LAB CYCLE 3

Date : 23/01/2021
AIM:

Problem Statement:

Write a program to learn a naïve Bayes classifier and use it to predict class labels of test data. 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.

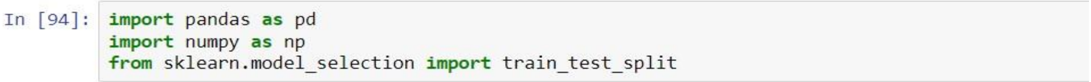
Data Set Description:

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

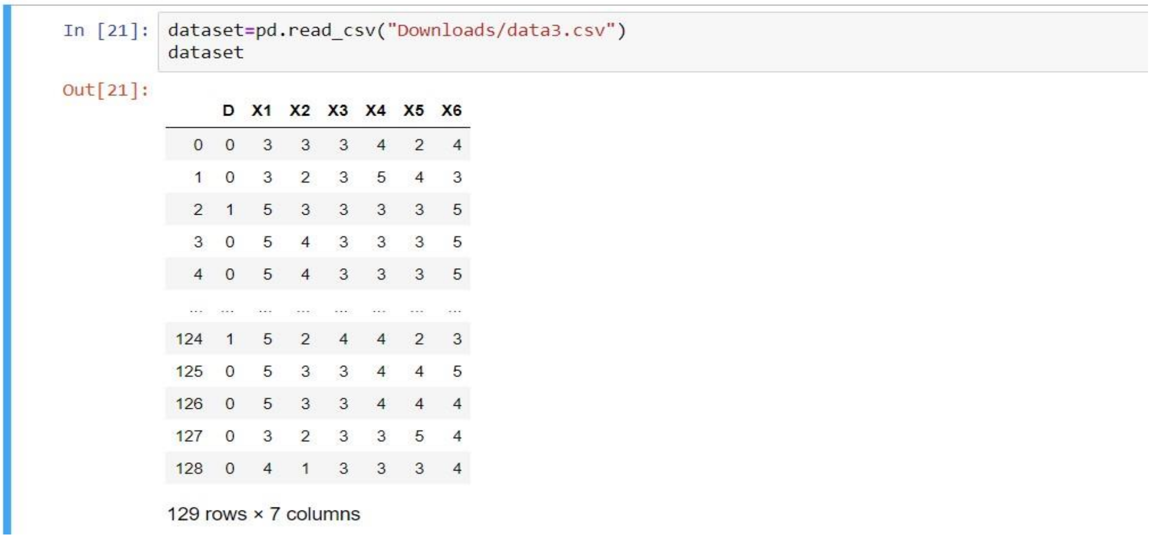
Attribute Information:

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 schools
X4 = your trust in the local police
X5 = the maintenance of streets and sidewalks
X6 = the availability of social community events
Attributes X1 to X6 have values 1 to 5.

SOURCE CODE



```
In [94]: import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
```

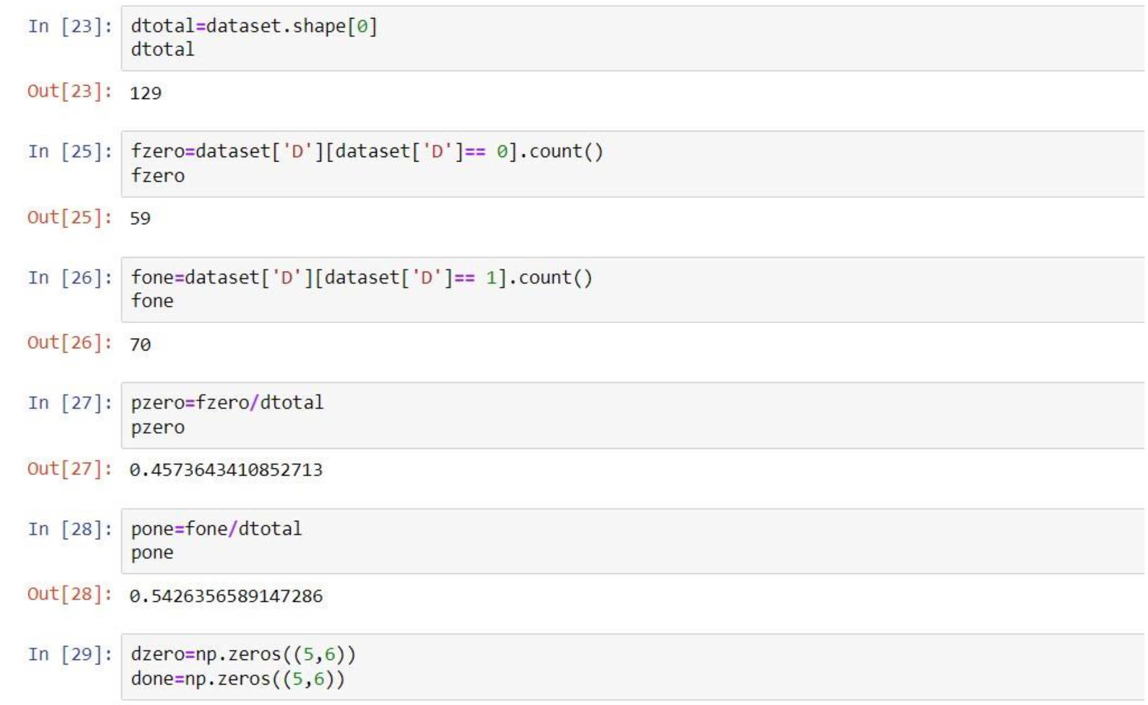


```
In [21]: dataset=pd.read_csv("Downloads/data3.csv")
         dataset
```

