In [44]:

```
import pandas as pd
import numpy as np
from collections import defaultdict
import re
```

In [45]:

```
Train = pd.read_csv('Train.csv')
Test = pd.read_csv('Test.csv')
```

In [46]:

```
Train.head(n=5)
Train = Train.values
```

In [47]:

```
X_Train = Train[:,0]
Y_Train = Train[:,-1]
print(X_Train[5])
print(X_Train.shape)
print(Y_Train[5])
print(Y_Train.shape)
```

```
Steve Carell comes into his own in his first starring role in the 40 Year Old Virgin, having only had supporti
witched, Bruce Almighty, Anchorman, and his work on the Daily Show, we had only gotten a small taste of the co
his own. You can tell that Will Ferrell influenced his "comedic air" but Carell takes it to another level, eve
lovable, and hilarious. I would not hesitate to say that Steve Carell is one of the next great comedians of ou
ar Old Virgin is two hours of non-stop laughs (or 4 hours if you see it twice like I did), a perfect supportin
the audience through the entire movie. The script was perfect with so many great lines that you will want to s
ry to remember them all. The music fit the tone of the movie great, and you can tell the director knew what he
d with sex jokes, some nudity, and a lot of language, this movie isn't for everyone but if you liked the Weddi
ny movie along those lines, you will absolutely love The 40 Year Old Virgin.
(40000,)
pos
(40000,)
```

In [48]:

```
def preprocess_string(str_arg):
    cleaned_str=re.sub('[^a-z\s]+',' ',str_arg,flags=re.IGNORECASE) #every char except
    cleaned_str=re.sub('(\s+)',' ',cleaned_str) #multiple spaces are replaced by singl
    cleaned_str=cleaned_str.lower() #converting the cleaned string to lower case

    return cleaned_str # returning the preprocessed string
```

