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

In [2]:
```
temp_df = pd.read_csv('IMDB Dataset - IMDB Dataset.csv')
```

In [3]:
```
df = temp_df.iloc[:10000]
```

In [4]:
```
df.head()
```

Out[4]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />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 [5]:
```
df['review'][1]
```

Out[5]:
```
'A wonderful little production. <br /><br />The filming technique is very unassu
ming- very old-time-BBC fashion and gives a comforting, and sometimes discomfort
ing, sense of realism to the entire piece. <br /><br />The actors are extremely
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 refere
nces to Williams\' diary entries, not only is it well worth the watching but 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 really comes
home with the little things: the fantasy of the guard which, rather than use the
traditional \'dream\' techniques remains solid then disappears. It plays on our
knowledge and our senses, particularly with the scenes concerning Orton and Hall
iwell and the sets (particularly of their flat with Halliwell\'s murals decorati
ng every surface) are terribly well done.'
```

In [6]:
```
df['sentiment'].value_counts()
```

Out[6]:
```
positive    5028
negative    4972
Name: sentiment, dtype: int64
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
review       0
sentiment    0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
17
```

In [11]:
```
df = df.drop_duplicates()
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
# Basic Preprocessing
# Remove tags
# lowercase
# remove stopwords
```

In [14]:
```
import re
def remove_tags(raw_text):
    cleaned_text = re.sub(re.compile('<.*?>'), '', raw_text)
    return cleaned_text
```

In [15]:
```
df['review'] = df['review'].apply(remove_tags)
```

In [16]:
```
df
```

Out[16]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The filming tec...	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
...	...	...
9995	Fun, entertaining movie about WWII German spy ...	positive
9996	Give me a break. How can anyone say that this ...	negative
9997	This movie is a bad movie. But after watching ...	negative
9998	This is a movie that was probably made to ente...	negative
9999	Smashing film about film-making. Shows the int...	positive

9983 rows × 2 columns

In [17]:
```
df['review'] = df['review'].apply(lambda x:x.lower())
```

In [20]:
```
X = df.iloc[:,0:1]
y = df['sentiment']
```

In [24]:
```
X
```

Out[24]:

	review
0	one of the other reviewers has mentioned that ...
1	a wonderful little production. the filming tec...
2	i thought this was a wonderful way to spend ti...
3	basically there's a family where a little boy ...
4	petter mattei's "love in the time of money" is...
...	...
9995	fun, entertaining movie about wwii german spy ...
9996	give me a break. how can anyone say that this ...
9997	this movie is a bad movie. but after watching ...
9998	this is a movie that was probably made to ente...
9999	smashing film about film-making. shows the int...

9983 rows × 1 columns

In [25]:
```
y
```

Out[25]:
```
0       positive
1       positive
2       positive
3       negative
4       positive
          ...
9995    positive
9996    negative
9997    negative
9998    negative
9999    positive
Name: sentiment, Length: 9983, dtype: object
```

In [26]:
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

y = encoder.fit_transform(y)
```

In [27]:
```
y
```

Out[27]:
```
array([1, 1, 1, ..., 0, 0, 1])
```

In [28]:
```
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,random_state=1
```

In [29]:
```
X_train.shape
```

Out[29]:
```
(7986, 1)
```

In [30]:
```
# Applying BoW
from sklearn.feature_extraction.text import CountVectorizer
```

In [31]:
```
cv = CountVectorizer()
```

In [32]:
```
X_train_bow = cv.fit_transform(X_train['review']).toarray()
X_test_bow = cv.transform(X_test['review']).toarray()
```

In [34]:
```
X_train_bow.shape
```

Out[34]:
```
(7986, 48284)
```

In [35]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()

gnb.fit(X_train_bow,y_train)
```

Out[35]:
```
▾ GaussianNB

GaussianNB()
```

In [36]:
```
y_pred = gnb.predict(X_test_bow)

from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(y_test,y_pred)
```

Out[36]:
```
0.6364546820230346
```

In [37]:
```
confusion_matrix(y_test,y_pred)
```

Out[37]:
```
array([[716, 236],
       [490, 555]], dtype=int64)
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()

rf.fit(X_train_bow,y_train)
y_pred = rf.predict(X_test_bow)
accuracy_score(y_test,y_pred)
```

Out[38]:
```
0.829744616925388
```

In [39]:
```
cv = CountVectorizer(max_features=3000)

X_train_bow = cv.fit_transform(X_train['review']).toarray()
X_test_bow = cv.transform(X_test['review']).toarray()

rf = RandomForestClassifier()

rf.fit(X_train_bow,y_train)
y_pred = rf.predict(X_test_bow)
accuracy_score(y_test,y_pred)
```

Out[39]:
```
0.8342513770655984
```

In [41]:
```
cv = CountVectorizer(ngram_range=(1,2),max_features=5000)

X_train_bow = cv.fit_transform(X_train['review']).toarray()
X_test_bow = cv.transform(X_test['review']).toarray()

rf = RandomForestClassifier()

rf.fit(X_train_bow,y_train)
y_pred = rf.predict(X_test_bow)
accuracy_score(y_test,y_pred)
```

Out[41]:
```
0.8327491236855283
```