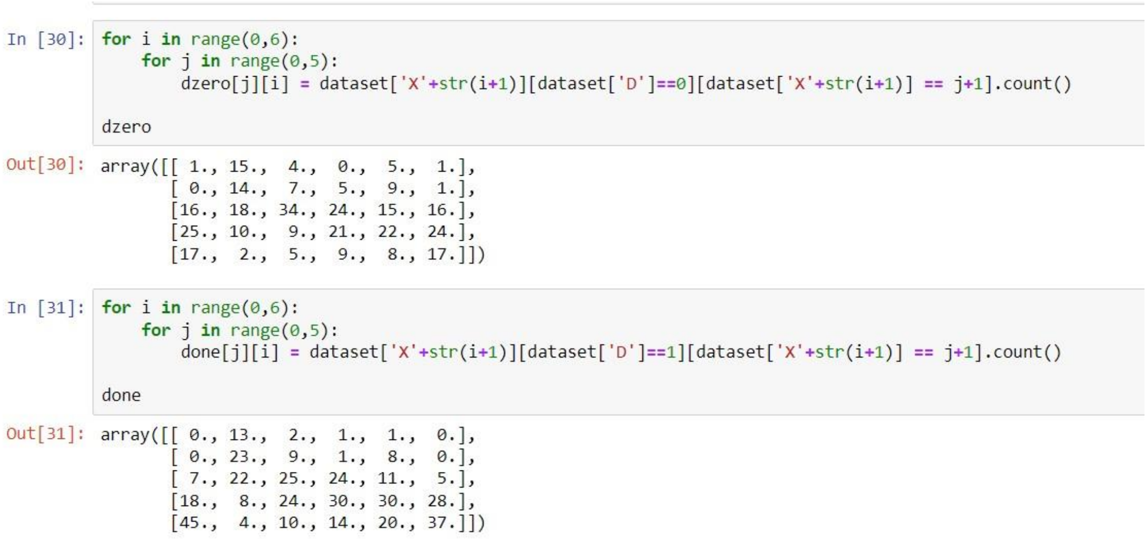
Out[21]:

	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

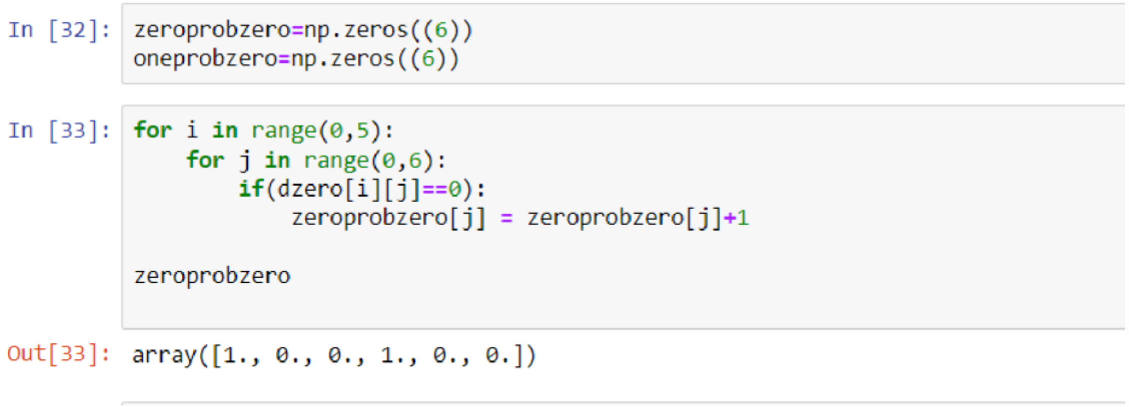
129 rows × 7 columns



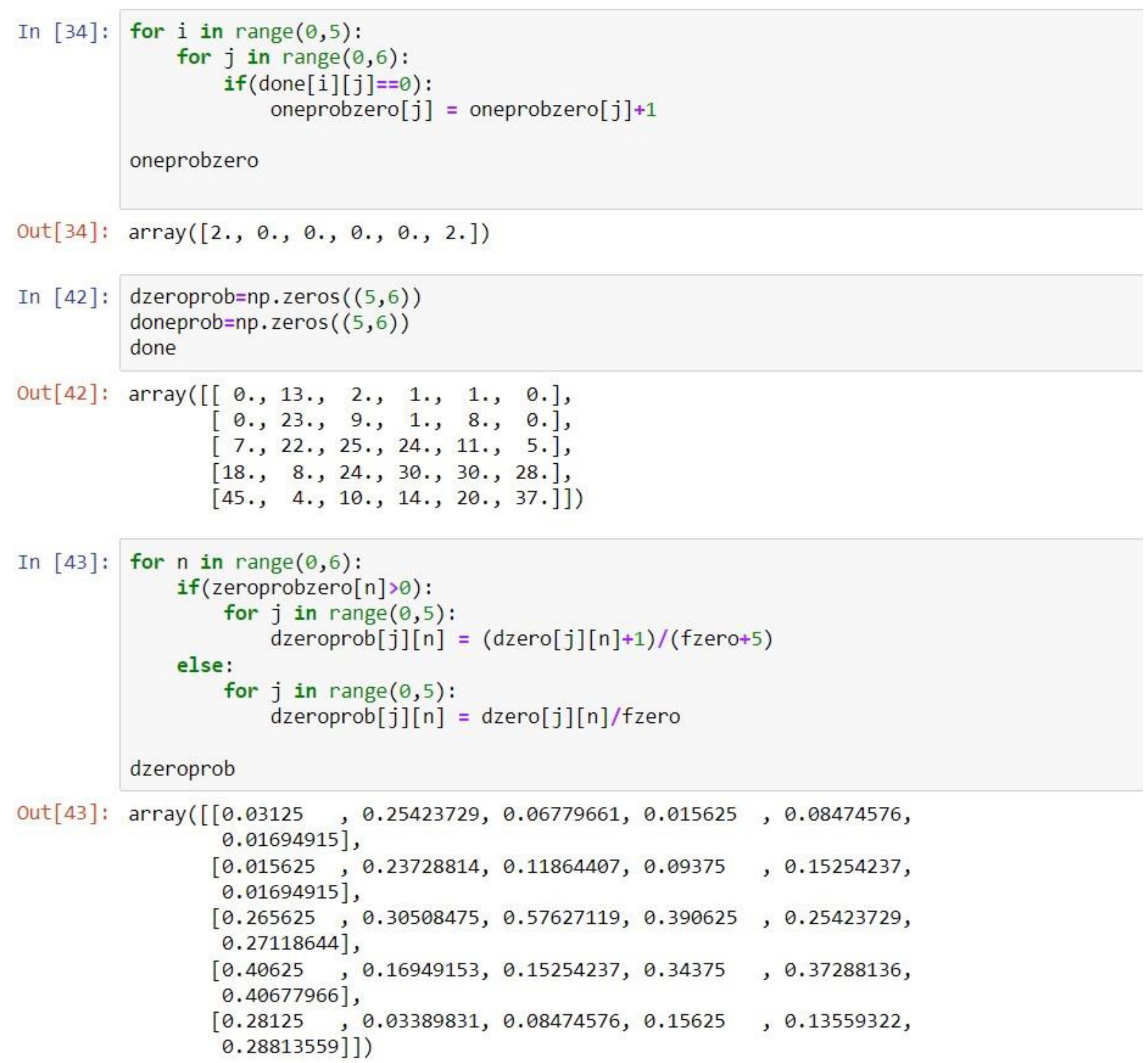
```
In [23]: dtotal=dataset.shape[0]
         dtotal

Out[23]: 129

In [25]: fzero=dataset['D'][dataset['D']== 0].count()
         fzero

Out[25]: 59

In [26]: fone=dataset['D'][dataset['D']== 1].count()
         fone

Out[26]: 70

In [27]: pzero=fzero/dtotal
         pzero

Out[27]: 0.4573643410852713

In [28]: pone=fone/dtotal
         pone

Out[28]: 0.5426356589147286

In [29]: dzero=np.zeros((5,6))
         done=np.zeros((5,6))
```



```
In [30]: for i in range(0,6):
             for j in range(0,5):
                 dzero[j][i] = dataset['X'+str(i+1)][dataset['D']==0][dataset['X'+str(i+1)] == j+1].count()

         dzero

Out[30]: 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 [31]: for i in range(0,6):
             for j in range(0,5):
                 done[j][i] = dataset['X'+str(i+1)][dataset['D']==1][dataset['X'+str(i+1)] == j+1].count()

         done

Out[31]: 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.]])
```



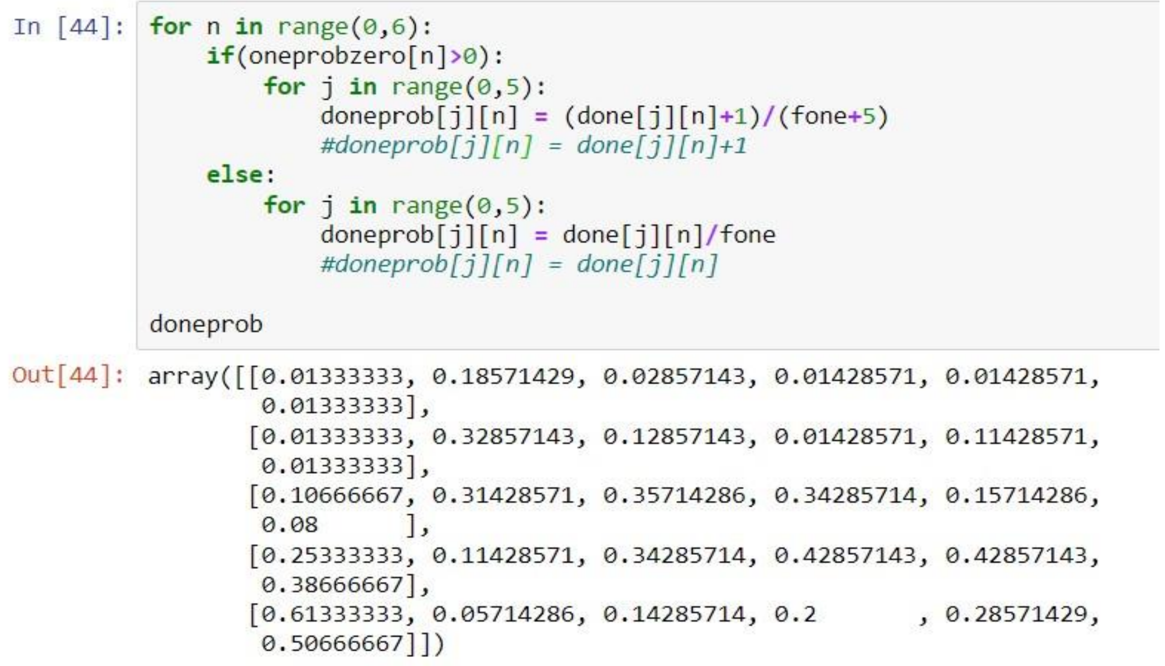
```
In [32]: zeroprobzero=np.zeros((6))
         oneprobzero=np.zeros((6))

In [33]: for i in range(0,5):
             for j in range(0,6):
                 if(dzero[i][j]==0):
                     zeroprobzero[j] = zeroprobzero[j]+1

         zeroprobzero

Out[33]: array([1., 0., 0., 1., 0., 0.])
```



