In [12]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
from bs4 import BeautifulSoup
import spacy
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from textblob import TextBlob
from textblob import Word
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

import os
import warnings
```

In [9]:
```
imdb_data=pd.read_csv('IMDB Dataset.csv')
print(imdb_data.shape)
imdb_data.head(10)
```

```
(50000, 2)
```

Out[9]:

	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
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

In [13]:
```
#Summary of the dataset
imdb_data.describe()
```

Out[13]:

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

In [14]:
```
#sentiment count
imdb_data['sentiment'].value_counts()
```

Out[14]:
```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

In [15]:
```
#split the dataset
#train dataset
train_reviews=imdb_data.review[:40000]
train_sentiments=imdb_data.sentiment[:40000]
#test dataset
test_reviews=imdb_data.review[40000:]
test_sentiments=imdb_data.sentiment[40000:]
print(train_reviews.shape,train_sentiments.shape)
print(test_reviews.shape,test_sentiments.shape)
```

```
(40000,) (40000,)
(10000,) (10000,)
```

In [18]:
```
import nltk
nltk.download('stopwords')
#Tokenization of text
tokenizer=ToktokTokenizer()
#Setting English stopwords
stopword_list=nltk.corpus.stopwords.words('english')
```

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

In [19]:
```
#Removing the html strips
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

#Removing the noisy text
def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(denoise_text)
```

```
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\bs4\__in
it__.py:435: MarkupResemblesLocatorWarning: The input looks more like a filename t
han markup. You may want to open this file and pass the filehandle into Beautiful
Soup.
  warnings.warn(
```

In [20]:
```
#Define function for removing special characters
def remove_special_characters(text, remove_digits=True):
    pattern=r'[^a-zA-z0-9\s]'
    text=re.sub(pattern,'',text)
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(remove_special_characters)
```

In [23]:
```
#Stemming the text
def simple_stemmer(text):
    ps=nltk.porter.PorterStemmer()
    text= ' '.join([ps.stem(word) for word in text.split()])
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(simple_stemmer)
```

In [22]:
```
#set stopwords to english
stop=set(stopwords.words('english'))
print(stop)

#removing the stopwords
def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stopword_list
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stopw
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(remove_stopwords)
```

```
{'has', 'yours', 'herself', 'should', 'wouldn', 'that', 'who', 'while', 'again',
'how', 'out', 'now', 'having', 'hadn', 'as', 't', 'themselves', "shouldn't", "yo
u'd", 'what', 'ma', 'didn', "won't", 'all', 'can', 'been', 'of', "hadn't", 'have
n', 'himself', "she's", 'to', 'does', 'and', 'don', 'down', "it's", 'wasn', 'ourse
lves', 'our', 'just', 'y', "doesn't", 'do', 'at', "don't", 'more', 'hers', "you'v
e", 'd', 'he', 'between', 'such', 'for', 'o', 'we', 'then', 'the', 'under', "have
n't", "isn't", 'aren', 're', "shan't", 'my', 'any', 'over', 'here', 'very', 'bot
h', 'your', 'were', 'there', 'most', 'above', 'weren', "didn't", 'a', 'are', 'in',
'on', 'each', 've', 'isn', 'from', "mustn't", 'hasn', 's', 'they', "hasn't", "tha
t'll", 'being', 'other', 'when', 'is', 'off', 'be', 'or', 'too', 'some', "you'll",
'did', 'against', 'if', "aren't", 'me', 'until', 'theirs', 'than', "mightn't", "co
uldn't", 'it', 'after', 'will', 'have', 'them', 'because', 'only', 'but', "need
n't", 'yourself', 'during', 'no', 'into', 'once', 'through', 'these', 'won', 'thi
s', 'an', 'doing', 'doesn', "wouldn't", 'shan', 'not', 'mightn', 'she', 'before',
"wasn't", 'had', 'itself', 'with', 'so', 'yourselves', 'm', 'up', 'which', 'll',
'ain', 'am', 'mustn', 'ours', 'further', 'few', 'his', 'him', 'myself', 'nor', 'ow
n', 'by', "weren't", 'below', 'was', 'its', 'her', 'about', 'why', 'couldn', 'who
m', 'where', 'i', "should've", 'their', "you're", 'same', 'needn', 'you', 'those',
'shouldn'}
```

In [24]:
```
#normalized train reviews
norm_train_reviews=imdb_data.review[:40000]
norm_train_reviews[0]
#convert dataframe to string
#norm_train_string=norm_train_reviews.to_string()
#Spelling correction using Textblob
#norm_train_spelling=TextBlob(norm_train_string)
#norm_train_spelling.correct()
#Tokenization using Textblob
```

```
#norm_train_words=norm_train_spelling.words
#norm_train_words
```