In [49]:

```
class NaiveBayes:

    def __init__(self,unique_classes):

        self.classes=unique_classes # Constructor is sinply passed with unique number c


    def addToBow(self,example,dict_index):

        #print("Ex 1: ",example)

        if isinstance(example,np.ndarray):
            example=example[0]
            #print("is instance executed")

        #print("Ex 2: ",example)
        #print("dict indx:",dict_index)

        for token_word in example.split(): #for every word in preprocessed example
            self.bow_dicts[dict_index][token_word]+=1 #increment in its count


    def train(self,dataset,labels):
        self.examples=dataset
        self.labels=labels
        self.bow_dicts=np.array([defaultdict(lambda:0) for index in range(self.classes
        print("Init Bow Dict",self.bow_dicts)

        if not isinstance(self.examples,np.ndarray): self.examples=np.array(self.examp
        if not isinstance(self.labels,np.ndarray): self.labels=np.array(self.labels)

        #constructing BoW for each category
        #print(self.labels==0)
        for cat_index,cat in enumerate(self.classes):
            all_cat_examples=self.examples[self.labels==cat]


            cleaned_examples=[preprocess_string(cat_example) for cat_example in all_cat
            #print("Cleaned Ex 1: ",cleaned_examples)
            #print("Cleaned Ex 1 type: ",type(cleaned_examples))

            cleaned_examples=pd.DataFrame(data=cleaned_examples)
```

```
        #print("Cleaned Ex 2: ",cleaned_examples)
        #print("Cleaned Ex 2 type: ",type(cleaned_examples))

        #now costruct BoW of this particular category
        np.apply_along_axis(self.addToBow,1,cleaned_examples,cat_index)

        prob_classes=np.empty(self.classes.shape[0])
        all_words=[]
        cat_word_counts=np.empty(self.classes.shape[0])

        for cat_index,cat in enumerate(self.classes):
            #Calculating prior probability p(c) for each class
            prob_classes[cat_index]=np.sum(self.labels==cat)/float(self.labels.shap

            #Calculating total counts of all the words of each class
            count=list(self.bow_dicts[cat_index].values())
            cat_word_counts[cat_index]=np.sum(np.array(count))+1 # |v| is remaining

            #get all words of this category
            all_words+=self.bow_dicts[cat_index].keys()

        #combine all words of every category & make them unique to get vocabulary

        self.vocab=np.unique(np.array(all_words))
        self.vocab_length=self.vocab.shape[0]

        #computing denominator value
        denoms=np.array([cat_word_counts[cat_index]+self.vocab_length+1 for cat_ind


        '''
        Now that we have everything precomputed as well, its better to organize eve
        rather than to have a separate list for every thing.

        Every element of self.cats_info has a tuple of values
        Each tuple has a dict at index 0, prior probability at index 1, denominato
        '''

        self.cats_info=[(self.bow_dicts[cat_index],prob_classes[cat_index],denoms[
        self.cats_info=np.array(self.cats_info)


    def getExampleProb(self,test_example):
```

```
        likelihood_prob=np.zeros(self.classes.shape[0]) #to store probability w.r.t ea

        #finding probability w.r.t each class of the given test example
        for cat_index,cat in enumerate(self.classes):

            for test_token in test_example.split(): #split the test example and get p

                #get total count of this test token from it's respective training dict
                test_token_counts=self.cats_info[cat_index][0].get(test_token,0)+1

                #now get likelihood of this test_token word
                test_token_prob=test_token_counts/float(self.cats_info[cat_index][2])

                #remember why taking log? To prevent underflow!
                likelihood_prob[cat_index]+=np.log(test_token_prob)

        # we have likelihood estimate of the given example against every class but we
        post_prob=np.empty(self.classes.shape[0])
        for cat_index,cat in enumerate(self.classes):
            post_prob[cat_index]=likelihood_prob[cat_index]+np.log(self.cats_info[cat_

        return post_prob


    def test(self,test_set):

        predictions=[] #to store prediction of each test example
        for example in test_set:

            #preprocess the test example the same way we did for training set exampels
            cleaned_example=preprocess_string(example)

            #simply get the posterior probability of every example
            post_prob=self.getExampleProb(cleaned_example) #get prob of this example fo

            #simply pick the max value and map against self.classes!
            predictions.append(self.classes[np.argmax(post_prob)])

        return np.array(predictions)
```

In [50]:

```
nb=NaiveBayes(np.unique(Y_Train))
```

In [51]:

```
nb.train(X_Train,Y_Train)
```

```
 Init Bow Dict [defaultdict(<function NaiveBayes.train.<locals>.<listcomp>.<lambda> at 0x00000104F2ED0310>, {})
  defaultdict(<function NaiveBayes.train.<locals>.<listcomp>.<lambda> at 0x00000104F2ED01F0>, {})]
```

In [52]:

```
Test.head(n=5)
```

	review
0	Remember those old kung fu movies we used to w...
1	This movie is another one on my List of Movies...
2	How in the world does a thing like this get in...
3	"Queen of the Damned" is one of the best vampi...
4	The Caprica episode (S01E01) is well done as a...

In [53]:

```
Test = Test.values
X_Test = Test[:,0]
print(X_Test.shape)
```

```
 (10000,)
```

In [54]:

```
Y_Pred = nb.test(X_Test)
```

In [57]:

```
print(Y_Pred[:5])
print(Y_Pred.shape)
```

```
 ['neg' 'neg' 'neg' 'pos' 'pos']
 (10000,)
```

In [56]:

```
# Saving File
df = pd.DataFrame(data=Y_Pred,columns=["label"])
df.to_csv("Movie_Y_Pred.csv")
```

In [15]:

```
print(Y_Pred.shape)
```

```
(10000,)
```

In [16]:

```
# Calculating accuracy on training data by splitting it
from sklearn.model_selection import train_test_split
```

In [17]:

```
x_train, x_test, y_train, y_test = train_test_split(X_Train, Y_Train, test_size=0.33)
```

