In [1]:
```
import numpy as np
import pandas as pd
import re
import os
```

In [2]:
```
negData="Assign3dataset/train/neg/"
posData="Assign3dataset/train/pos/"
negFiles = os.listdir(negData)
posFiles = os.listdir(posData)
msgNeg=[]
msgPos=[]
for i in negFiles:
    f = open(negData+i,'r')
    msgNeg.append(f.read())
for i in posFiles:
    f = open(posData+i,'r')
    msgPos.append(f.read())
```

In [3]:
```
msgNegStr=np.str(msgNeg)
msgPosStr=np.str(msgPos)
```

In [5]:
```
import re
msgNegStr2 = re.sub(r'[^\w\s]','',msgNegStr)
msgPosStr2 = re.sub(r'[^\w\s]','',msgPosStr)
```

In [6]:
```
msgNegStrLow=msgNegStr2.lower()
msgPosStrLow=msgPosStr2.lower()
```

In [7]:
```
from nltk.tokenize import word_tokenize
wdNeg_tkn=word_tokenize(msgNegStrLow)
wdPos_tkn=word_tokenize(msgPosStrLow)
```

In [8]:
```
msgNegStrLow[0:16],wdNeg_tkn[0:6],msgPosStrLow[0:16],wdPos_tkn[0:6]
```

Out[8]:
```
('working with one',
 ['working', 'with', 'one', 'of', 'the', 'best'],
 'for a movie that',
 ['for', 'a', 'movie', 'that', 'gets', 'no'])
```

In [42]:
```
len(msgNeg)
```

Out[42]:
```
12500
```

In [9]:
```
from nltk.corpus import stopwords
```

In [10]:
```
# stWds=re.compile('|'.join(map(re.escape,stopwords.words("english"))))
stpwrds=set(stopwords.words("english"))
```

In [11]:
```
WdNeg_tkn2=[]
WdPos_tkn2=[]
for i in wdNeg_tkn:
    if i not in stpwrds:
        WdNeg_tkn2.append(i)
for i in wdPos_tkn:
    if i not in stpwrds:
        WdPos_tkn2.append(i)
```

In [12]:
```
print(WdNeg_tkn2[0:10]),print(WdPos_tkn2[0:10])
```

```
['working', 'one', 'best', 'shakespeare', 'sources', 'film', 'manages',
'creditable', 'source', 'whilst']
['movie', 'gets', 'respect', 'sure', 'lot', 'memorable', 'quotes', 'lis
ted', 'gem', 'imagine']
```

Out[12]: (None, None)

In [13]:
```
# Need to run first two lines
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
def cleaning(msgNeg,msgPos):
    msgNegStr=np.str(msgNeg)
    msgPosStr=np.str(msgPos)
    msgNegStr2 = re.sub(r'[^\w\s]','',msgNegStr)
    msgPosStr2 = re.sub(r'[^\w\s]','',msgPosStr)
    msgNegStrLow=msgNegStr2.lower()
    msgPosStrLow=msgPosStr2.lower()
    wdNeg_tkn=word_tokenize(msgNegStrLow)
    wdPos_tkn=word_tokenize(msgPosStrLow)
    stpwrds=set(stopwords.words("english"))
    WdNeg_tkn2=[]
    WdPos_tkn2=[]
    for i in wdNeg_tkn:
        if i not in stpwrds:
            WdNeg_tkn2.append(i)
    for i in wdPos_tkn:
        if i not in stpwrds:
            WdPos_tkn2.append(i)
    return WdNeg_tkn2, WdPos_tkn2
WdNeg_tkn3,WdPos_tkn3=cleaning(msgNeg,msgPos)
```

In [14]:
```
print(WdNeg_tkn3[0:10]),print(WdPos_tkn3[0:10])
```

```
['working', 'one', 'best', 'shakespeare', 'sources', 'film', 'manages',
'creditable', 'source', 'whilst']
['movie', 'gets', 'respect', 'sure', 'lot', 'memorable', 'quotes', 'lis
ted', 'gem', 'imagine']
```

Out[14]: (None, None)

In [68]:
```
list_data=[]
for i in msgNeg:
    list_data.append("".join(i))
for i in msgPos:
    list_data.append("".join(i))
```

In [71]:
```
positive_class=np.ones(12500,dtype=int)
negative_class=np.zeros(12500,dtype=int)
```

In [72]:
```
list_class=[]
for i in positive_class:
    list_class.append(i)
for i in negative_class:
    list_class.append(i)
```

In [73]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer =CountVectorizer()
```

In [74]:
```
text=vectorizer.fit_transform(list_data).toarray()
```

In [75]:
```
vocab=vectorizer.vocabulary_
print(list(vocab.keys())[0:10])
```

```
['working', 'with', 'one', 'of', 'the', 'best', 'shakespeare', 'source
s', 'this', 'film']
```

In [76]:
```
import pandas as pd
df=pd.DataFrame(text,columns=vectorizer.get_feature_names())
class_text=pd.Series(list_class)
```

In [77]:
```
len(df),len(df.iloc[0])
```

Out[77]: (25000, 74849)

In [83]:
```
df.tail(5)
```

Out[83]:

	00	000	0000000000001	00001	00015	000s	001	003830	006	007	...	était	état
24995	0	0	0	0	0	0	0	0	0	0	...	0	0
24996	0	0	0	0	0	0	0	0	0	0	...	0	0
24997	0	0	0	0	0	0	0	0	0	0	...	0	0
24998	0	0	0	0	0	0	0	0	0	0	...	0	0
24999	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 74849 columns

In [84]:
```
#for spliting your data into training and test set
from sklearn.model_selection import train_test_split
#for analyzing the result of your model
from sklearn import metrics
#the two naive bayes algorithms
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

In [85]:
```
x_train,x_test,y_train,y_test=train_test_split(df,class_text,test_size=
0.25,random_state=5)
```

In [86]:
```
x_train=np.asarray(x_train)
y_train=np.asarray(y_train)
```

In [87]:
```
Alg_NB=GaussianNB()
```

In [88]:
```
Alg_NB.fit(x_train,y_train)
```

Out[88]:
```
GaussianNB(priors=None)
```

In [90]:
```
output=Alg_NB.predict(x_test)
output
```

Out[90]:
```
array([0, 0, 1, ..., 0, 1, 1])
```

In [91]:
```
metrics.accuracy_score(y_test,output)
```

Out[91]:
```
0.66688
```

In [92]:
```
metrics.confusion_matrix(y_test,output)
```

Out[92]:
```
array([[1696, 1421],
       [ 661, 2472]])
```