Out[24]:
```
'one review ha mention watch 1 oz episod youll hook right thi exactli happen meth
first thing struck oz wa brutal unflinch scene violenc set right word go trust thi
show faint heart timid thi show pull punch regard drug sex violenc hardcor classic
use wordit call oz nicknam given oswald maximum secur state penitentari focu mainl
i emerald citi experi section prison cell glass front face inward privaci high age
nda em citi home manyaryan muslim gangsta latino christian italian irish moreso sc
uffl death stare dodgi deal shadi agreement never far awayi would say main appeal
show due fact goe show wouldnt dare forget pretti pictur paint mainstream audienc
forget charm forget romanceoz doesnt mess around first episod ever saw struck nast
i wa surreal couldnt say wa readi watch develop tast oz got accustom high level gr
aphic violenc violenc injust crook guard wholl sold nickel inmat wholl kill order
get away well manner middl class inmat turn prison bitch due lack street skill pri
son experi watch oz may becom comfort uncomfort viewingthat get touch darker side'
```

In [25]:
```
#Normalized test reviews
norm_test_reviews=imdb_data.review[40000:]
norm_test_reviews[45005]
##convert dataframe to string
#norm_test_string=norm_test_reviews.to_string()
#spelling correction using Textblob
#norm_test_spelling=TextBlob(norm_test_string)
#print(norm_test_spelling.correct())
#Tokenization using Textblob
#norm_test_words=norm_test_spelling.words
#norm_test_words
```

Out[25]:
```
'read review watch thi piec cinemat garbag took least 2 page find somebodi el didn
t think thi appallingli unfunni montag wasnt acm humour 70 ind ani era thi isnt le
ast funni set sketch comedi ive ever seen itll till come along half skit alreadi d
one infinit better act monti python woodi allen wa say nice piec anim last 90 seco
nd highlight thi film would still get close sum mindless drivelridden thi wast 75
minut semin comedi onli world semin realli doe mean semen scatolog humour onli wor
ld scat actual fece precursor joke onli mean thi handbook comedi tit bum odd beave
r niceif pubesc boy least one hand free havent found playboy exist give break beca
u wa earli 70 way sketch comedi go back least ten year prior onli way could even f
orgiv thi film even made wa gunpoint retro hardli sketch clown subtli pervert chil
dren may cut edg circl could actual funni come realli quit sad kept go throughout
entir 75 minut sheer belief may save genuin funni skit end gave film 1 becau wa lo
wer scoreand onli recommend insomniac coma patientsor perhap peopl suffer lockjawt
heir jaw would final drop open disbelief'
```

In [28]:
```
#Count vectorizer for bag of words
cv=CountVectorizer(min_df=0,max_df=1,binary=False,ngram_range=(1,3))
#transformed train reviews
cv_train_reviews=cv.fit_transform(norm_train_reviews)
#transformed test reviews
cv_test_reviews=cv.transform(norm_test_reviews)

print('BOW_cv_train:',cv_train_reviews.shape)
print('BOW_cv_test:',cv_test_reviews.shape)
#vocab=cv.get_feature_names()-toget feature names
```

```
BOW_cv_train: (40000, 6200508)
BOW_cv_test: (10000, 6200508)
```

In [29]:
```
#Tfidf vectorizer
tv=TfidfVectorizer(min_df=0,max_df=1,use_idf=True,ngram_range=(1,3))
#transformed train reviews
tv_train_reviews=tv.fit_transform(norm_train_reviews)
#transformed test reviews
tv_test_reviews=tv.transform(norm_test_reviews)
```

```
print('Tfidf_train:',tv_train_reviews.shape)
print('Tfidf_test:',tv_test_reviews.shape)
```

```
Tfidf_train: (40000, 6200508)
Tfidf_test: (10000, 6200508)
```

In [30]:
```
#Labeling the sentient data
lb=LabelBinarizer()
#transformed sentiment data
sentiment_data=lb.fit_transform(imdb_data['sentiment'])
print(sentiment_data.shape)
```

```
(50000, 1)
```

In [31]:
```
#Spliting the sentiment data
train_sentiments=sentiment_data[:40000]
test_sentiments=sentiment_data[40000:]
print(train_sentiments)
print(test_sentiments)
```

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

In [32]:
```
#training the model
lr=LogisticRegression(penalty='l2',max_iter=500,C=1,random_state=42)
#Fitting the model for Bag of words
lr_bow=lr.fit(cv_train_reviews,train_sentiments)
print(lr_bow)
#Fitting the model for tfidf features
lr_tfidf=lr.fit(tv_train_reviews,train_sentiments)
print(lr_tfidf)
```

```
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
LogisticRegression(C=1, max_iter=500, random_state=42)
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
LogisticRegression(C=1, max_iter=500, random_state=42)
```