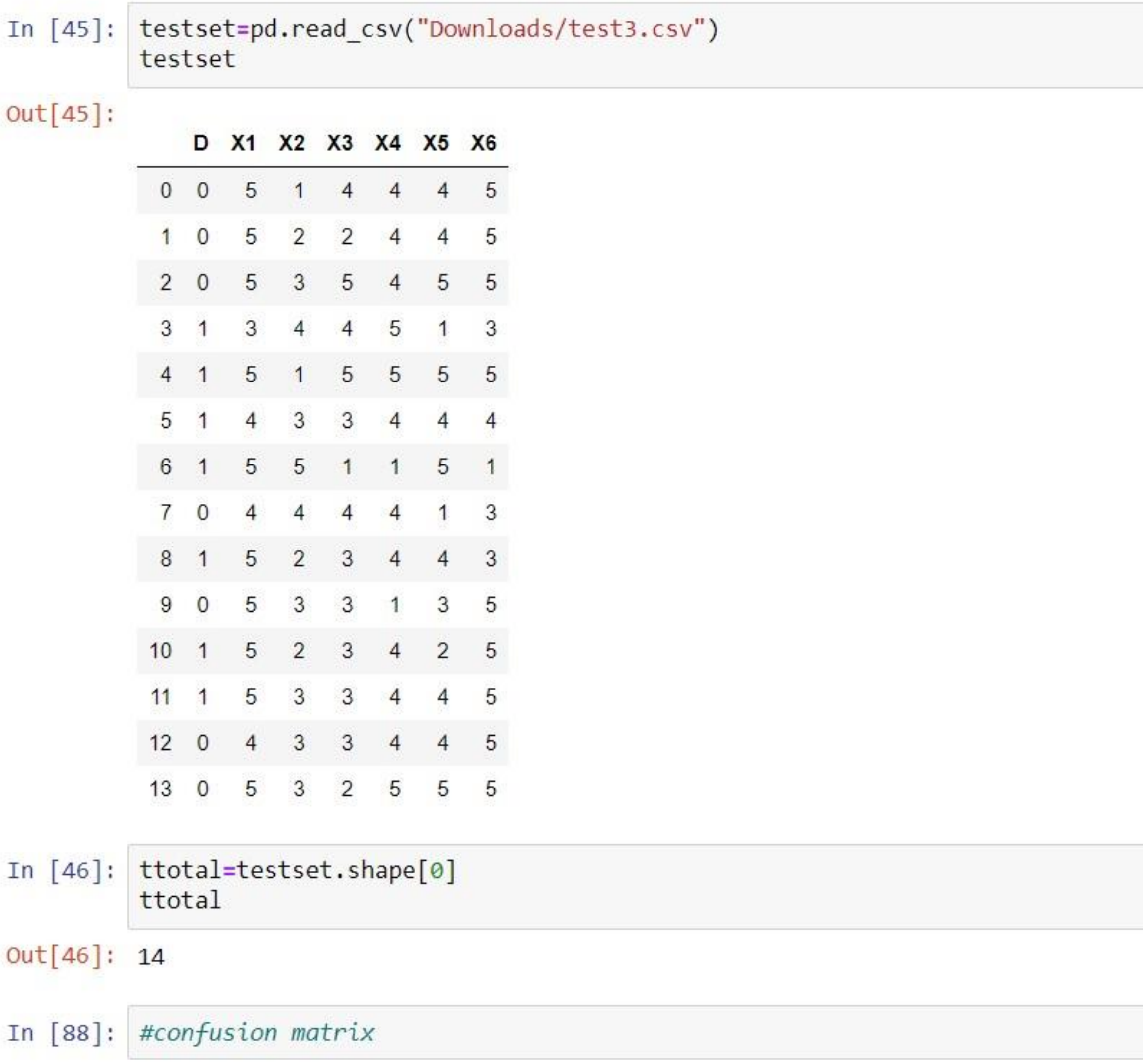
```
In [34]: for i in range(0,5):
             for j in range(0,6):
                 if(done[i][j]==0):
                     oneprobzero[j] = oneprobzero[j]+1

         oneprobzero

Out[34]: array([2., 0., 0., 0., 0., 2.])
```

```
In [42]: dzeroprob=np.zeros((5,6))
         doneprob=np.zeros((5,6))
         done

Out[42]: 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 [43]: for n in range(0,6):
             if(zeroprobzero[n]>0):
                 for j in range(0,5):
                     dzeroprob[j][n] = (dzero[j][n]+1)/(fzero+5)
             else:
                 for j in range(0,5):
                     dzeroprob[j][n] = dzero[j][n]/fzero

         dzeroprob

Out[43]: 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 [44]: for n in range(0,6):
             if(oneprobzero[n]>0):
                 for j in range(0,5):
                     doneprob[j][n] = (done[j][n]+1)/(fone+5)
                     #doneprob[j][n] = done[j][n]+1
             else:
                 for j in range(0,5):
                     doneprob[j][n] = done[j][n]/fone
                     #doneprob[j][n] = done[j][n]

         doneprob

Out[44]: 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]])
```

