MACHINE LEARNING MINI PROJECT - ASSIGNMENT 8

MOVIE RECOMMENDATION SYSTEM WITH MOVIE ANALYSER

> Done By: 527 - Aditi Hinge, 528 - Vaishnavi Ingawale, 531 - Shweta Katkar, 537 - Saniya Mahalle

PROBLEM STATEMENT

The objective of this work is to use Machine Learning Algorithms to give users movie reccomendations and a garphical analysis of movie performance based on the feedbacks from the users. The system automatically predicts the sentiment of the feedback that helps in analysing the movies.

INTRODUCTION

The movie industry has always been a popular source of entertainment for people around the world. With the advent of technology, movie streaming services and online movie databases have become more prevalent. However, with so many movies to choose from, it can be overwhelming for users to decide what to watch next. Our website aims to address this issue by providing movie recommendations to users based on their input. Additionally, the website asks users to provide a review for the recommended movies, which are then classified as positive or negative. This information is used to generate a chart that shows how many people liked the movie and how many did not. This allows users to make more informed decisions about what movies to watch next.

Dataset Information

> Movie recommendation system: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

> Movie Review Sentiment Analysis: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

AIM: The objective of this is to use Machine Learning to give movie reccomendations and predict whether the review given by the user is positive or negative.

MOVIE RECCOMENDER

Here are the steps involved in creating a movie recommendation system:

1. Data collection: Collect movie data such as title, genre, cast, crew, and ratings from datasets.
2. Data preprocessing: Clean and preprocess the data to remove duplicates, missing value and irrelevant information.
3. Feature engineering: Extract meaningful features such as genre, cast, director, and ratio from the movie data.
4. Algorithm selection: Choose a suitable algorithm for building the recommendation system, such as collaborative filtering, content-based filtering, or hybrid filtering.

CODE

In [2]:
```
import numpy as np
import pandas as pd
```

In [4]:
```
#reading the csv files
credits = pd.read_csv("credits[1].csv")                          #cast, crew from credits
keywords = pd.read_csv("keywords[1].csv")                        #keywords
movies_metadata_new = pd.read_csv("movies_metadata_new.csv")     #id,genres,title,overview,companies from movies.metadata
```

In [14]:
```
#extraction of required attributes
movies_metadata_new = movies_metadata_new[['id','genres','original_title','overview','production_companies']]
credits = credits[['id','cast','crew']]
keywords = keywords[['id','keywords']]
print(movies_metadata_new.shape)
```

```
(45463, 5)
```

READING THE DATASETS

In [15]:
```
credits.head()
```

Out[15]:

	id	cast	crew
0	862	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...
1	8844	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...
2	15602	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...
3	31357	[{'cast_id': 1, 'character': "Savannah 'Vannah...	[{'credit_id': '52fe44779251416c91011acb', 'de...
4	11862	[{'cast_id': 1, 'character': 'George Banks', '...	[{'credit_id': '52fe44959251416c75039ed7', 'de...

In [16]:

```
keywords.head()
```

Out[16]:

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...
2	15602	[{'id': 1495, 'name': 'fishing'}, {'id': 12392...
3	31357	[{'id': 818, 'name': 'based on novel'}, {'id':...
4	11862	[{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n...

In [18]:

```
movies_metadata_new.head()
```

Out[18]:

	id	genres	original_title	overview	production_companies
0	862	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	Toy Story	Led by Woody, Andy's toys live happily in his ...	[{'name': 'Pixar Animation Studios', 'id': 3}]
1	8844	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	Jumanji	When siblings Judy and Peter discover an encha...	[{'name': 'TriStar Pictures', 'id': 559}, {'na...
2	15602	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	Grumpier Old Men	A family wedding reignites the ancient feud be...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...
3	31357	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	[{'name': 'Twentieth Century Fox Film Corporat...
4	11862	[{'id': 35, 'name': 'Comedy'}]	Father of the Bride Part II	Just when George Banks has recovered from his ...	[{'name': 'Sandollar Productions', 'id': 5842}...

DATASET CLEANING AND PREPARATION

In [7]:

```
movies_metadata_new.isnull().sum()       #checking for null values
credits.isnull().sum()
```

Out[7]:

```
cast    0
crew    0
id      0
dtype: int64
```

In [8]:

```
keywords.isnull().sum()                 #checking for null values
```

Out[8]:

```
id          0
keywords    0
dtype: int64
```

In [20]:

```
movies_metadata_new['id'] = movies_metadata_new['id'].fillna(0)
movies_metadata_new['id']=pd.to_numeric(movies_metadata_new['id']).astype('Int64')

#merging two datasets
merged_data = keywords.merge(movies_metadata_new, on = 'id')
merged_data = merged_data.merge(credits, on = 'id')
```

In [21]:

```
merged_data.iloc[0].genres          #genres of a movie
```

Out[21]:

```
"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name': 'Family'}]"
```

In [22]:

```
merged_data.iloc[0].keywords       #keywords of a movie
```

Out[22]:

```
"[{'id': 931, 'name': 'jealousy'}, {'id': 4290, 'name': 'toy'}, {'id': 5202, 'name': 'boy'}, {'id': 6054, 'name': 'friendshi
p'}, {'id': 9713, 'name': 'friends'}, {'id': 9823, 'name': 'rivalry'}, {'id': 165503, 'name': 'boy next door'}, {'id': 17072
2, 'name': 'new toy'}, {'id': 187065, 'name': 'toy comes to life'}]"
```

In [24]:

```
import ast
```

In [25]:

```
def toList(text):
    list = []
    for i in ast.literal_eval(text):            #to convert the string object into a list
            list.append(i['name'])
    return list
```

```
In [26]: merged_data['genres'] = merged_data['genres'].apply(toList)
         merged_data['keywords'] = merged_data['keywords'].apply(toList)
```

```
In [27]: merged_data.head()
```

Out[27]:

	id	keywords	genres	original_title	overview	production_companies	cast	crew
0	862	[jealousy, toy, boy, friendship, friends, riva...	[Animation, Comedy, Family]	Toy Story	Led by Woody, Andy's toys live happily in his ...	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...
1	8844	[board game, disappearance, based on children'...	[Adventure, Fantasy, Family]	Jumanji	When siblings Judy and Peter discover an encha...	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...
2	15602	[fishing, best friend, duringcreditsstinger, o...	[Romance, Comedy]	Grumpier Old Men	A family wedding reignites the ancient feud be...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...
3	31357	[based on novel, interracial relationship, sin...	[Comedy, Drama, Romance]	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	[{'name': 'Twentieth Century Fox Film Corporat...	[{'cast_id': 1, 'character': "Savannah 'Vannah...	[{'credit_id': '52fe44779251416c91011acb', 'de...
4	11862	[baby, midlife crisis, confidence, aging, daug...	[Comedy]	Father of the Bride Part II	Just when George Banks has recovered from his ...	[{'name': 'Sandollar Productions', 'id': 5842}...	[{'cast_id': 1, 'character': 'George Banks', '...	[{'credit_id': '52fe44959251416c75039ed7', 'de...

```
In [28]: #getting director
         def getDir(obj):
             list = []
             for i in ast.literal_eval(obj):              #to convert the string object into a list
                 if i['job'] == 'Director':
                     list.append(i['name'])
                     break;
             return list
```

```
In [29]: merged_data['crew'] = merged_data['crew'].apply(getDir)
```

```
In [30]: merged_data.head()             #corrected format of keywords and genres
```