In [33]:
```
#Predicting the model for bag of words
lr_bow_predict=lr.predict(cv_test_reviews)
print(lr_bow_predict)
##Predicting the model for tfidf features
lr_tfidf_predict=lr.predict(tv_test_reviews)
print(lr_tfidf_predict)
```

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

In [34]:
```
#Accuracy score for bag of words
lr_bow_score=accuracy_score(test_sentiments,lr_bow_predict)
print("lr_bow_score :",lr_bow_score)
#Accuracy score for tfidf features
lr_tfidf_score=accuracy_score(test_sentiments,lr_tfidf_predict)
print("lr_tfidf_score :",lr_tfidf_score)
```

```
lr_bow_score : 0.7513
lr_tfidf_score : 0.7516
```

In [35]:
```
#Classification report for bag of words
lr_bow_report=classification_report(test_sentiments,lr_bow_predict,target_names=['
print(lr_bow_report)

#Classification report for tfidf features
lr_tfidf_report=classification_report(test_sentiments,lr_tfidf_predict,target_name
print(lr_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.75      0.75      4993
    Negative       0.75      0.75      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000

              precision    recall  f1-score   support

    Positive       0.74      0.77      0.76      4993
    Negative       0.76      0.73      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000
```

In [36]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,lr_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
cm_tfidf=confusion_matrix(test_sentiments,lr_tfidf_predict,labels=[1,0])
print(cm_tfidf)
```

```
[[3767 1240]
 [1247 3746]]
[[3675 1332]
 [1152 3841]]
```

In [37]:
```
#training the linear svm
svm=SGDClassifier(loss='hinge',max_iter=500,random_state=42)
#fitting the svm for bag of words
svm_bow=svm.fit(cv_train_reviews,train_sentiments)
print(svm_bow)
#fitting the svm for tfidf features
svm_tfidf=svm.fit(tv_train_reviews,train_sentiments)
print(svm_tfidf)
```

```
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
SGDClassifier(max_iter=500, random_state=42)
```

```
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
SGDClassifier(max_iter=500, random_state=42)
```

In [38]:
```
#Predicting the model for bag of words
svm_bow_predict=svm.predict(cv_test_reviews)
print(svm_bow_predict)
#Predicting the model for tfidf features
svm_tfidf_predict=svm.predict(tv_test_reviews)
print(svm_tfidf_predict)
```

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

In [39]:
```
#Accuracy score for bag of words
svm_bow_score=accuracy_score(test_sentiments,svm_bow_predict)
print("svm_bow_score :",svm_bow_score)
#Accuracy score for tfidf features
svm_tfidf_score=accuracy_score(test_sentiments,svm_tfidf_predict)
print("svm_tfidf_score :",svm_tfidf_score)
```

```
svm_bow_score : 0.5827
svm_tfidf_score : 0.5112
```

In [40]:
```
#Classification report for bag of words
svm_bow_report=classification_report(test_sentiments,svm_bow_predict,target_names=
print(svm_bow_report)
#Classification report for tfidf features
svm_tfidf_report=classification_report(test_sentiments,svm_tfidf_predict,target_nar
print(svm_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.94      0.18      0.30      4993
    Negative       0.55      0.99      0.70      5007

    accuracy                           0.58     10000
   macro avg       0.74      0.58      0.50     10000
weighted avg       0.74      0.58      0.50     10000

              precision    recall  f1-score   support

    Positive       1.00      0.02      0.04      4993
    Negative       0.51      1.00      0.67      5007

    accuracy                           0.51     10000
   macro avg       0.75      0.51      0.36     10000
weighted avg       0.75      0.51      0.36     10000
```

In [41]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,svm_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
```

```
cm_tfidf=confusion_matrix(test_sentiments,svm_tfidf_predict,labels=[1,0])
print(cm_tfidf)
```

```
[[4948   59]
 [4114  879]]
[[5007    0]
 [4888  105]]
```

In [42]:
```
#training the model
mnb=MultinomialNB()
#fitting the svm for bag of words
mnb_bow=mnb.fit(cv_train_reviews,train_sentiments)
print(mnb_bow)
#fitting the svm for tfidf features
mnb_tfidf=mnb.fit(tv_train_reviews,train_sentiments)
print(mnb_tfidf)
```

```
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
MultinomialNB()
C:\Users\Alekha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ), for exampl
e using ravel().
  y = column_or_1d(y, warn=True)
MultinomialNB()
```

In [43]:
```
#Predicting the model for bag of words
mnb_bow_predict=mnb.predict(cv_test_reviews)
print(mnb_bow_predict)
#Predicting the model for tfidf features
mnb_tfidf_predict=mnb.predict(tv_test_reviews)
print(mnb_tfidf_predict)
```