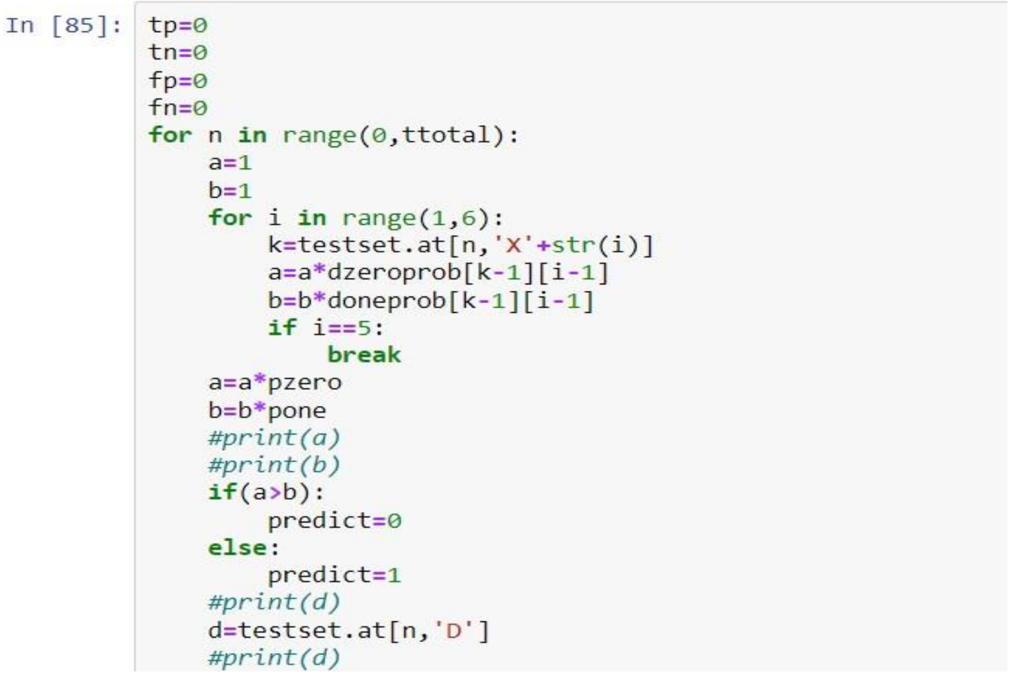


```
In [45]: testset=pd.read_csv("Downloads/test3.csv")
         testset
```

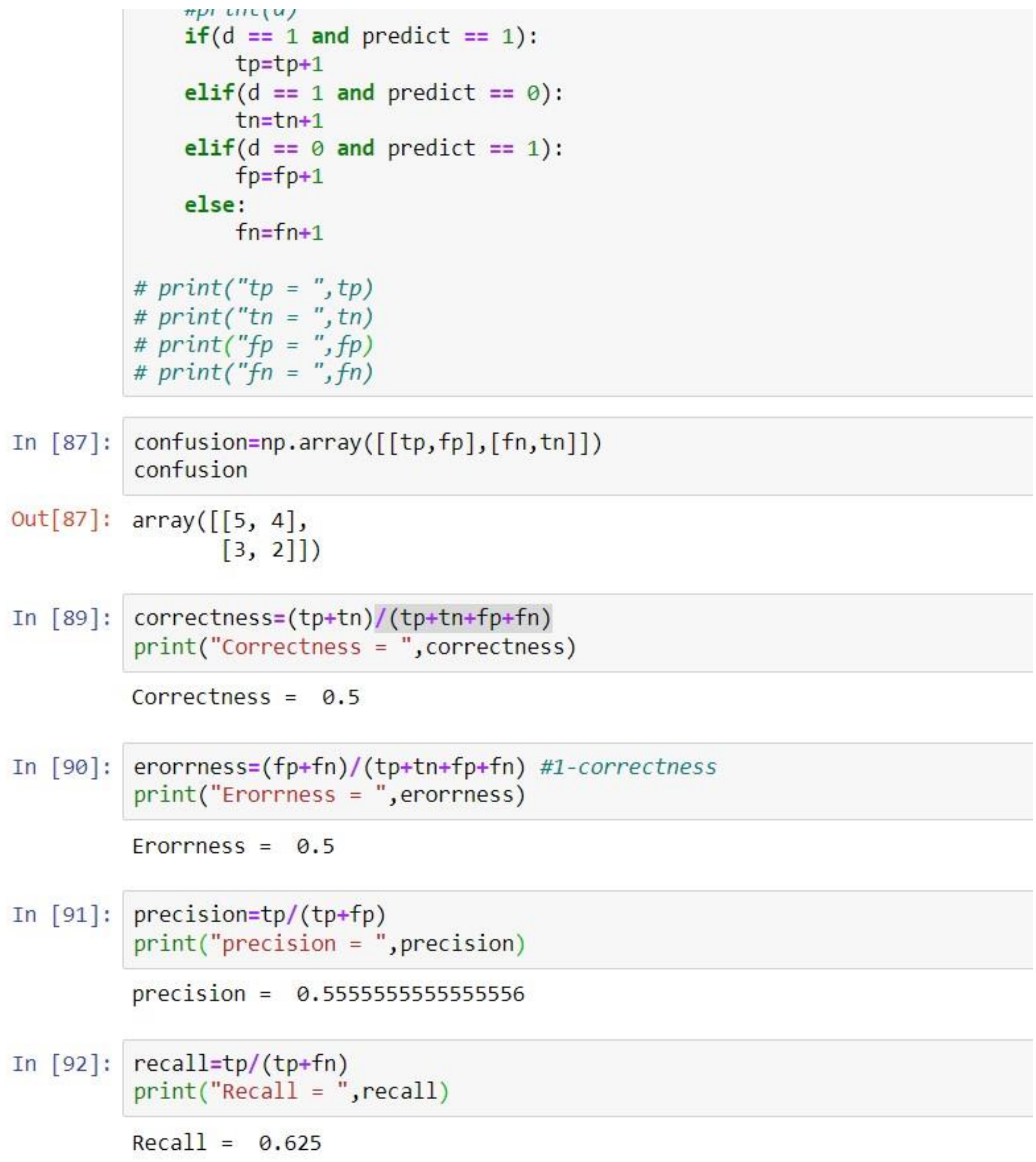
Out[45]:

	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 [46]: ttotal=testset.shape[0]
         ttotal

Out[46]: 14

In [88]: #confusion matrix
```



```
In [85]: 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=testset.at[n,'X'+str(i)]
                 a=a*dzeroprob[k-1][i-1]
                 b=b*doneprob[k-1][i-1]
                 if i==5:
                     break
             a=a*pzero
             b=b*pone
             #print(a)
             #print(b)
             if(a>b):
                 predict=0
             else:
                 predict=1
             #print(d)
             d=testset.at[n,'D']
             #print(d)
```



```
    #print(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 [87]:
```
confusion=np.array([[tp,fp],[fn,tn]])
confusion
```

Out[87]:
```
array([[5, 4],
       [3, 2]])
```

In [89]:
```
correctness=(tp+tn)/(tp+tn+fp+fn)
print("Correctness = ",correctness)
```

```
Correctness =  0.5
```

In [90]:
```
erorrness=(fp+fn)/(tp+tn+fp+fn) #1-correctness
print("Erorrness = ",erorrness)
```

```
Erorrness =  0.5
```

In [91]:
```
precision=tp/(tp+fp)
print("precision = ",precision)
```

```
precision =  0.5555555555555556
```

In [92]:
```
recall=tp/(tp+fn)
print("Recall = ",recall)
```

```
Recall =  0.625
```