Importing Libraries

In [1]:
```
import os
import re
import numpy as np
import pandas as pd
import nltk
from nltk.corpus  import stopwords
from sklearn.feature_extraction.text import CountVectorizer
```

Creating neg list with review files

In [2]:
```
negTweets = []
i = 0
for file in os.listdir("data/test/neg"):
    if file.endswith(".txt"):
        #print(os.path.join("/data/test/neg", file))
        File2 = open(os.path.join("data/test/neg", file), encoding="utf8")
        #print(i)
        negTweets.append(File2.read())
        i = i + 1
negTweets = negTweets[0:200]
print(len(negTweets))
```

```
200
```

Creating pos list with review files

In [3]:
```
posTweets = []
i = 0
for file in os.listdir("data/test/pos"):
    if file.endswith(".txt"):
        #print(os.path.join("/data/test/pos", file))
        File2 = open(os.path.join("data/test/pos", file), encoding="utf8")
        #print(i)
        posTweets.append(File2.read())
        i = i + 1
posTweets = posTweets[0:200]
print(len(posTweets))
```

```
200
```

Removing punctuation and number. Changing to lowercase

In [4]:
```
i = 0
arraylen = len(negTweets)
for i in range(arraylen):
    negTweets[i] = re.sub('[^A-Za-z ]+',"",negTweets[i]).lower()
i = 0
arraylen = len(posTweets)
for i in range(arraylen):
    posTweets[i] = re.sub('[^A-Za-z ]+',"",posTweets[i]).lower()
```

Tokenizing list of reviews

In [5]:
```
arraylen = len(negTweets)
negToken = []
for i in range(arraylen):
    negToken.append(negTweets[i].split())

arraylen = len(posTweets)
posToken = []
for i in range(arraylen):
    posToken.append(posTweets[i].split())

print(len(negToken))
print(len(posToken))
```

```
200
200
```

Making a flat list with all the words

In [6]:
```
flatNegToken = []
arraylen = len(negToken)
for i in range(arraylen):
    for item in negToken[i]:
        flatNegToken.append(item)
print(len(flatNegToken))


flatPosToken = []
arraylen = len(posToken)
for i in range(arraylen):
    for item in posToken[i]:
        flatPosToken.append(item)
print(len(flatPosToken))
```

```
43484
49978
```

Create stopwords list from NLTK

In [7]:
```
stopwords = set(stopwords.words('english'))
```

Filtering stopwords from data

In [8]:
```
filteredFlatNegToken = []
for item in flatNegToken:
    if item not in stopwords:
        filteredFlatNegToken.append(item)

filteredFlatPosToken = []
for item in flatPosToken:
    if item not in stopwords:
        filteredFlatPosToken.append(item)

print(len(filteredFlatNegToken), len(filteredFlatPosToken))

print(filteredFlatNegToken[1:9], filteredFlatPosToken[1:9])
```

```
23126 26496
['costner', 'dragged', 'movie', 'far', 'longer', 'necessary', 'aside', 'terrific'] ['saw', 'movie', 'last', 'night', 'coaxed', 'friends', 'mine', 'ill']
```

Creating class list

In [12]:
```
targetClass = []
negTarget = np.zeros(len(negTweets), dtype='int')
posTarget = np.ones(len(posTweets), dtype='int')

for items in negTarget:
    targetClass.append(items)
for items in posTarget:
    targetClass.append(items)
print(len(targetClass))
```

```
400
```

Creating masterList

In [13]:
```
masterToken = []
for items in negTweets:
    masterToken.append(items)
len(masterToken)
for items in posTweets:
    masterToken.append(items)
len(masterToken)
```

Out[13]:
```
400
```

Creating Vectorizer object

In [15]:
```
vec = CountVectorizer(stop_words='english')
```

Creating DataFrame from countVectorizer

In [17]:
```
vecMasterToken = vec.fit_transform(masterToken).toarray()
df=pd.DataFrame(vecMasterToken,columns=vec.get_feature_names())
df.sum()
```

Out[17]:
```
aback                1
abandoned            3
abandons             1
abbott               1
abcs                 1
abducted             2
abduction            2
abductions           1
abel                 1
abetted              1
abilities            1
ability              6
able                16
ably                 1
abodes               1
abortion             1
aboutyou             1
abr                  3
abre                 1
abroad               3
abrupt               1
abruptly             2
absence              2
absent               1
absolute             3
absolutely          25
absorb               2
absorbing            1
absutely             1
abstract             1
                   ...
youth                4
youthful             1
youve               14
yr                   1
yum                  2
yuppie               1
zanatoes             1
zaniness             1
zany                 2
zaphod               5
zaphods              1
zapped               1
zealous              1
zeffirellis          1
zenith               1
zero                 2
zeugma               1
zeus                 4
zhang                1
ziegfield            1
zillion              1
zingers              1
zombie               1
zombies              1
zone                 3
zooms                1
zseries              1
zuotian              1
zwart                2
zzzzzzzzzzzz         1
Length: 10734, dtype: int64
```

Creating Target Class series

In [20]:
```
targetClassSeries=pd.Series(targetClass)
targetClassSeries
```

Out[20]:
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0
21     0
22     0
23     0
24     0
25     0
26     0
27     0
28     0
29     0
      ..
370    1
371    1
372    1
373    1
374    1
375    1
376    1
377    1
378    1
379    1
380    1
381    1
382    1
383    1
384    1
385    1
386    1
387    1
388    1
389    1
390    1
391    1
392    1
393    1
394    1
395    1
396    1
397    1
398    1
399    1
Length: 400, dtype: int64
```

Machine Learning

In [18]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

split train data

In [21]:
```
x_train,x_test,y_train,y_test=train_test_split(df,targetClassSeries,test_size=0.2,random_state=5)
```

Training model

In [22]:
```
algorithm_a=MultinomialNB()
#model.fit(data,target)
#training your data
algorithm_a.fit(x_train,y_train)
#testing your data
output=algorithm_a.predict(x_test)
```

In [23]:
```
metrics.accuracy_score(y_test,output)
```

Out[23]:
```
0.9625
```