Sentiment Analysis

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

In [2]:
```
# load the dataset
df_sentiment = pd.read_csv('imdb_labelled.txt',sep='\t',names=['comment','label'])
```

In [3]:
```
# view the first ten observation
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 more information
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]:
```
# columns names and data types
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 [7]:
```
# view using groupby and describe
df_sentiment.groupby('label').describe()
```

Out[7]:

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

In [9]:
```
# add one column lenght which is equal to length of comment
df_sentiment['length']= df_sentiment['comment'].apply(len)
```

In [10]:
```
# view the first five rows to check the length of comments
df_sentiment.head()
```

Out[10]:

	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 [11]:
```
# import Count vectorizer
from sklearn.feature_extraction.text import CountVectorizer
# initialise a instance of CountVectozer
vectorizer = CountVectorizer()
```

In [15]:
```
# import library
import string
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[15]:
```
True
```

In [13]:
```
# define a code to get rid of punctuation and stopwords
def message_text_process(mess):
  # check and remove punctuation
  no_punctuations = [char for char in mess if char not in string.punctuation]
  # join the words to form sentences
  no_punctuations = ''.join(no_punctuations)
  # remove stopwords
  return[word for word in no_punctuations.split() if word.lower() not in stopwords.words('english')]
```

Apply CountVectorizer and TfidfTransformer

In [16]:
```
# create bag of words
bag_of_words = CountVectorizer(analyzer=message_text_process).fit(df_sentiment['comment'])
```

In [17]:
```
# apply transform method
comment_bagofwords = bag_of_words.transform(df_sentiment['comment'])
```

In [18]:
```
# import tfidf
from sklearn.feature_extraction.text import TfidfTransformer
# apply tfidf transformer fit method
tfidf_transformer = TfidfTransformer().fit(comment_bagofwords)
```

In [19]:
```
# apply tfidf transformer transform method
comment_tfidf = tfidf_transformer.transform(comment_bagofwords)
```

In [20]:
```
# print the shape of tfidf
print(comment_tfidf.shape)
```

```
(748, 3259)
```

Build model and test model

In [21]:
```
# import naive bayes model
from sklearn.naive_bayes import MultinomialNB
```

In [22]:
```
# fit the tfidf data into naive bayes model
sentiment_detection_model = MultinomialNB().fit(comment_tfidf,df_sentiment['label'])
```

In [23]:
```
# check model for comment 5
comment = df_sentiment['comment'][4]
comment
```

Out[23]:
```
'The best scene in the movie was when Gerardo is trying to find a song that keeps running through his head.  '
```

In [24]:
```
# apply count vectorizer and tfidf tranformer to message
bag_of_words_for_comment = bag_of_words.transform([comment])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment)
```

In [27]:
```
# make predition
print('predicted label is ', sentiment_detection_model.predict(tfidf)[0])
```

```
predicted label is  1
```

In [29]:
```
# actual label
print('actual label is' , df_sentiment.label[4])
```

```
actual label is 1
```

In [31]:
```
# comment 1
comment1 = df_sentiment['comment'][0]
comment1
```

Out[31]:
```
'A very, very, very slow-moving, aimless movie about a distressed, drifting young man.  '
```

In [32]:
```
# apply count vectorizer and tfidf tranformer to comment1
bag_of_words_for_comment1 = bag_of_words.transform([comment1])
tfidf = tfidf_transformer.transform(bag_of_words_for_comment1)
```

In [33]:
```
# make predition
print('predicted label is ', sentiment_detection_model.predict(tfidf)[0])
```

```
predicted label is  0
```

In [34]:
```
# actual label
print('actual label is' , df_sentiment.label[0])
```

```
actual label is 0
```