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

In [2]:
```
#read data from csv file
df_sentiment =pd.read_csv(r'C:\Users\ctoqu\Desktop\imdb_labelled.txt',sep='\t'
,names=['comment','label'])
```

In [3]:
```
#View 10 firsts comments.
#1 indicates positive sentiment and 0 negative
df_sentiment.head(10)
```

Out[3]:

	comment	label
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
5	The rest of the movie lacks art, charm, meanin...	0
6	Wasted two hours.	0
7	Saw the movie today and thought it was a good ...	1
8	A bit predictable.	0
9	Loved the casting of Jimmy Buffet as the scien...	1

In [4]:
```
#view data statistics using describe()
df_sentiment.describe()
```

Out[4]:

	label
count	748.000000
mean	0.516043
std	0.500077
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

In [5]:
```
# View more infoo on data
df_sentiment.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   comment  748 non-null    object
 1   label    748 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 11.8+ KB
```

In [6]:
```
# view data using groupby and describe method. Statistical summary re label
df_sentiment.groupby('label').describe()
```

Out[6]:

	comment			
	count	unique	top	freq
label				
0	362	361	Not recommended.	2
1	386	384	10/10	2

In [9]:
```
# Apply(len) method to add new column follow by the head method to view the le
nght of the messages
df_sentiment['lenght'] =df_sentiment['comment'].apply(len)
```

In [10]:
```
#view columns of the dataset
df_sentiment.head()
```

Out[10]:

	comment	label	lenght
0	A very, very, very slow-moving, aimless movie ...	0	87
1	Not sure who was more lost - the flat characte...	0	99
2	Attempting artiness with black & white and cle...	0	188
3	Very little music or anything to speak of.	0	44
4	The best scene in the movie was when Gerardo i...	1	108

In [11]:
```
#View the first comment which has lenght greater than 50
df_sentiment[df_sentiment['lenght']>50]['comment'].iloc[0]
```

Out[11]:
```
'A very, very, very slow-moving, aimless movie about a distressed, drifting y
oung man.  '
```

In [12]:
```
# start text processing with vectorizer
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

In [15]:
```
#Define a function to get rid of stopwords present in the messages
def message_text_process(mess):
#Chech characters to see if there are punctuations
    no_punctuation = [char for char in mess if char not in string.punctuation]
# now form the sentence
    no_punctuation = ''.join(no_punctuation)
# Eliminate any stopwords
    return [word for word in no_punctuation.split() if word.lower() not in sto
pwords.words('english')]
```

In [18]:
```
# bag of words by applying the function and fit the data (comment) into it
import string
from nltk.corpus import stopwords
bag_of_words = CountVectorizer(analyzer=message_text_process).fit(df_sentiment
['comment'])
```

In [19]:
```
#apply transform method for the bag of words
comment_bagofwords = bag_of_words.transform(df_sentiment['comment'])
```

In [21]:
```
#Apply tfidf transformer and fit the bag of words into it (transform version)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer =TfidfTransformer().fit(comment_bagofwords)
```

In [22]:
```
#Print shape of tfidf
comment_tfidf = tfidf_transformer.transform(comment_bagofwords)
print(comment_tfidf.shape)
```

```
(748, 3259)
```

In [23]:
```
#choose naive Bayes model to detect the spam and fit the tfidf data into it
from sklearn.naive_bayes import MultinomialNB
sentiment_detection_model = MultinomialNB().fit(comment_tfidf,df_sentiment['la
bel'])
```

In [26]:
```
#check model for the predicted and expected value say for comment #1 and #5
comment = df_sentiment['comment'][0]
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)

print('predicted sentiment label', sentiment_detection_model.predict(tfidf)[0
])
print('expected sentiment label', df_sentiment.label[0])
```

```
predicted sentiment label 0
expected sentiment label 0
```

In [27]:
```
comment = df_sentiment['comment'][4]
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)

print('predicted sentiment label', sentiment_detection_model.predict(tfidf)[0
])
print('expected sentiment label', df_sentiment.label[4])
```

```
predicted sentiment label 1
expected sentiment label 1
```

In [ ]:
```

```