Out[30]:

	id	keywords	genres	original_title	overview	production_companies	cast	crew
0	862	[jealousy, toy, boy, friendship, friends, riva...	[Animation, Comedy, Family]	Toy Story	Led by Woody, Andy's toys live happily in his ...	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'cast_id': 14, 'character': 'Woody (voice)',...	[John Lasseter]
1	8844	[board game, disappearance, based on children'...	[Adventure, Fantasy, Family]	Jumanji	When siblings Judy and Peter discover an encha...	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[Joe Johnston]
2	15602	[fishing, best friend, duringcreditsstinger, o...	[Romance, Comedy]	Grumpier Old Men	A family wedding reignites the ancient feud be...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[Howard Deutch]
3	31357	[based on novel, interracial relationship, sin...	[Comedy, Drama, Romance]	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	[{'name': 'Twentieth Century Fox Film Corporat...	[{'cast_id': 1, 'character': "Savannah 'Vannah...	[Forest Whitaker]
4	11862	[baby, midlife crisis, confidence, aging, daug...	[Comedy]	Father of the Bride Part II	Just when George Banks has recovered from his ...	[{'name': 'Sandollar Productions', 'id': 5842}...	[{'cast_id': 1, 'character': 'George Banks', '...	[Charles Shyer]

```
In [31]: #getting top 5 actors
         def castNames(text):
             list = []
             counter = 0
```

```python
    for i in ast.literal_eval(text):          #to convert the string object into a list
        if(counter<5):
            list.append(i['name'])
            counter+=1
        else:
            break
    return list
```

In [32]:
```python
merged_data['cast'] = merged_data['cast'].apply(castNames)
```

In [33]:
```python
merged_data.head()
```

Out[33]:

	id	keywords	genres	original_title	overview	production_companies	cast	crew
0	862	[jealousy, toy, boy, friendship, friends, riva...	[Animation, Comedy, Family]	Toy Story	Led by Woody, Andy's toys live happily in his ...	[{'name': 'Pixar Animation Studios', 'id': 3}]	[Tom Hanks, Tim Allen, Don Rickles, Jim Varney...	[John Lasseter]
1	8844	[board game, disappearance, based on children'...	[Adventure, Fantasy, Family]	Jumanji	When siblings Judy and Peter discover an encha...	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[Robin Williams, Jonathan Hyde, Kirsten Dunst,...	[Joe Johnston]
2	15602	[fishing, best friend, duringcreditsstinger, o...	[Romance, Comedy]	Grumpier Old Men	A family wedding reignites the ancient feud be...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[Walter Matthau, Jack Lemmon, Ann-Margret, Sop...	[Howard Deutch]
3	31357	[based on novel, interracial relationship, sin...	[Comedy, Drama, Romance]	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	[{'name': 'Twentieth Century Fox Film Corporat...	[Whitney Houston, Angela Bassett, Loretta Devi...	[Forest Whitaker]
4	11862	[baby, midlife crisis, confidence, aging, daug...	[Comedy]	Father of the Bride Part II	Just when George Banks has recovered from his ...	[{'name': 'Sandollar Productions', 'id': 5842}...	[Steve Martin, Diane Keaton, Martin Short, Kim...	[Charles Shyer]

In [17]:
```python
merged_data['overview'] = merged_data['overview'].astype(str)
merged_data['overview'] = merged_data['overview'].apply(lambda x : x.split())
```

In [18]:
```python
def collapse(list):
    collapsed_list = []
    for i in list:
        collapsed_list.append(i.replace(" ",""))
    return collapsed_list
```

In [19]:
```python
merged_data['keywords'] = merged_data['keywords'].apply(collapse)
merged_data['genres'] = merged_data['genres'].apply(collapse)
merged_data['crew'] = merged_data['crew'].apply(collapse)
merged_data['cast'] = merged_data['cast'].apply(collapse)
```

In [20]:
```python
merged_data.head()
```

Out[20]:

	id	keywords	genres	original_title	overview	production_companies	cast	crew
0	862	[jealousy, toy, boy, friendship, friends, riva...	[Animation, Comedy, Family]	Toy Story	[Led, by, Woody,, Andy's, toys, live, happily,...	[{'name': 'Pixar Animation Studios', 'id': 3}]	[TomHanks, TimAllen, DonRickles, JimVarney, Wa...	[JohnLasseter]
1	8844	[boardgame, disappearance, basedonchildren'sbo...	[Adventure, Fantasy, Family]	Jumanji	[When, siblings, Judy, and, Peter, discover, a...	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[RobinWilliams, JonathanHyde, KirstenDunst, Br...	[JoeJohnston]
2	15602	[fishing, bestfriend, duringcreditsstinger, ol...	[Romance, Comedy]	Grumpier Old Men	[A, family, wedding, reignites, the, ancient, ...	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[WalterMatthau, JackLemmon, Ann-Margret, Sophi...	[HowardDeutch]
3	31357	[basedonnovel, interracialrelationship, single...	[Comedy, Drama, Romance]	Waiting to Exhale	[Cheated, on,, mistreated, and, stepped, on,, ...	[{'name': 'Twentieth Century Fox Film Corporat...	[WhitneyHouston, AngelaBassett, LorettaDevine,...	[ForestWhitaker]
4	11862	[baby, midlifecrisis, confidence, aging, daugh...	[Comedy]	Father of the Bride Part II	[Just, when, George, Banks, has, recovered, fr...	[{'name': 'Sandollar Productions', 'id': 5842}...	[SteveMartin, DianeKeaton, MartinShort, Kimber...	[CharlesShyer]

In [21]:
```
merged_data['collection'] = merged_data['keywords'] + merged_data['genres'] + merged_data['overview'] + merged_data['cast'] +
```

In [64]:
```
new_data = merged_data[['id','original_title', 'collection']].head(5000)
new_data
```

Out[64]:

	id	original_title	collection
0	862	Toy Story	[jealousy, toy, boy, friendship, friends, riva...
1	8844	Jumanji	[boardgame, disappearance, basedonchildren'sbo...
2	15602	Grumpier Old Men	[fishing, bestfriend, duringcreditsstinger, ol...
3	31357	Waiting to Exhale	[basedonnovel, interracialrelationship, single...
4	11862	Father of the Bride Part II	[baby, midlifecrisis, confidence, aging, daugh...
...	...	...	...
4995	51044	Carmen Jones	[opera, love, desire, Drama, Music, Romance, V...
4996	29475	The Five Heartbeats	[Drama, Music, In, the, early, 1960's,, a, qui...
4997	4267	Préparez vos mouchoirs	[wifehusbandrelationship, 1970s, bisexuality, ...
4998	4031	Les Valseuses	[suicide, malenudity, femalenudity, bisexualit...
4999	47871	Honky Tonk Freeway	[Action, Comedy, Ticlaw,, a, small, town, in, ...

5000 rows × 3 columns

In [65]:
```
new_data['collection'] = new_data['collection'].apply(lambda x: " ".join(x))
```

In [66]:
```
new_data['collection'] = new_data['collection'].apply(lambda x: x.lower())
```

Final Dataset

In [67]:
```
new_data.head()
```

Out[67]:

	id	original_title	collection
0	862	Toy Story	jealousy toy boy friendship friends rivalry bo...
1	8844	Jumanji	boardgame disappearance basedonchildren'sbook ...
2	15602	Grumpier Old Men	fishing bestfriend duringcreditsstinger oldmen...
3	31357	Waiting to Exhale	basedonnovel interracialrelationship singlemot...
4	11862	Father of the Bride Part II	baby midlifecrisis confidence aging daughter m...

Data Cleaning

Stemming Words

Stemming is the process of reducing a word to its base or root form. The most common algorithm used for stemming is the Porter stemming algorithm. It removes the suffix from a word to produce the base form of the word, also known as the stem.

For example, the word "running" can be stemmed to "run", "jogging" can be stemmed to "jog", and "swimming" can be stemmed to "swim". The idea behind stemming is to reduce the number of unique words in a dataset, making it easier to analyze and process the data.

Removing Stop Words

Stop words are words that do not carry any meaning on their own. Words in english such as - 'the', 'is', 'they, 'he' etc. do not really contribute to our review analysis and hence are removed. We can remove these stop words from the text in a given corpus to clean up the data, and identify words that are more rare and potentially more relevant to what we're interested in.¶

In [68]:
```
import nltk
```

In [69]:
```
from nltk.stem.porter import PorterStemmer    #stemming
ps = PorterStemmer()
```

In [70]:
```
def stem(text):
    y = []

    for i in text.split():
        y.append(ps.stem(i))

    return " ".join(y)
```

In [71]:
```
new_data['collection'] = new_data['collection'].apply(stem)
```

Converting movies to vectors

In [72]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

In [73]:
```
vector = cv.fit_transform(new_data['collection']).toarray()
vector.shape
```

Out[73]:
```
(5000, 5000)
```

Applying Cosine Similarity

In [74]:
```
from sklearn.metrics.pairwise import cosine_similarity          #finding the cosine distance between movies
```

In [75]:
```
similarity = cosine_similarity(vector)
```

In [76]:
```
def recommend(movie):
    index = new_data[new_data['original_title'] == movie].index[0]
    distances = similarity[index]
    most_similar_movies = sorted(list(enumerate(distances)),reverse=True,key = lambda x: x[1])[1:6]

    for i in most_similar_movies:
        print(new_data.iloc[i[0]].original_title)
```

Testing the Model

In [77]:
```
recommend('Toy Story')
```

```
Toy Story 2
The Million Dollar Duck
The Adventures of Elmo in Grouchland
Brighton Beach Memoirs
Carried Away
```

Storing the model in file

In [78]:
```
import pickle
pickle.dump(new_data.to_dict(), open('final-movie-dict.pkl', 'wb'))
pickle.dump(similarity,open('final-similarity.pkl', 'wb'))
```

MOVIE REVIEW SENTIMENT ANALYSIS

Here are the steps involved in performing movie review sentiment analysis:

1. Data collection: Collect movie reviews from various datasets.
2. Data preprocessing: Clean and preprocess the data to remove stop words, punctuation, and special characters.
3. Feature extraction: Extract relevant features such as the frequency of words, word embeddings, or n-grams.
4. Algorithm selection: Choose a suitable algorithm for performing sentiment analysis, such as Naive Bayes, Support Vector Machines
5. Training and testing: Train the model on a dataset of labelled movie reviews and test its performance on a subset of the data.

CODE

Loading the dataset

In [10]:
```
import numpy as np;                  #importing required libraries
import pandas as pd;
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv("IMDB Dataset.csv")   #reading the dataset
df.head()
```

Out[3]:

	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

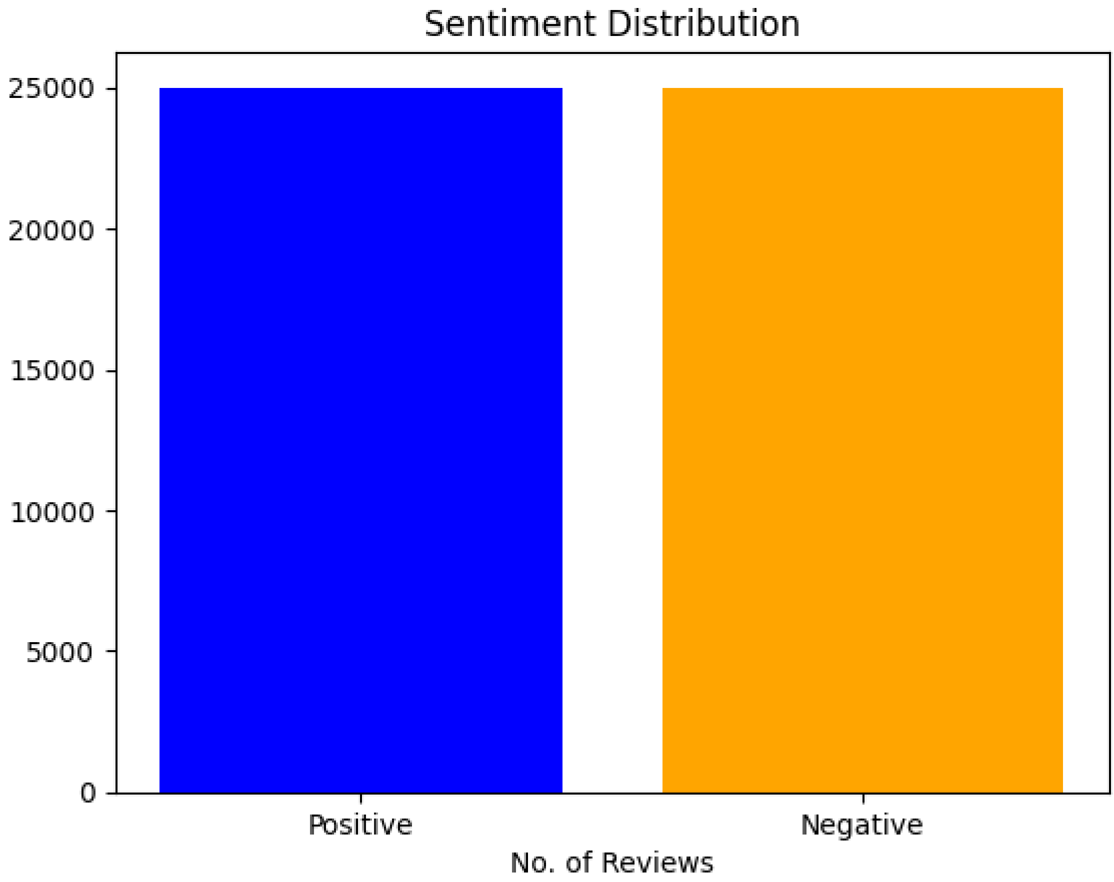
In [164…
```
df['review'][1]                          #displaying a review
```