In [18]:

```
print(x_train.shape,y_train.shape)
print(x_test.shape,y_test.shape)
```

```
(670,) (670,)
(330,) (330,)
```

In [19]:

```
nb=NaiveBayes(np.unique(y_train))
nb.train(x_train,y_train)
```

```
Init Bow Dict [defaultdict(<function NaiveBayes.train.<locals>.<listcomp>.<lambda> at 0x00000104E4B82820>, {})
 defaultdict(<function NaiveBayes.train.<locals>.<listcomp>.<lambda> at 0x00000104DCAE4160>, {})]
```

In [20]:

```
y_pred = nb.test(x_test)
print(y_pred[:5])
```

```
['pos' 'pos' 'neg' 'pos' 'neg']
```

In [21]:

```
acc = np.sum(y_pred==y_test)/y_test.shape[0]
print(acc)
```

```
0.8272727272727273
```

In [22]:

```
from sklearn.naive_bayes import MultinomialNB, BernoulliNB, GaussianNB
import clean_review  as cr
from sklearn.feature_extraction.text import CountVectorizer
```

In [23]:

```
x_clean = [cr.getCleanReview(i) for i in x_train]
xt_clean = [cr.getCleanReview(i) for i in x_test]
```

In [26]:

```
print(x_clean[:5])
```

```
['watch show cousin hate first girl dress style cloth first letter name come could better villain spare first r
gay version devil pink hillbilli gang green gang whit iron name spoil princess iron name among other also found
h rather watch sailor moon much better someon els want watch show room find way break televis believ save half
v whatev watch', 'twelv monkey got element becom terri gilliam masterpiec outstand screenplay sustain rhythm cl
reov good nose cast twelv monkey also first movi bruce willi stand back kind charact use play previou movi jade
knam prison took fearless invinc hero case die hard matter tri prison time movi contain thrill end got real dra
reflect man danger dread notabl one could caus end world viru creat ill matter long take twelv monkey estim tru
nineti', 'white chick hold dress black chick oh yeah look differ anyon give one wayan movi dress ladi menac cou
nza ghost wrote norton trio member act director white chick never realli joke wayan act like girl hour setup pu
a play time crisi least time exact somebodi tell kenan ivori damon marlon shawn damien talent one kim rakeesha
p make movi hurt zone layer verdict', 'rocket govern experi effect cosmic ray anim crash small texa town peopl
i investig hamper effort govern offici turn mutant space gorilla loos kill teenag wood like low budget scienc t
ster movi thought would good chanc would like movi sadli mind bad act corni dialog atroci music score giant plo
m seen enjoy bad good kind way other type night fright differ night fright terribl pace drag scene go without a
e peopl walk forest long time sever seemingli endless danc teen parti wood noth interest go scene shorter movi
pli cut scene would save one given movi three view make sure gave chanc slam review sadli gotten wors watch fou
ewbal like comic strip drama farc set franc implos play wide eye straight face intens talent cast chockablock a
ri authent period detail slick brillantin hairdo marcel hairdo fleet citroen traction rollick soundtrack brief
aull marshal petain simpli best entertain recent shown screen devoid presumpt messag movi train creativ recreat
rk despit steam locomot expens prop doubt would tgv']
```

In [38]:

```
cv = CountVectorizer()

# Vectorization on train set
x_vec = cv.fit_transform(x_clean).toarray()
xt_vec = cv.transform(xt_clean).toarray()
print(x_vec.shape)
```

```
(670, 10097)
```

In [34]:

```
mnb = MultinomialNB()
```

In [35]:

```
mnb.fit(x_vec,y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [39]:

```
mnb.score(xt_vec,y_test)
```

```
0.8393939393939394
```

In [42]:

```
bnb = BernoulliNB()
bnb.fit(x_vec,y_train)
```

```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

In [43]:

```
bnb.score(xt_vec,y_test)
```

```
0.8393939393939394
```

In [ ]: