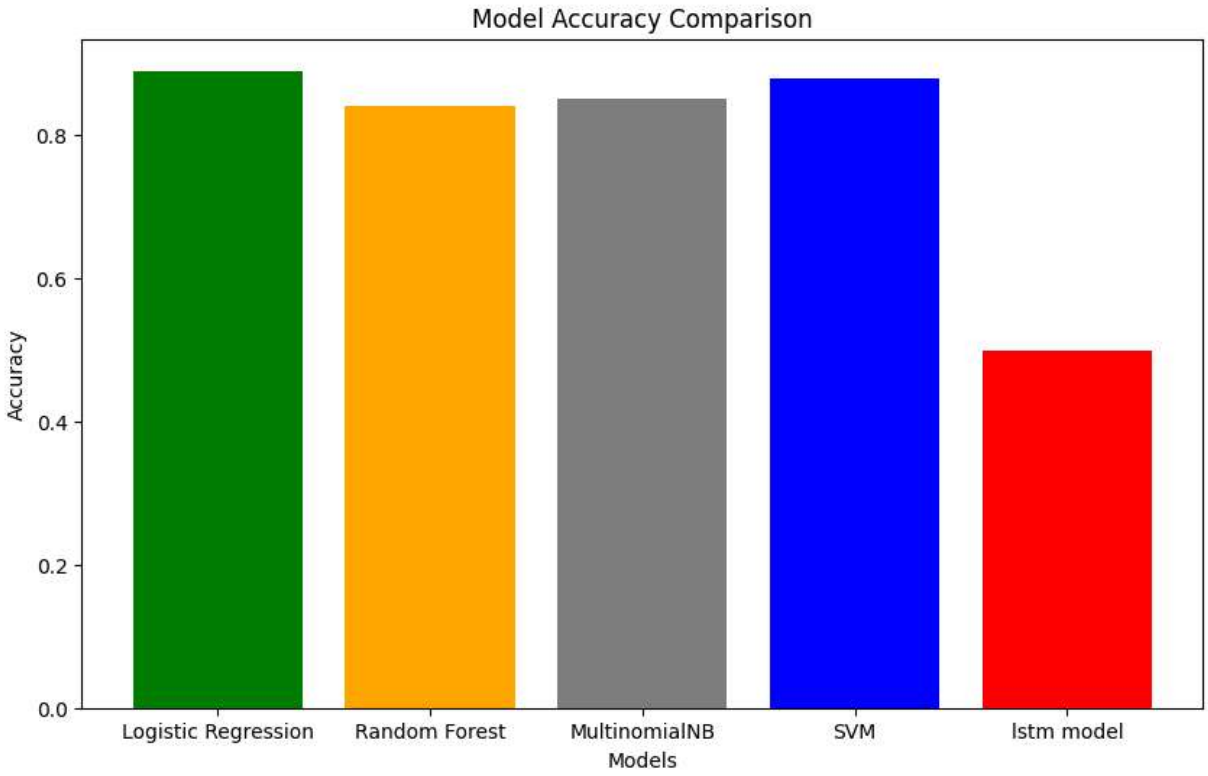
In [82]:
```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.imshow(wordcloud_pos, interpolation="bilinear")
plt.axis("off")
plt.title("Positive Reviews Word Cloud")
plt.subplot(1,2,2)
plt.imshow(wordcloud_neg, interpolation="bilinear")
plt.axis("off")
plt.title("Negative Reviews Word Cloud")
plt.show()
```

Model names and accuracies

In [84]:
```
models = ['Logistic Regression', 'Random Forest', 'MultinomialNB', 'SVM', 'lstm mod
accuracies = [0.89, 0.84, 0.85, 0.88, 0.50]
```

Plotting the bar chart

In [89]:
```
plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color=['green', 'orange', 'grey', 'blue','red'])
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.show()
```



Final Report: Sentiment Analysis of IMDb Movie Reviews

Project Overview:

Sentiment analysis is a technique to check opinion of text , is that positive or negative.

In this project, i worked with IMDb dataset, which contains movie reviews, and try to predict that, is review negative

or positive. I used different methods like text preprocessing, features extraction, and machine learning algorithms, so

that I can develop a model which can predict sentiment accurately. This analysis helps movie producers, critics, and

platforms like IMDb to understand public opinion and to make better marketing strategies.

Objective:

The goal of this project was, to develop a machine learning model, which can predict the sentiment of

movie reviews accurately for IMDb. Given reviews In the dataset are either positive, negative or neutral label.

I used text preprocessing, feature extraction (TF-IDF), and different machine learning algorithms to classify the sentiment.

Then evaluated the models accuracy, F-1 score, precision, and recall metrics.

Data Exploration and Preprocessing

Data Cleaning:

I checked Missing values but there were no missing values in the data.

I checked duplicates there were 419 duplicates I removed the duplicates reviews.

Checked sentiment analysis distribution to ensure data balance.

Analyzed the review length distribution to know how long or short the review is written.

Feature Engineering: Feature engineering

Created a function to convert text reviews into lower case and to remove stopwords and punctuation, to tokenize the text,

to lemmitize the text.

Used TF-IDF method to convert text into numerical format.

Used label encoder to convert sentiment labels into numerical format.

I have also added extra features like word count(number of words), character count (number of characters), and the average word

length in each review.

Visualizations:

I have created a histogram to show the distribution of review lengths.

I made a bar plot to show that the number of positive and negative reviews was balanced (25,000 reviews for each sentiment).

I have generated a word cloud to highlight the most common words in the reviews.

Model Selection and Training

Models Used and Training: I have tested four different machine learning models to predict the sentiment of the movie reviews:

Logistic Regression (max_iter=1000)

Random Forest (n_estimators=100)

Support Vector Machine (SVM) (kernel='linear', C=0.1)

Multinomial Naive Bayes (MultinomialNB)

Model Evaluation Metrics: Here's how each model performed on key evaluation metrics:

Logistic Regression Accuracy(0.89) F1-Score(0.89) Precision(0.88) Recall(0.90) ROC-AUC Score(0.89)

Random Forest Accuracy(0.84) F1-Score(0.84) Precision(0.84) Recall(0.84) ROC-AUC Score(0.84)

MultinomialNB Accuracy(0.85) F1-Score(0.86) Precision(0.84) Recall(0.87) ROC-AUC Score(0.85)

SVM Accuracy(0.88) F1-Score(0.88) Precision(0.86) Recall(0.90) ROC-AUC Score(0.88)

Confusion Matrix it is used to analyze the models performance.

True Positives (TP): Correctly predicted positive reviews.

True Negatives (TN): Correctly predicted negative reviews.

False Positives (FP): Negative reviews wrongly predicted as positive.

False Negatives (FN): Positive reviews wrongly predicted as negative.

For Logistic Regression, the confusion matrix looked like this:

TP = 4371 (Correctly predicted positive reviews)

TN = 4423 (Correctly predicted negative reviews)

FP = 624 (Incorrectly predicted positive reviews)

FN = 499 (Incorrectly predicted negative reviews)

Model Comparison and Best Performing Model

I have compared all the models based on their performance:

Logistic Regression had the highest accuracy of 0.89, meaning it made the correct prediction 89% of the time.

SVM came in second with an accuracy of 0.88.

MultinomialNB scored 0.85, and Random Forest had the lowest accuracy at 0.84.

Best Performing Model: Logistic Regression performed the best in this project, with the highest accuracy (0.89),

F1-score (0.89), and ROC-AUC score (0.89). So, Logistic Regression is the best model for predicting sentiment in

IMDb movie reviews.

Conclusion and Recommendations

Conclusion: The Logistic Regression model was the best performer among all the models tested. It gave the highest

accuracy and F1-score, making it the most reliable for predicting the sentiment of movie reviews.

Recommendations:

To improve the model performence, deep learning models can be used like BERT

which can provide better results.

This model can be deploy on IMDb or other platform, so that movie reviews sentiment can be analysed.

Part A Explaination Video link = https://drive.google.com/file/d/108EnNXRl3jvXNqGVWFZ1GSRvqusp=drivesdk