In [12]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
import re
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
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
```

In [6]:
```
imdb_data=pd.read_csv('IMDB Dataset.csv')
print(imdb_data.shape)
imdb_data.head(10)
```

```
(50000, 2)
```

Out[6]:

	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 [7]:
```
imdb_data.describe()
```

Out[7]:

	review	sentiment
count	50000	50000
unique	49582	2

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

In [8]:
```
imdb_data['sentiment'].value_counts()
```

Out[8]:
```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

In [9]:
```
#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 [13]:
```
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)
```

In [14]:
```
#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 [15]:
```
#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 [16]:
```
#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 [20]:
```
#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[20]:
```
'one of the other review ha mention that after watch just 1 oz episod youll be hook they
are right as thi is exactli what happen with meth first thing that struck me about oz wa
it brutal and unflinch scene of violenc which set in right from the word go trust me thi
is not a show for the faint heart or timid thi show pull no punch with regard to drug se
x or violenc it is hardcor in the classic use of the wordit is call oz as that is the ni
cknam given to the oswald maximum secur state penitentari it focu mainli on emerald citi
an experi section of the prison where all the cell have glass front and face inward so p
rivaci is not high on the agenda em citi is home to manyaryan muslim gangsta latino chri
stian italian irish and moreso scuffl death stare dodgi deal and shadi agreement are nev
er far awayi would say the main appeal of the show is due to the fact that it goe where
other show wouldnt dare forget pretti pictur paint for mainstream audienc forget charm f
orget romanceoz doesnt mess around the first episod i ever saw struck me as so nasti it
wa surreal i couldnt say i wa readi for it but as i watch more i develop a tast for oz a
nd got accustom to the high level of graphic violenc not just violenc but injust crook g
uard wholl be sold out for a nickel inmat wholl kill on order and get away with it well
manner middl class inmat be turn into prison bitch due to their lack of street skill or
prison experi watch oz you may becom comfort with what is uncomfort viewingthat if you c
an get in touch with your darker side'
```

In [21]:
```
#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[21]:
```
'i read all the review here after watch thi piec of cinemat garbag and it took me at lea
st 2 page to find out that somebodi el didnt think that thi appallingli unfunni montag w
asnt the acm of humour in the 70 or ind in ani other era if thi isnt the least funni set
```

```
of sketch comedi ive ever seen itll do till it come along half of the skit had alreadi b
een done and infinit better by act such as monti python and woodi allen if i wa to say t
hat a nice piec of anim that last about 90 second is the highlight of thi film it would
still not get close to sum up just how mindless and drivelridden thi wast of 75 minut is
semin comedi onli in the world where semin realli doe mean semen scatolog humour onli in
a world where scat is actual fece precursor joke onli if by that we mean that thi is a h
andbook of how not to do comedi tit and bum and the odd beaver niceif you are a pubesc b
oy with at least one hand free and havent found out that playboy exist give it a break b
ecau it wa the earli 70 no way there had been sketch comedi go back at least ten year pr
ior the onli way i could even forgiv thi film even be made is if it wa at gunpoint retro
hardli sketch about clown subtli pervert children may be cut edg in some circl and it co
uld actual have been funni but it just come off as realli quit sad what kept me go throu
ghout the entir 75 minut sheer belief that they may have save a genuin funni skit for th
e end i gave the film a 1 becau there wa no lower scoreand i can onli recommend it to in
somniac or coma patientsor perhap peopl suffer from lockjawtheir jaw would final drop op
en in disbelief'
```

In [22]:

```
#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, 6020317)
BOW_cv_test: (10000, 6020317)
```

In [23]:

```
#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, 6020317)
Tfidf_test: (10000, 6020317)
```

In [24]:

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

```
(50000, 1)
```

In [25]:

```
#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 [26]:

```
#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\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
LogisticRegression(C=1, max_iter=500, random_state=42)
C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
LogisticRegression(C=1, max_iter=500, random_state=42)
```

In [27]:

```
#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)
```

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

In [28]:

```
#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 : 0.7591
lr_tfidf_score : 0.7563
```

In [29]:

```
#Classification report for bag of words
lr_bow_report=classification_report(test_sentiments,lr_bow_predict,target_names=['Posit
```

```
print(lr_bow_report)

#Classification report for tfidf features
lr_tfidf_report=classification_report(test_sentiments,lr_tfidf_predict,target_names=['P
print(lr_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.76      0.76      0.76      4993
    Negative       0.76      0.76      0.76      5007

    accuracy                           0.76     10000
   macro avg       0.76      0.76      0.76     10000
weighted avg       0.76      0.76      0.76     10000

              precision    recall  f1-score   support

    Positive       0.75      0.77      0.76      4993
    Negative       0.77      0.74      0.75      5007

    accuracy                           0.76     10000
   macro avg       0.76      0.76      0.76     10000
weighted avg       0.76      0.76      0.76     10000
```

In [30]:

```
#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)
```

```
[[3813 1194]
 [1215 3778]]
[[3700 1307]
 [1130 3863]]
```

In [31]:

```
#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\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
SGDClassifier(max_iter=500, random_state=42)
C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
SGDClassifier(max_iter=500, random_state=42)
```

In [32]:
```
#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 0 1 ... 0 1 1]
[1 1 1 ... 1 1 1]
```

In [33]:
```
#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 : 0.6227
svm_tfidf_score : 0.5111
```

In [34]:
```
#Classification report for bag of words
svm_bow_report=classification_report(test_sentiments,svm_bow_predict,target_names=['Pos
print(svm_bow_report)
#Classification report for tfidf features
svm_tfidf_report=classification_report(test_sentiments,svm_tfidf_predict,target_names=[
print(svm_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.92      0.27      0.41      4993
    Negative       0.57      0.98      0.72      5007

    accuracy                           0.62     10000
   macro avg       0.75      0.62      0.57     10000
weighted avg       0.75      0.62      0.57     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 [35]:
```
#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)
```

```
[[4896  111]
 [3662 1331]]
[[5007    0]
 [4889  104]]
```

In [36]:

```
#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\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
MultinomialNB()
C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shap
e of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
MultinomialNB()
```

In [37]:

```
#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)
```

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

In [38]:

```
#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 : 0.7562
mnb_tfidf_score : 0.756
```

In [39]:

```
#Classification report for bag of words
mnb_bow_report=classification_report(test_sentiments,mnb_bow_predict,target_names=['Pos
print(mnb_bow_report)
#Classification report for tfidf features
mnb_tfidf_report=classification_report(test_sentiments,mnb_tfidf_predict,target_names=[
print(mnb_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.76      0.76      4993
    Negative       0.76      0.75      0.76      5007
```

```
              accuracy                           0.76     10000
             macro avg       0.76      0.76      0.76     10000
          weighted avg       0.76      0.76      0.76     10000

                        precision    recall  f1-score   support

              Positive       0.75      0.76      0.76      4993
              Negative       0.76      0.75      0.75      5007

              accuracy                           0.76     10000
             macro avg       0.76      0.76      0.76     10000
          weighted avg       0.76      0.76      0.76     10000
```

In [40]:

```
#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)
```

```
[[3759 1248]
 [1190 3803]]
[[3752 1255]
 [1185 3808]]
```

In [ ]:

In [ ]:

In [ ]: