LAB CYCLE 3

Experiment No:3
Date:08/01/22

Aim:

Write a program to learn a naïve Bayes classifier and use it to predict class labels of testdata. Laplacian smoothing should be used. The learned classifier should be tested on test instances and the accuracy of prediction for the test instances should be printed as output. A single program should train the classifier on the training set as well as test it on the test set.

Data Set Description:

The task is to predict whether a citizen is happy to live in a city based on certain parametersof the city as rated by the citizens in a scale of 1-5 during a survey.

Attribute Information:

D = decision/class attribute (D) with values 0 (unhappy) and 1 (happy)

(Column 1 of file)X1 = the availability of information about the city services

(Column 2 of file)

X2 = the cost of housing

X3 = the overall quality of public

schoolsX4 = your trust in the local

police

X5 = the maintenance of streets and

sidewalks X6 = the availability of social

community eventsAttributes X1 to X6

have values 1 to 5.

Training Data Filename: data3.csv, Test Data Filename: test3.csv .

Source Code:

In [1]:
```
#import packages
```

In [2]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [3]:
```
#read data
df = pd.read_csv('data3.csv')
```

In [4]:
```
df
```

Out[4]:

	D	X1	X2	X3	X4	X5	X6
0	0	3	3	3	4	2	4
1	0	3	2	3	5	4	3
2	1	5	3	3	3	3	5
3	0	5	4	3	3	3	5
4	0	5	4	3	3	3	5
...	...	...	...	...	...	...	...
124	1	5	2	4	4	2	3
125	0	5	3	3	4	4	5
126	0	5	3	3	4	4	4
127	0	3	2	3	3	5	4
128	0	4	1	3	3	3	4

129 rows × 7 columns

In [5]:
```
total=df.shape[0]
total
```

Out[5]:
```
129
```

In [6]:
```
fzero=df['D'][df['D'] == 0].count()
fzero
```

Out[6]:
```
59
```

In [7]:
```
fone=df['D'][df['D']==1].count()
fone
```

Out[7]:
```
70
```

In [8]:
```
pzero=fzero/total
pzero
```

Out[8]:
```
0.4573643410852713
```

In [9]:
```
pone=fone/total
pone
```

Out[9]:
```
0.5426356589147286
```

In [10]:
```
czero=np.zeros((5,6))
czero
```

Out[10]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

In [11]:
```
cone=np.zeros((5,6))
cone
```

Out[11]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

In [12]:
```
df['X1'][df['D']==1][df['X1']==3].count()
```

Out[12]:
```
7
```

In [13]:
```
df.X1.value_counts()
```

Out[13]:
```
5    62
4    43
3    23
1     1
Name: X1, dtype: int64
```

In [14]:
```
df.X2.value_counts()
```

Out[14]:
```
3    40
2    37
1    28
4    18
5     6
Name: X2, dtype: int64
```

In [15]:
```
df.X3.value_counts()
```

Out[15]:
```
3    59
4    33
2    16
5    15
1     6
Name: X3, dtype: int64
```

In [16]:
```
4    51
3    48
5    23
2     6
1     1
Name: X4, dtype: int64
```

Out[16]:

```
df.X5.value_counts()
```

In [17]:
```
4    52
5    28
3    26
2    17
1     6
Name: X5, dtype: int64
```

Out[17]:

```
df.X4.value_counts()
```

In [18]:
```
df.X6.value_counts()
```

Out[18]:
```
5    54
4    52
3    21
2     1
1     1
Name: X6, dtype: int64
```

In [ ]:

In [19]:
```
'X'+str(1)
```

Out[19]:
```
'X1'
```

In [20]:
```
for i in range(0,6):
    for j in range(0,5):
        czero[j][i]=df['X'+str(i+1)][df['D']==0][df['X'+str(i+1)]==j+1].count()
czero
```

Out[20]:
```
array([[ 1., 15.,  4.,  0.,  5.,  1.],
       [ 0., 14.,  7.,  5.,  9.,  1.],
       [16., 18., 34., 24., 15., 16.],
       [25., 10.,  9., 21., 22., 24.],
       [17.,  2.,  5.,  9.,  8., 17.]])
```

In [21]:
```
for i in range(0,6):
    for j in range(0,5):
        cone[j][i]=df['X' +str(i+1)][df['D']==1][df['X' +str(i+1)]==j+1].count()
```

In [22]:
```
Cone
```

Out[22]
```
array([[ 0., 13.,  2.,  1.,  1.,  0.],
       [ 0., 23.,  9.,  1.,  8.,  0.],
       [ 7., 22., 25., 24., 11.,  5.],
       [18.,  8., 24., 30., 30., 28.],
       [45.,  4., 10., 14., 20., 37.]])
```

In [23]:
```
zeroprobzero=np.zeros((6))
oneprobzero=np.zeros((6))
```

In [24]:
```
for i in range(0,5):
    for j in range(0,6):
        if(czero[i][j]==0):
            zeroprobzero[j]=zeroprobzero[j]+1
zeroprobzero
```

Out[24]:
```
array([1., 0., 0., 1., 0., 0.])
```

In [25]:
```
for i in range(0,5):
    for j in range(0,6):
        if(cone[i][j]==0):
            oneprobzero[j]=oneprobzero[j]+1
oneprobzero
```

Out[25]:
```
array([2., 0., 0., 0., 0., 2.])
```

In [26]:
```
czeroprob=np.zeros((5,6))
```

In [27]:
```
coneprob=np.zeros((5,6))
cone
```

Out[27]:
```
array([[ 0., 13.,  2.,  1.,  1.,  0.],
       [ 0., 23.,  9.,  1.,  8.,  0.],
       [ 7., 22., 25., 24., 11.,  5.],
       [18.,  8., 24., 30., 30., 28.],
       [45.,  4., 10., 14., 20., 37.]])
```

In [28]:
```
for n in range(0,6):
    if(zeroprobzero[n]>0):
        for j in range(0,5):
            czeroprob[j][n]=(czero[j][n]+1)/(fzero+5)
    else:
        for j in range(0,5):
            czeroprob[j][n]=(czero[j][n]/fzero)

czeroprob
```

Out[28]:
```
array([[0.03125   , 0.25423729, 0.06779661, 0.015625  , 0.08474576,
        0.01694915],
       [0.015625  , 0.23728814, 0.11864407, 0.09375   , 0.15254237,
        0.01694915],
       [0.265625  , 0.30508475, 0.57627119, 0.390625  , 0.25423729,
        0.27118644],
       [0.40625   , 0.16949153, 0.15254237, 0.34375   , 0.37288136,
        0.40677966],
       [0.28125   , 0.03389831, 0.08474576, 0.15625   , 0.13559322,
        0.28813559]])
```

In [29]:
```
for n in range(0,6):
    if(oneprobzero[n]>0):
        for j in range(0,5):
            coneprob[j][n]=(cone[j][n]+1)/(fone+5)
    else:
        for j in range(0,5):
```

```
                coneprob[j][n]=(cone[j][n]/fone)

coneprob
```

Out[29]:
```
array([[0.01333333, 0.18571429, 0.02857143, 0.01428571, 0.01428571,
        0.01333333],
       [0.01333333, 0.32857143, 0.12857143, 0.01428571, 0.11428571,
        0.01333333],
       [0.10666667, 0.31428571, 0.35714286, 0.34285714, 0.15714286,
        0.08      ],
       [0.25333333, 0.11428571, 0.34285714, 0.42857143, 0.42857143,
        0.38666667],
       [0.61333333, 0.05714286, 0.14285714, 0.2       , 0.28571429,
        0.50666667]])
```

In [ ]:

In [ ]:

In [30]:
```
#Testphase,prepare confusion matrix and find precision,recall,accuracy and error rat
```

In [ ]:

In [31]:
```
tf=pd.read_csv("test3.csv")
```

In [32]:
```
tf
```

Out[32]:

	D	X1	X2	X3	X4	X5	X6
0	0	5	1	4	4	4	5
1	0	5	2	2	4	4	5
2	0	5	3	5	4	5	5
3	1	3	4	4	5	1	3
4	1	5	1	5	5	5	5
5	1	4	3	3	4	4	4
6	1	5	5	1	1	5	1
7	0	4	4	4	4	1	3
8	1	5	2	3	4	4	3
9	0	5	3	3	1	3	5
10	1	5	2	3	4	2	5
11	1	5	3	3	4	4	5
12	0	4	3	3	4	4	5
13	0	5	3	2	5	5	5

In [33]:
```
ttotal=tf.shape[0]
```

```
Ttotal
```

Out[33]:
```
14
```

In [34]:
```
#Confusion matrix
```

In [35]:
```
tp=0
tn=0
fp=0
fn=0
for n in range(0,ttotal):
    a=1
    b=1
    for i in range(1,6):
        k=tf.at[n,'X'+str(i)]
        a=a*czeroprob[k-1][i-1]
        b=b*coneprob[k-1][i-1]
        if i==5:
            break
    a=a*pzero
    b=b*pone

    if (a>b):
        predict=0
    else:
        predict=1
        #print(d)
    d=tf.at[n,'D']
    if(d==1 and predict==1):
        tp=tp+1
    elif(d==1 and predict==0):
        tn=tn+1
    elif(d==0 and predict==1):
        fp=fp+1
    else:
        fn=fn+1

            #print("tp=",tp)
            #print("tn=",tn)
            #print("fp=",fp)
            #print("fn=",fn)
```

In [36]:
```
confusion=np.array([[tp,fp],[fn,tn]])
confusion
```

Out[36]:
```
array([[5, 4],
       [3, 2]])
```

In [37]:
```
#Accuracy
accuracy=(tp+tn)/(tp+tn+fp+fn)
print("accuracy = ",accuracy)
```

```
accuracy =  0.5
```

In [38]:
```
#Error rate
error_rate=1-accuracy
print("error_rate =",error_rate)
```

```
error_rate = 0.5
```

In [39]:

```
#Precision
precision=tp/(tp+fp)
print("precision= ",precision)
```

```
precision= 0.5555555555555556
```

In [40]:

```
#Recall
recall=tp/(tp+fn)
print("recall= ",recall)
```

```
recall=  0.625
```