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

In [44]:
```
#Accuracy score for bag of words
mnb_bow_score=accuracy_score(test_sentiments,mnb_bow_predict)
print("mnb_bow_score :",mnb_bow_score)
#Accuracy score for tfidf features
mnb_tfidf_score=accuracy_score(test_sentiments,mnb_tfidf_predict)
print("mnb_tfidf_score :",mnb_tfidf_score)
```

```
mnb_bow_score : 0.7518
mnb_tfidf_score : 0.7518
```

In [45]:
```
#Classification report for bag of words
mnb_bow_report=classification_report(test_sentiments,mnb_bow_predict,target_names=
print(mnb_bow_report)
#Classification report for tfidf features
mnb_tfidf_report=classification_report(test_sentiments,mnb_tfidf_predict,target_nar
print(mnb_tfidf_report)
```

	precision	recall	f1-score	support
Positive	0.75	0.76	0.75	4993
Negative	0.75	0.75	0.75	5007
accuracy			0.75	10000
macro avg	0.75	0.75	0.75	10000
weighted avg	0.75	0.75	0.75	10000

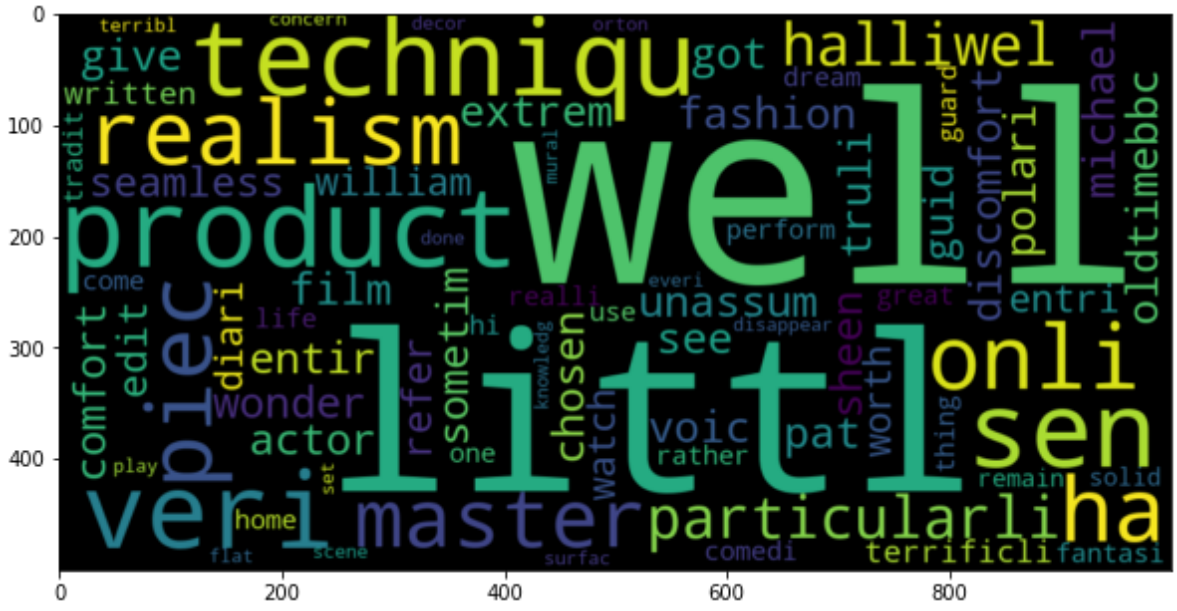
	precision	recall	f1-score	support
Positive	0.75	0.76	0.75	4993
Negative	0.75	0.75	0.75	5007
accuracy			0.75	10000
macro avg	0.75	0.75	0.75	10000
weighted avg	0.75	0.75	0.75	10000

In [46]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,mnb_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
cm_tfidf=confusion_matrix(test_sentiments,mnb_tfidf_predict,labels=[1,0])
print(cm_tfidf)
```

```
[[3744 1263]
 [1219 3774]]
[[3738 1269]
 [1213 3780]]
```

In [47]:
```
#word cloud for positive review words
plt.figure(figsize=(10,10))
positive_text=norm_train_reviews[1]
WC=WordCloud(width=1000,height=500,max_words=500,min_font_size=5)
positive_words=WC.generate(positive_text)
plt.imshow(positive_words,interpolation='bilinear')
plt.show
```

Out[47]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [48]:
```
#Word cloud for negative review words
plt.figure(figsize=(10,10))
negative_text=norm_train_reviews[8]
```

```
WC=WordCloud(width=1000,height=500,max_words=500,min_font_size=5)
negative_words=WC.generate(negative_text)
plt.imshow(negative_words,interpolation='bilinear')
plt.show
```

Out[48]: <function matplotlib.pyplot.show(close=None, block=None)>

In [ ]: