In [1]:

```
#mport required libraty
import pandas as pd
```

In [2]:

```
#import dataset
df=pd.read_csv("imdb_labelled.txt",sep='\t',names=["comment","label"])
df
```

Out[2]:

	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
...	...	...
743	I just got bored watching Jessice Lange take h...	0
744	Unfortunately, any virtue in this film's produ...	0
745	In a word, it is embarrassing.	0
746	Exceptionally bad!	0
747	All in all its an insult to one's intelligence...	0

748 rows × 2 columns

In [3]:

```
#view top 5 rows
df.head()
```

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

In [4]:

```
#view data using describe method
df.describe
```

Out[4]:

```
<bound method NDFrame.describe of                                                         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
..                                                 ...    ...
743  I just got bored watching Jessice Lange take h...      0
```

```
744  Unfortunately, any virtue in this film's produ...      0
745                    In a word, it is embarrassing.      0
746                                 Exceptionally bad!      0
747  All in all its an insult to one's intelligence...      0

[748 rows x 2 columns]>
```

In [5]:

```
#view more data use info
df.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 [7]:

```
#view data using groupby with describe method
df.groupby("label").describe()
```

Out[7]:

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

In [13]:

```
#add column length to the df
df['length'] =df['comment'].apply(len)
```

In [14]:

```
#view the column added and top 5 rows.
df.head()
```

Out[14]:

	comment	label	length
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 [15]:

```
#apply a filter on length col to get morethan 50 of length data
df[df['length']>50]['comment'].iloc[0]
```

Out[15]:

```
'A very, very, very slow-moving, aimless movie about a distressed, drifting young man.  '
```

In [16]:

```
#text processing with vectorization
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
```

In [17]:

```
# define a function to get rid of stopwords present in the messages
def message_text_process(mess):
    # Check 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)
    # Now eliminate any stopwords
    return [word for word in no_punctuation.split() if word.lower() not in stopwords.wor
ds('english')]
```

In [19]:

```
# 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['comment'])
```

In [20]:

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

In [21]:

```
# apply tfidf transformer and fit the bag of words into it (transformed version)
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(comment_bagofwords)
```

In [23]:

```
# print shape of the tfidf
comment_tfidf = tfidf_transformer.transform(comment_bagofwords)
print (comment_tfidf.shape)
```

```
(748, 3259)
```

In [25]:

```
#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['label'])
```

In [31]:

```
# check model for the predicted  and expected value say for comment# 1 and comment#5
comment = df['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.label[4])
```

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

In [32]:

```
print("Sucessfully completed a Project on NLP sentiment analysis")
```

```
Sucessfully completed a Project on NLP sentiment analysis
```

In [34]:

```
print("Thank you Simplilearn")
```

```
Thank you Simplilearn
```

In [ ]:

```

```