Out[164]:
```
'A wonderful little production. <br /><br />The filming technique is very unass
uming- very old-time-BBC fashion and gives a comforting, and sometimes discomfo
rting, sense of realism to the entire piece. <br /><br />The actors are extreme
ly well chosen- Michael Sheen not only "has got all the polari" but he has all
the voices down pat too! You can truly see the seamless editing guided by the r
eferences to Williams\' diary entries, not only is it well worth the watching b
ut it is a terrificly written and performed piece. A masterful production about
one of the great master\'s of comedy and his life. <br /><br />The realism real
ly comes home with the little things: the fantasy of the guard which, rather th
an use the traditional \'dream\' techniques remains solid then disappears. It p
lays on our knowledge and our senses, particularly with the scenes concerning O
rton and Halliwell and the sets (particularly of their flat with Halliwell\'s m
urals decorating every surface) are terribly well done.'
```

In [16]:
```
pos = (df["sentiment"]=='positive').sum()     #counting no. of positive reviews
print(pos)
neg = (df["sentiment"]=='negative').sum()     #counting no. of negative reviews
print(neg)

plt.bar(["Positive","Negative"], [pos,neg], color = ["Blue", "Orange"])   #distr
plt.xlabel("Reviews")
plt.xlabel("No. of Reviews")
plt.title("Sentiment Distribution")
```

```
25000
25000
```

Out[16]: `Text(0.5, 1.0, 'Sentiment Distribution')`



Data Preprocessing / Data Cleaning

In [93]:
```
#Data Cleaning
#1) Remove html tags (<br>)
#2) Remove special characters
#3) Convert to lower case
#4) Remove stop words
#5) Stemming
```

In [166…
```
#converting catergorical data to numeric data
df['sentiment'].replace({'positive':1, 'negative':0}, inplace=True)
```

In [167…
```
df.head()
```

Out[167]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. 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

Removing HTML tags

In [168…
```python
import re
clean = re.compile('<br\s*/?>')
re.sub(clean,'',df.iloc[0].review)


# <br: matches the literal characters "<br".
# \s*: matches zero or more whitespace characters (such as spaces or tabs).
# /: matches the forward slash character.
# ?: makes the preceding character (the forward slash) optional, so that the pat
# >: matches the closing angle bracket character.
# So, the entire pattern matches "<br>", "<br />", or "<br/>" (with any amount c
```

Out[168]:
```
"One of the other reviewers has mentioned that after watching just 1 Oz episode
you'll be hooked. They are right, as this is exactly what happened with me.The
first thing that struck me about Oz was its brutality and unflinching scenes of
violence, which set in right from the word GO. Trust me, this is not a show for
the faint hearted or timid. This show pulls no punches with regards to drugs, s
ex or violence. Its is hardcore, in the classic use of the word.It is called OZ
as that is the nickname given to the Oswald Maximum Security State Penitentary.
It focuses mainly on Emerald City, an experimental section of the prison where
all the cells have glass fronts and face inwards, so privacy is not high on the
agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christian
s, Italians, Irish and more....so scuffles, death stares, dodgy dealings and sh
ady agreements are never far away.I would say the main appeal of the show is du
e to the fact that it goes where other shows wouldn't dare. Forget pretty pictu
res painted for mainstream audiences, forget charm, forget romance...OZ doesn't
mess around. The first episode I ever saw struck me as so nasty it was surreal,
I couldn't say I was ready for it, but as I watched more, I developed a taste f
or Oz, and got accustomed to the high levels of graphic violence. Not just viol
ence, but injustice (crooked guards who'll be sold out for a nickel, inmates wh
o'll kill on order and get away with it, well mannered, middle class inmates be
ing turned into prison bitches due to their lack of street skills or prison exp
erience) Watching Oz, you may become comfortable with what is uncomfortable vie
wing....thats if you can get in touch with your darker side."
```

In [169…
```python
#function to remove html tags

def clean_html(text):
    clean = re.compile('<br\s*/?>')
    return re.sub(clean, '', text)
```

In [170…
```python
df['review'] =df['review'].apply(clean_html)
```

In [171…
```python
df.head()
```

Out[171]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming tec...	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

In [172...
```
df['review'][3]   #all html tags are removed
```

Out[172]:
```
"Basically there's a family where a little boy (Jake) thinks there's a zombie i
n his closet & his parents are fighting all the time.This movie is slower than
a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.
OK, first of all when you're going to make a film you must Decide if its a thri
ller or a drama! As a drama the movie is watchable. Parents are divorcing & arg
uing like in real life. And then we have Jake with his closet which totally rui
ns all the film! I expected to see a BOOGEYMAN similar movie, and instead i wat
ched a drama with some meaningless thriller spots.3 out of 10 just for the well
playing parents & descent dialogs. As for the shots with Jake: just ignore the
m."
```

Converting to lower case

In [173...
```
#method for converting to lower case
def convert_lower(text):
    return text.lower()
```

In [12]:
```
df['review'] = df['review'].apply(convert_lower)
```

In [13]:
```
df['review'][2]  #converted to lower case
```

Out[13]:
```
'i thought this was a wonderful way to spend time on a too hot summer weekend,
sitting in the air conditioned theater and watching a light-hearted comedy. the
plot is simplistic, but the dialogue is witty and the characters are likable (e
ven the well bread suspected serial killer). while some may be disappointed whe
n they realize this is not match point 2: risk addiction, i thought it was proo
f that woody allen is still fully in control of the style many of us have grown
to love.this was the most i\'d laughed at one of woody\'s comedies in years (da
re i say a decade?). while i\'ve never been impressed with scarlet johanson, in
this she managed to tone down her "sexy" image and jumped right into a average,
but spirited young woman.this may not be the crown jewel of his career, but it
was wittier than "devil wears prada" and more interesting than "superman" a gre
at comedy to go see with friends.'
```

Removing Special Characters

In [174...
```
#function to remove special characters

def remove_special(text):
    x=''
    for i in text:
        if i.isalnum():
            x=x+i
        else:
            x=x+' '
    return x
```

In [15]:
```
remove_special(df['review'][0])
```

Out[15]:
```
'one of the other reviewers has mentioned that after watching just 1 oz episode
you ll be hooked  they are right  as this is exactly what happened with me the
first thing that struck me about oz was its brutality and unflinching scenes of
violence  which set in right from the word go  trust me  this is not a show for
the faint hearted or timid  this show pulls no punches with regards to drugs  s
ex or violence  its is hardcore  in the classic use of the word it is called oz
as that is the nickname given to the oswald maximum security state penitentary
it focuses mainly on emerald city  an experimental section of the prison where
all the cells have glass fronts and face inwards  so privacy is not high on the
agenda  em city is home to many  aryans  muslims  gangstas  latinos  christians
italians  irish and more    so scuffles  death stares  dodgy dealings and shady
agreements are never far away i would say the main appeal of the show is due to
the fact that it goes where other shows wouldn t dare  forget pretty pictures p
ainted for mainstream audiences  forget charm  forget romance   oz doesn t mess
around  the first episode i ever saw struck me as so nasty it was surreal  i co
uldn t say i was ready for it  but as i watched more  i developed a taste for o
z  and got accustomed to the high levels of graphic violence  not just violence
but injustice  crooked guards who ll be sold out for a nickel  inmates who ll k
ill on order and get away with it  well mannered  middle class inmates being tu
rned into prison bitches due to their lack of street skills or prison experienc
e  watching oz  you may become comfortable with what is uncomfortable viewing
thats if you can get in touch with your darker side '
```

In [16]:
```
df['review']=df['review'].apply(remove_special)
```

In [17]:
```
df.head()
```

Out[17]:

	review	sentiment
0	one of the other reviewers has mentioned that ...	1
1	a wonderful little production the filming tec...	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

Removing Stop Words

Stop words are words that do not carry any meaning on their own. Words in english such as - 'the', 'is', 'they, 'he' etc. do not really contribute to our review analysis and hence are removed. We can remove these stop words from the text in a given corpus to clean up the data, and identify words that are more rare and potentially more relevant to what we're interested in.

In [175…
```
#removing stop words

import nltk
nltk.download('stopwords')
# from nltk.corpus import stopwords
# stopwords.words('english')
```

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

Out[175]: True

In [319…

```
from nltk.corpus import stopwords
#stopwords.words('english')
```

In [200…

```
#function to remove stopwords

def remove_stopwords(text):
    x=[]
    for i in text.split():
        if i not in stopwords.words('english'):
            x.append(i)
    z=x[:]
    x.clear()
    return z
```

In [21]:

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

In [23]:

```
df.head()
```

Out[23]:

	review	sentiment
0	[one, reviewers, mentioned, watching, 1, oz, e...	1
1	[wonderful, little, production, filming, techn...	1
2	[thought, wonderful, way, spend, time, hot, su...	1
3	[basically, family, little, boy, jake, thinks,...	0
4	[petter, mattei, love, time, money, visually, ...	1

Stemming Words

Stemming is the process of reducing a word to its base or root form. The most common algorithm used for stemming is the Porter stemming algorithm. It removes the suffix from a word to produce the base form of the word, also known as the stem.

For example, the word "running" can be stemmed to "run", "jogging" can be stemmed to "jog", and "swimming" can be stemmed to "swim". The idea behind stemming is to reduce the number of unique words in a dataset, making it easier to analyze and process the data.

In [208…

```
#perform stemming
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
p=[]
def stem_words(text):

    for i in text:
        p.append(ps.stem(i))
    z=p[:]
    p.clear()
    return z
```

In [25]:

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

In [26]:

```
df.head()
```

Out[26]:

	review	sentiment
0	[one, review, mention, watch, 1, oz, episod, h...	1
1	[wonder, littl, product, film, techniqu, unass...	1
2	[thought, wonder, way, spend, time, hot, summe...	1
3	[basic, famili, littl, boy, jake, think, zombi...	0
4	[petter, mattei, love, time, money, visual, st...	1

In [179…

```
#join back
def join_back(list_input):
    return " ".join(list_input)

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

In [28]:

```
df.head()
```

Out[28]:

	review	sentiment
0	one review mention watch 1 oz episod hook righ...	1
1	wonder littl product film techniqu unassum old...	1
2	thought wonder way spend time hot summer weeke...	1
3	basic famili littl boy jake think zombi closet...	0
4	petter mattei love time money visual stun film...	1

In [33]:

```
#storing the cleaned data into a new csv file
df.to_csv('cleanReviews.csv', index=False)  #storing data after preprocessing ar
```

In [244…

```
new_df = pd.read_csv('cleanReviews.csv')
new_df.head(5)
```

Out[244]:

	review	sentiment
0	one review mention watch 1 oz episod hook righ...	1
1	wonder littl product film techniqu unassum old...	1
2	thought wonder way spend time hot summer weeke...	1
3	basic famili littl boy jake think zombi closet...	0
4	petter mattei love time money visual stun film...	1

Vectorizing

CountVectorizer is a preprocessing step used in natural language processing (NLP) that converts a collection of text documents to a matrix of token counts. It is a process of converting text data into a matrix of token counts. In other words, it is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.

In [245…
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=6000)    #6000 most frequent words

X = cv.fit_transform(new_df['review']).toarray()
```

In [246…
```
X
```

Out[246]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [247…
```
y=new_df.iloc[:,-1].values
print(y)
```

```
[1 1 1 ... 0 0 0]
```

Building and Training the Model

Splitting data into test and training data

In [248…
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [249…
```
print(len(x_train))
print(len(x_test))
x_train.shape
```

```
40000
10000
```

Out[249]:
```
(40000, 6000)
```

In [250…
```
x_test
```

Out[250]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 3, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0]], dtype=int64)
```

Applying Naive Bayes Algorithm

In [251…
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
clf1=GaussianNB()
clf2 = MultinomialNB()
clf3 = BernoulliNB()
```

In [252…
```
clf1.fit(x_train, y_train)              #fitting the data into the model
clf2.fit(x_train, y_train)
clf3.fit(x_train, y_train)
```

Out[252]:

```
▾ BernoulliNB
BernoulliNB()
```

In [253…

```
y_pred1 = clf1.predict(x_test)       #predicting on test data
y_pred2 = clf2.predict(x_test)
y_pred3 = clf3.predict(x_test)
```

Comparing accuracies of three Naive Bayes Models

1. Guasssian NB

2. Mulinomial NB

3. Bernaulli NB

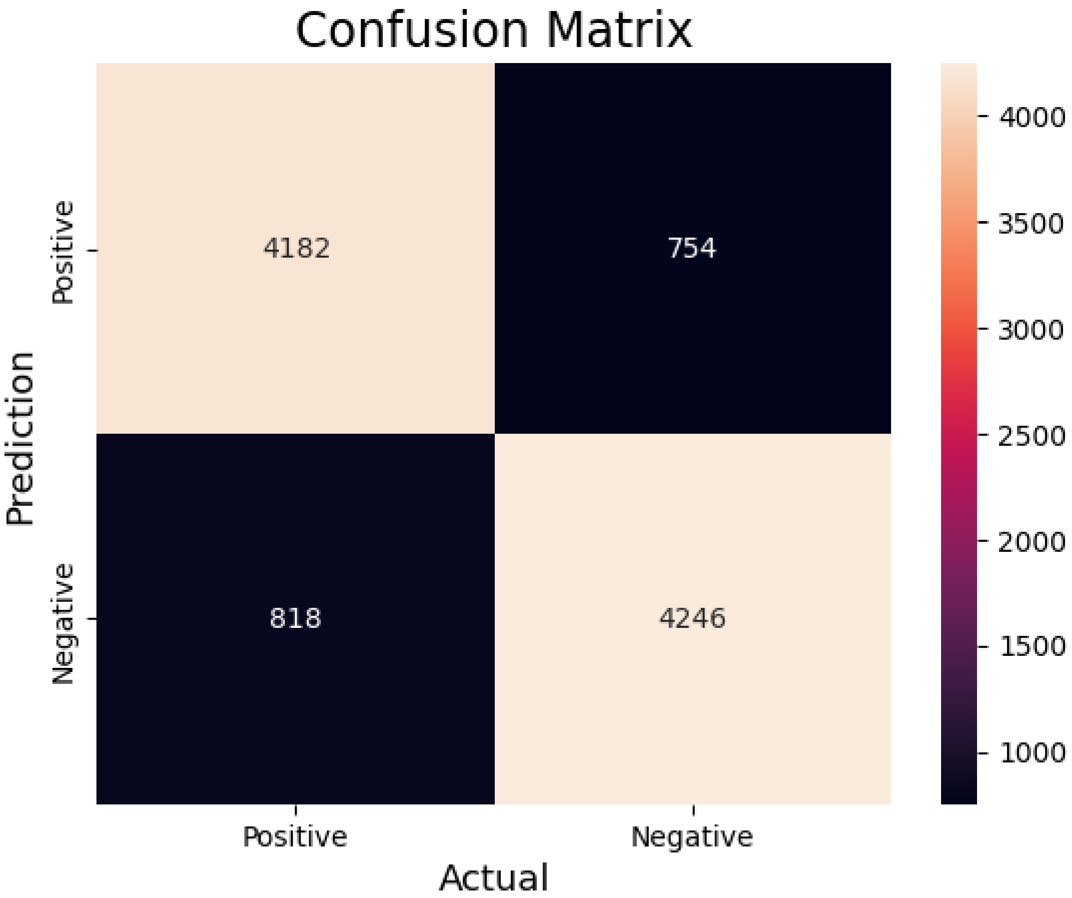
In [315…

```
from sklearn.metrics import accuracy_score

print('Gaussian:\t', accuracy_score(y_test, y_pred1)*100,'%')
print('Multinomial:\t', accuracy_score(y_test, y_pred2)*100,'%')
print('Bernaulli:\t', accuracy_score(y_test, y_pred3)*100,'%')
```

```
Gaussian:        69.83 %
Multinomial:     84.28 %
Bernaulli:       84.50999999999999 %
```

In [316…

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test,y_pred2)

#Plot the confusion matrix.
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Positive','Negative'],
            yticklabels=['Positive','Negative'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



In [293…

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred2))
```

```
              precision    recall  f1-score   support

           0       0.84      0.85      0.84      4936
           1       0.85      0.84      0.84      5064

    accuracy                           0.84     10000
   macro avg       0.84      0.84      0.84     10000
weighted avg       0.84      0.84      0.84     10000
```

Storing the Model

By using pickle.dump(), the models can be saved to files and loaded back into memory at a later time for prediction on new data.

In [255…

```
import pickle
pickle.dump(cv, open("count-Vectorizer.pkl", "wb"))
pickle.dump(clf2, open("movie_review_sentiment.pkl", "wb"))
```

In [256…

```
save_cv=pickle.load(open('count-Vectorizer.pkl', 'rb'))
model = pickle.load(open('movie_review_sentiment.pkl', 'rb'))
```

In [267…

```
userReviews = []
```

In [268…

```
def test_model(text):
    sen = save_cv.transform([text]).toarray()
    res = clf2.predict(sen)[0]
```

```python
    userReviews.append(res)
    if res==1:
        return 'Positive Review'
    else:
        return 'Negative Rewiew'
```

In [299…

```python
sen = "I could think of better ways to spend time"    #True Sentiment:      Nega
res = test_model(sen)                                  #Predicted Sentiment: Nega
print(res)
```

```
Negative Rewiew
```

In [270…

```python
sen = "It was so emotional, I cried."                 #True Sentiment:      Posi
res = test_model(sen)                                  #Predicted Sentiment: Nega
print(res)
```

```
Negative Rewiew
```

In [300…

```python
sen = "It really touched my heart."                     #True Sentiment:      Pos
res = test_model(sen)                                   #Predicted Sentiment: Pos
print(res)
```

```
Positive Review
```

In [314…

```python
sen = "It was so funny! I enjoyed it!"                  #True Sentiment:      Pos
res = test_model(sen)                                   #Predicted Sentiment: Neg
print(res)
```

```
Negative Rewiew
```

Applying Linear Support Vector Classifier Model

In [32]:
```
import numpy as np;            #importing required libraries
import pandas as pd;

df = pd.read_csv("cleanReviews.csv")      #reading the clean csv
df.head()
```

Out[32]:

	review	sentiment
0	one review mention watch 1 oz episod hook righ...	1
1	wonder littl product film techniqu unassum old...	1
2	thought wonder way spend time hot summer weeke...	1
3	basic famili littl boy jake think zombi closet...	0
4	petter mattei love time money visual stun film...	1

In [33]:
```
df.shape
```

Out[33]: (50000, 2)

In [7]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
```

In [5]:
```
df.head()
```

Out[5]:

	review	sentiment
0	one review mention watch 1 oz episod hook righ...	1
1	wonder littl product film techniqu unassum old...	1
2	thought wonder way spend time hot summer weeke...	1
3	basic famili littl boy jake think zombi closet...	0
4	petter mattei love time money visual stun film...	1

Vectorizing

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical technique used to evaluate the importance of a word in a document in a collection or corpus of documents.

In [34]:
```
tfidf = TfidfVectorizer(max_features = 3000)
X=df['review']
y = df['sentiment']
X = tfidf.fit_transform(X).toarray()

X
```

```
Out[34]: array([[0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.]])
```

```
In [35]: y=df.iloc[:,-1].values
         print(y)
```

```
[1 1 1 ... 0 0 0]
```

Building and Training the Model

Splitting the data into testing (20%) and training data (80%)

```
In [36]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Applying Linear LVC Algorithm

```
In [37]: clf = LinearSVC()
         clf.fit(X_train, y_train)
```

```
Out[37]: ▾ LinearSVC
         LinearSVC()
```

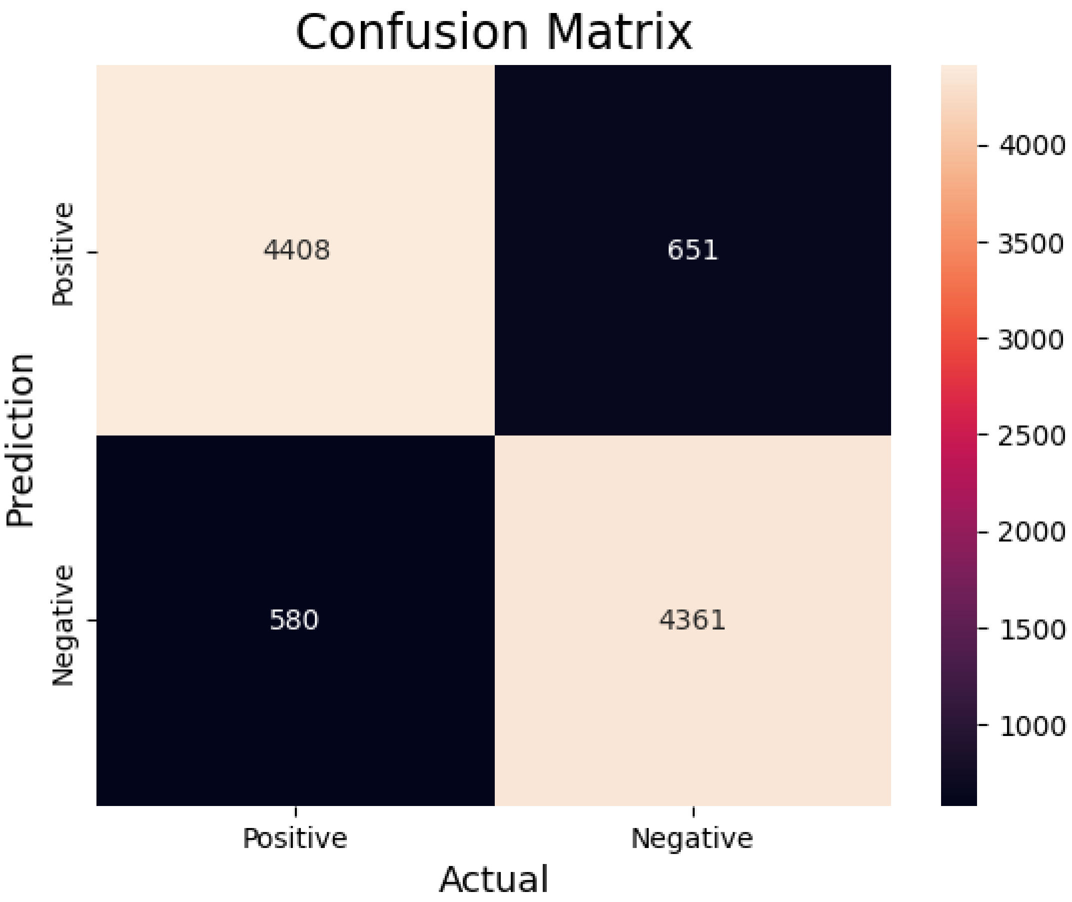
```
In [38]: prediction = clf.predict(X_test)
```

```
In [39]: print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.88      0.87      0.88      5059
           1       0.87      0.88      0.88      4941

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

```
In [49]: import matplotlib.pyplot as plt                        #plotting the confus
         from sklearn.metrics import confusion_matrix
         import seaborn as sns

         cm = confusion_matrix(y_test,prediction)

         #Plot the confusion matrix.
         sns.heatmap(cm,
                     annot=True,
                     fmt='g',
                     xticklabels=['Positive','Negative'],
                     yticklabels=['Positive','Negative'])
         plt.ylabel('Prediction',fontsize=13)
```

```
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



In [40]:
```
import re
def clean_html(text):
    clean = re.compile('<br\s*/?>')
    return re.sub(clean, '', text)

#converting to lower case
def convert_lower(text):
    return text.lower()

def remove_special(text):
    x=''
    for i in text:
        if i.isalnum():
            x=x+i
        else:
            x=x+' '
    return x

import nltk
from nltk.corpus import stopwords
def remove_stopwords(text):
    x=[]
    for i in text.split():
        if i not in stopwords.words('english'):
            x.append(i)
    y=x[:]
    x.clear()
    return y
```

```
#perform stemming
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
p=[]
def stem_words(text):

    for i in text:
        p.append(ps.stem(i))
    z=p[:]
    p.clear()
    return z

def join_back(list_input):
    return " ".join(list_input)
```

In [41]:
```
def clean(text):
    text = clean_html(text)
    text = convert_lower(text)
    text = remove_special(text)
    text = remove_stopwords(text)
    text = stem_words(text)
    text = join_back(text)
    return text
```

Testing the Model

In [55]:
```
def predictor(text):                        #method to clean the review and pass it
    text = clean(text)
    vec = tfidf.transform([text])
    res = clf.predict(vec)
    if res == 1:
        print("Positive Review")
    else:
        print("Negative Review")
```

In [56]:
```
predictor("I could think of better ways to spend time")
```

```
Negative Review
```

In [57]:
```
predictor("It was so emotional, I cried.")
```

```
Positive Review
```

In [58]:
```
predictor("It really touched my heart")
```

```
Positive Review
```

In [59]:
```
predictor("It was so funny! I enjoyed it!")
```

```
Positive Review
```

In [68]:
```
predictor("This movie is a masterpiece! The acting, cinematography, and storylin
```

```
Positive Review
```

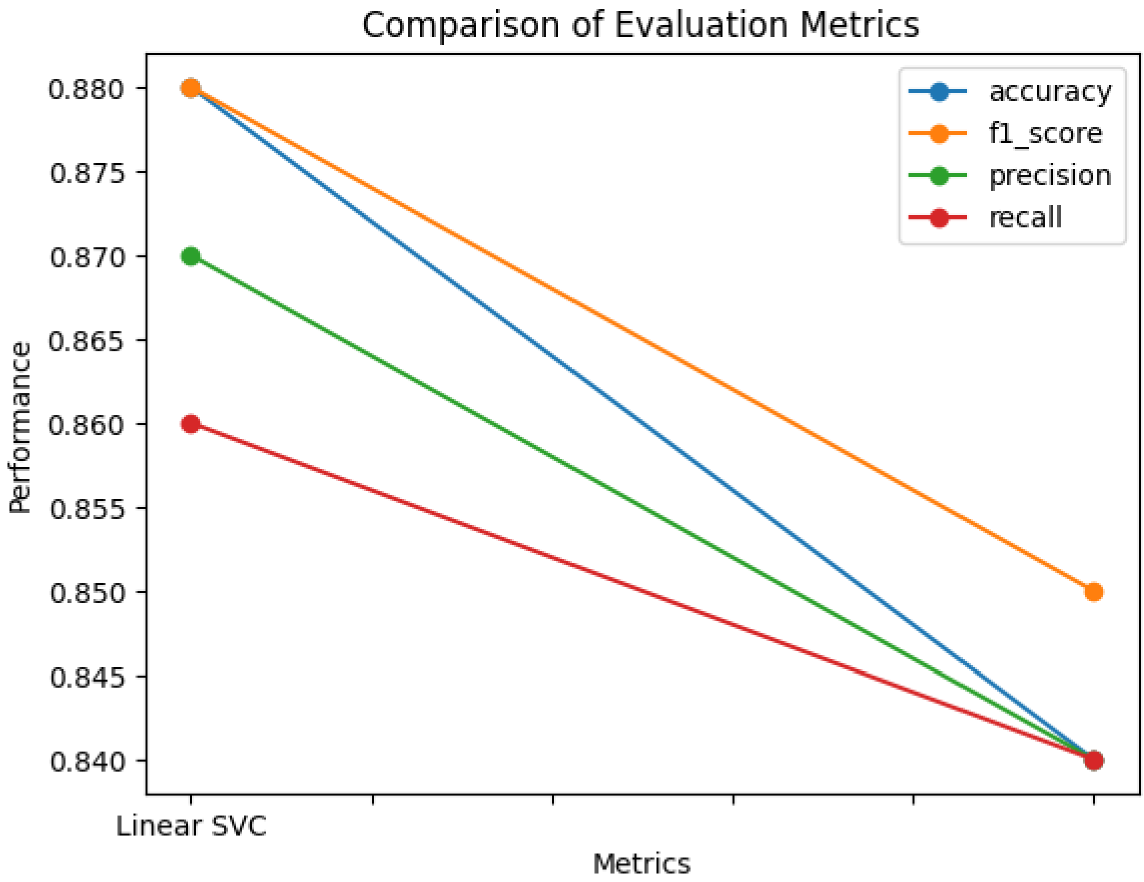
Storing the Model

By using pickle.dump(), the models can be saved to files and loaded back into memory at a later time for prediction on new data.

In [60]:
```
import pickle
pickle.dump(tfidf, open("tfid_Vectorizer.pkl", "wb"))
pickle.dump(clf, open("LinearSVM_review_sentiment.pkl", "wb"))
```

Comparing both Models

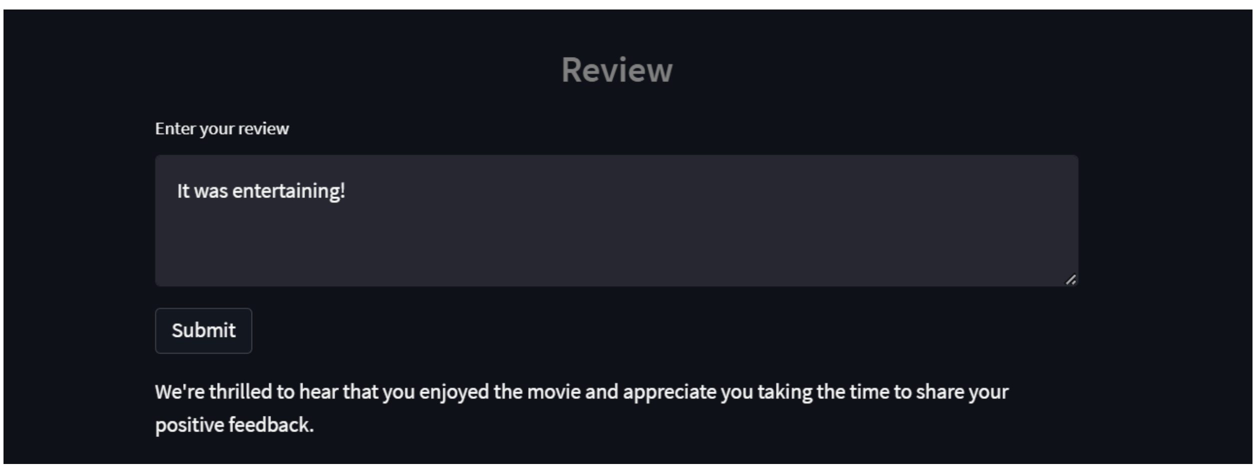
In [87]:
```
import pandas as pd
import matplotlib.pyplot as plt

# create a pandas dataframe with evaluation metrics of both models
data = {'accuracy': [0.88, 0.84],
        'f1_score': [0.88, 0.85],
        'precision': [0.87, 0.84],
        'recall': [0.86, 0.84]}
df = pd.DataFrame(data, index=['Linear SVC', 'Multinomial Naive Bayes'])

# create a line graph
df.plot(kind='line', marker='o', label='Multinomial Naive Bayes')

# add title, axis labels, and legend
plt.title('Comparison of Evaluation Metrics')
plt.xlabel('Metrics')
plt.ylabel('Performance')
plt.legend()
plt.xticks(rotation=0)

# display the line graph
plt.show()
print("\t\t\t\t\t\t\tMultinomial Naive Bayes")
```

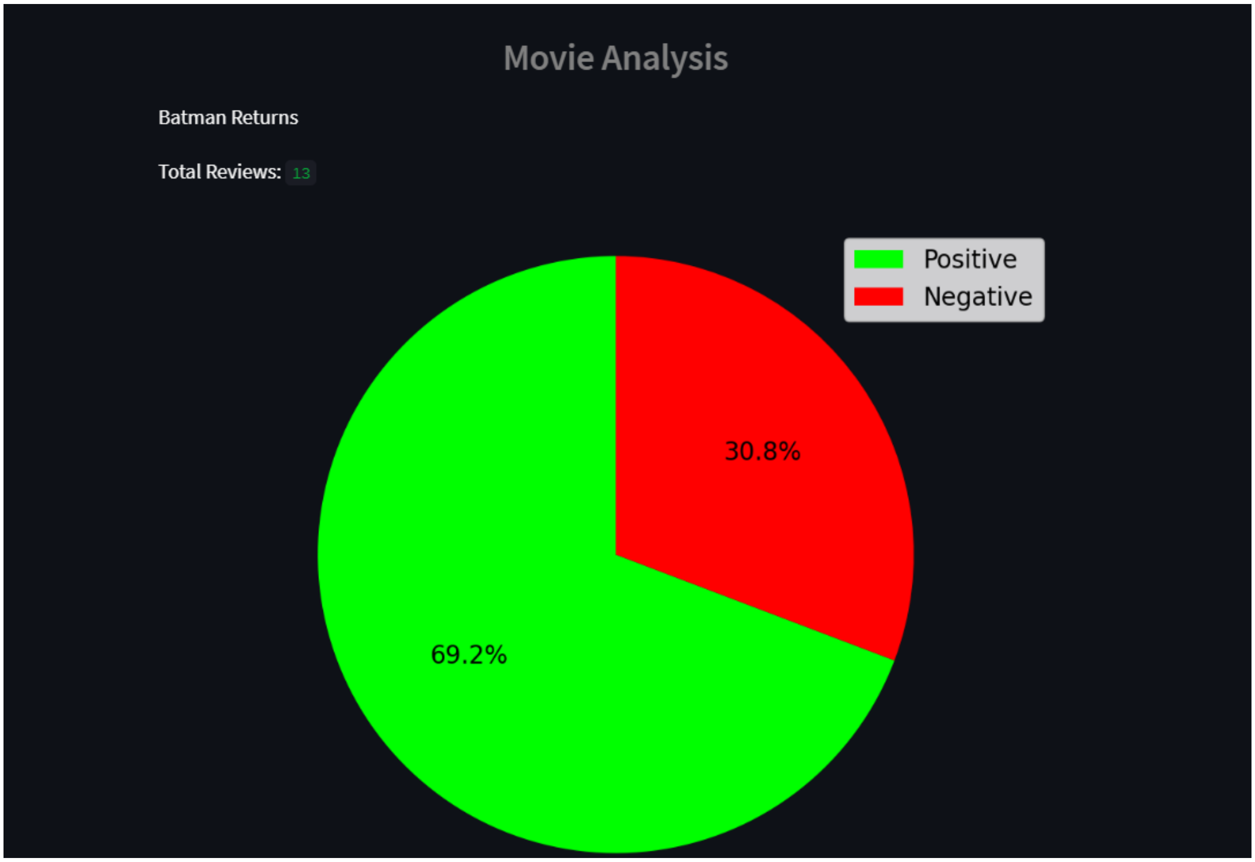


Multinomial Naive Bayes

OUTPUT:


Movie Analyser
Movie Recommender
What would you wanna watch?
Batman Returns
Recommend
Batman: Mask of the Phantasm
Batman
Batman Forever
Batman & Robin
What's the Worst That Could Happen?


Review
Enter your review
It was entertaining!
Submit
We're thrilled to hear that you enjoyed the movie and appreciate you taking the time to share your positive feedback.



CONCLUSION

In conclusion, a movie recommendation system and movie review sentiment analysis can greatly benefit a movie streaming platform by providing personalized movie recommendations to users and analyzing their feedback.
The recommendation system can be built using machine learning algorithms to analyze user data and generate personalized movie suggestions based on their viewing history, ratings, and preferences. Additionally, the sentiment analysis of movie reviews can help the platform understand user feedback and improve their movie recommendations further.
Deploying the recommendation system and sentiment analysis as a web application or API will provide a seamless user experience, and evaluating the performance of the system using metrics such as precision, recall, and F1 score will ensure that the platform is meeting its objectives. By implementing these techniques, the platform can increase user engagement, customer satisfaction, and ultimately drive business growth.

REFERENCES:

1) Kaggle
2) Google
3) YouTube

****THANK-YOU****