In [ ]:
```
Problem Statement
Write a program to use a K-nearest neighbor to predict class labels of test d
distance should be used as the distance metric. Consider K=5. The learned cla
be tested on test instances with unknown class labels, and the predicted clas
test instances should be printed as output.
Data Set Description:
Training Data Filename: data4.csv, Training Data File Format: Boolean input a
(x1, x2, ... , x8) in first 8 columns. The last (9th) column represents the E
(y). Each row is a training instance. There are 20 training instances.
Test Data Filename: test4.csv, Test Data File Format: Boolean input attribute
x8) in each of the 8 columns. Note that, there is no class label column. Each
instance. There are 4 test instances. The row number corresponds to the insta
the test instances.
Input Format: Assume the data files data4.csv and test4.csv are present in th
directory and contain the training and test data. Thus, your program should r
input from the user and should read from these files.
Output Format: Predicted class labels (0/1) for the test data exactly in the
test instances are present in the test file. e.g., output 0 0 1 1 can give cl
Instance 1: 0, Test Instance 2: 0, Test Instance 3: 1, Test Instance 4: 0
```

In [8]:
```
import numpy as np
import pandas as pd
```

In [9]:
```
dataset=pd.read_csv("data4.csv",header=None)
d=dataset.to_numpy()
```

In [10]:
```
k=5
```

In [11]:
```
testdataset=pd.read_csv("test4.csv",header=None)
t=testdataset.to_numpy()
```

In [12]:
```
e=np.zeros((d.shape[0],t.shape[0]))
```

In [19]:
```
for i in range(0,t.shape[0]):
   for j in range(0,d.shape[0]):
        e[j,i]=np.sum(np.square(d[j,0:8] - t[i,:]))
```

In [20]:
```
e
```

Out[20]:
```
array([[2., 6., 4., 3.],
       [3., 5., 3., 2.],
       [1., 7., 5., 4.],
       [3., 5., 3., 2.],
       [4., 4., 2., 1.],
       [3., 5., 3., 4.],
       [3., 5., 5., 4.],
       [4., 4., 2., 3.],
       [4., 4., 2., 3.],
       [5., 3., 1., 2.],
       [1., 7., 3., 2.],
       [2., 6., 2., 1.],
       [3., 5., 5., 4.],
       [2., 6., 2., 1.],
       [4., 4., 6., 5.],
```

```
       [4., 4., 6., 5.],
       [4., 4., 8., 7.],
       [3., 5., 1., 0.],
       [2., 6., 6., 5.],
       [2., 6., 2., 3.]])
```

In [21]:
```
x=np.argsort(e,axis=0)
x
```

Out[21]:
```
array([[ 2,  9,  9, 17],
       [10,  4, 17,  4],
       [ 0,  7, 13, 13],
       [13,  8, 11, 11],
       [11, 16,  8,  9],
       [18, 15,  7,  1],
       [19, 14, 19,  3],
       [ 5,  1,  4, 10],
       [ 3, 17,  5,  0],
       [12,  3, 10,  7],
       [ 1,  5,  3,  8],
       [17,  6,  1, 19],
       [ 6, 12,  0,  6],
       [ 7,  0, 12, 12],
       [ 8, 11,  2,  5],
       [ 4, 18,  6,  2],
       [14, 13, 18, 18],
       [15, 19, 14, 14],
       [16, 10, 15, 15],
       [ 9,  2, 16, 16]])
```

In [23]:
```
knn=np.zeros((k,t.shape[0]))
for j in range(0,t.shape[0]):
    for j in range(0,k):
        knn[j,i]=d[x[j,i],8]
```

In [24]:
```
knn
```

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

In [26]:
```
b=[]
for i in range(0,knn.shape[1]):
    a=knn[:,i]
    a=a.astype(int)
    c=np.bincount(a)
    maxc=np.argmax(c)
    b=np.append(b,maxc)
```

In [27]:
```
print(b)
```

```
[0. 0. 0. 1.]
```

In [ ]:
```

```