* I have attached Two CSV files and python script file
  + Reviews\_40.csv (Training Set)
  + Tenreviews.csv
  + NaveBayesAmazonReviews.py
* I have used DataFrames of pandas to operate on the CSV file
* Here is the google drive link of my files

<https://drive.google.com/file/d/1BTHQ77BwSWeIiF0PCyzC14guNFvvoiHb/view?usp=sharing>

My python code: NaiveBayesReviews.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Sep 7 14:03:42 2019

@author: Naveen Kumar

"""

from collections import OrderedDict #Ordered Dictionary to store probability of each feature

import pandas as pd

import string

# reviews cleaning

from nltk.corpus import stopwords

STOP\_WORDS = set(stopwords.words('english'))

STOP\_WORDS.add('')

#method to clean reviews

def clean\_review(review):

exclude = set(string.punctuation)

review = ''.join(ch for ch in review if ch not in exclude)

split\_sentence = review.lower().split(" ")

clean = [word for word in split\_sentence if word not in STOP\_WORDS] #removal of stop words

return clean

#data frame to store 40 Reviews Training data

df=pd.read\_csv("C:\\Users\\LENOVO\\Desktop\\Reviews40.csv")

new\_df=df.fillna('not available') #filling of NaN with not available

#features list of 10 keywords or key phrases indicative of buyer’s attitude

features=['excellent','good','comfortable','recommend','perfect','bad','problem','terrible','issues','noise']

#cleaning of reviews with clean\_review method

reviews\_list=[rev for rev in new\_df['Review']]

clean\_40review\_list=[]

for rl in reviews\_list:

clean\_40review\_list.append(clean\_review(rl))

#frequency of each feature in reviews

for f in features:

feature\_count=[]

for review in clean\_40review\_list:

n=review.count(f)

if n>=1:

n=1

feature\_count.append(n)

new\_df[f]= feature\_count #data frame with frequency of each feature in reviews

# listing the positive review indexes to find the corresponding feature count in each review

positive\_review=new\_df['feedback'].unique()[0]

p=new\_df['feedback']==positive\_review

positive\_df=new\_df[p]

positive\_index=list(positive\_df.index) #positive review indexes

pos\_review\_count=len(positive\_index) #positive review count in 40 reviews

# listing the negative review indexes to find the corresponding feature count in each review

negative\_review=new\_df['feedback'].unique()[1]

n=new\_df['feedback']==negative\_review

negative\_df=new\_df[n]

negative\_index=list(negative\_df.index) #negative review indexes

neg\_review\_count=len(negative\_index) #negative review count in 40 reviews

print(new\_df)

#Each feature count in positive reviews

feature\_count\_pos=[]

for f1 in features:

f\_occurance\_p=[]

for pi in positive\_index:

f\_occurance\_p.append(new\_df.at[pi,f1]) #[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]

each\_feat\_count\_p=f\_occurance\_p.count(1)

feature\_count\_pos.append(each\_feat\_count\_p)

print(feature\_count\_pos) #count of each positive feature[12, 12, 9, 7, 4, 1, 4, 1, 6, 7]

feature\_count\_neg=[]

for f1 in features:

f\_occurance\_n=[]

for ni in negative\_index:

f\_occurance\_n.append(new\_df.at[ni,f1])

each\_neg\_feat\_count=f\_occurance\_n.count(1)

feature\_count\_neg.append(each\_neg\_feat\_count) #count of each negative feature [1, 8, 2, 2, 1, 14, 8, 0, 8, 1]

print(feature\_count\_neg) #[0, 0, 0, 0, 0, 0, 0, 10, 0, 0]

############probability of each feature with feedback positive and negative : 𝑝 (𝑥1/Ci),,,,,,,,𝑝 (𝑥1/Ci)

prob\_pos=[]

for x in feature\_count\_pos:

prob\_pos.append(x/pos\_review\_count)

print(prob\_pos) #positive probabilities of each feature[0.6, 0.6, 0.45, 0.35, 0.2, 0.05, 0.2, 0.05, 0.3, 0.35]

prob\_neg=[]

for y in feature\_count\_neg:

prob\_neg.append(y/neg\_review\_count)

print(prob\_neg) #negative probabilities of each feature[0.05, 0.4, 0.1, 0.1, 0.05, 0.7, 0.4, 0.0, 0.4, 0.05]

################ Dictionary with (key:value) as (feature,probability)

class NaveBayesClassifier\_Positive(OrderedDict): #customization the dictionary output

def \_\_missing\_\_(self,k):

self[k]=' '

return self[k]

positive\_dic=NaveBayesClassifier\_Positive()

i=0

for epf in features:

positive\_dic[epf]=prob\_pos[i]

i=i+1

print(positive\_dic)

#NaveBayesClassifier\_Positive([('excellent', 0.6), ('good', 0.6), ('comfortable', 0.45), ('recommend', 0.35),

#('perfect', 0.2), ('bad', 0.05), ('problem', 0.2), ('terrible', 0.05), ('issues', 0.3), ('noise', 0.35)])

class NaveBayesClassifier\_Negative(OrderedDict):

def \_\_missing\_\_(self,k):

self[k]=' '

return self[k]

negative\_dic=NaveBayesClassifier\_Negative()

j=0

print(prob\_neg[0])

for enf in features:

negative\_dic[enf]=prob\_neg[j]

j=j+1

print(negative\_dic)

#NaveBayesClassifier\_Negative([('excellent', 0.05), ('good', 0.4), ('comfortable', 0.1), ('recommend', 0.1),

#('perfect', 0.05), ('bad', 0.7), ('problem', 0.4), ('terrible', 0.0), ('issues', 0.4), ('noise', 0.05)])

############Validation with TEN reviews################

#data frame with 10 Reviews data

tenreviewsread\_df=pd.read\_csv("C:\\Users\\LENOVO\\Desktop\\TenReviews.csv")

tenreviews\_df=tenreviewsread\_df.fillna('not available')

#print(tenreviews\_df)

list\_rev=[r for r in tenreviews\_df['Reviews10']]

clean\_10review\_list=[]

for lr in list\_rev:

clean\_10review\_list.append(clean\_review(lr))

pb\_pos\_review=pos\_review\_count/(neg\_review\_count+pos\_review\_count) #probability that a review is positive of 40 reviews

pb\_neg\_review=neg\_review\_count/(neg\_review\_count+pos\_review\_count) #probability that a review is negative of 40 reviews

##### positive probabilities of ten reviews

pb\_pos=[]

for er in clean\_10review\_list:

m=1

for f in features:

n=er.count(f)

if n>=1:

m=m\*float(positive\_dic[f])

pb\_pos.append(m\*pb\_pos\_review)

tenreviews\_df['Postive\_pb']=pb\_pos

#### negative probabilities of ten reviews

pb\_neg=[]

for ern in clean\_10review\_list:

m1=1

for f1 in features:

n1=ern.count(f1)

if n1>=1:

m1=m1\*float(negative\_dic[f1])

pb\_neg.append(m1\*pb\_neg\_review)

tenreviews\_df['Negative\_pb']=pb\_neg

### Classification of Feedbacks based on probabilities

u=0

validated\_feedback=[]

for er in clean\_10review\_list:

if (float(pb\_pos[u]) > float(pb\_neg[u])):

validated\_feedback.append('positive')

else:

validated\_feedback.append('negative')

u=u+1

tenreviews\_df['ValidatedFeedback']=validated\_feedback

actual\_feedback=tenreviews\_df['ActualFeedback']

print(tenreviews\_df)

##### Accuracy of classifier

v=0

pr=0

for er in clean\_10review\_list:

if validated\_feedback[v] == actual\_feedback[v]: #comparison of positive and negative feedback in actual and validated feedbacks

pr=pr+1

v=v+1

print('Accuracy of the classifier:',pr/10)