# LAB CYCLE 2

## Experiment No:2

## Date:15/12/21

## Aim:

**(KNN – Classifier)**

**Write a program to use a K-nearest neighbour to predict class labels of test data.Euclidean distanceshould be used as the distance metric.consider K=5.The learned classifier should be tested on test instances with unknown class labels, and the predicted class labels for the test instances should be printed as output.**

## Source Code:

In [51]:
```
#Assignment 2: KNN Classifier
```

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
#Reading the training data
data4=pd.read_csv("data4.csv",header=None)
d4=data4.to_numpy()
```

In [3]:
```
data4
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| **2** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **3** | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **4** | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| **5** | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| **6** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| **7** | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| **8** | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| **9** | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| **10** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| **11** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| **12** | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| **13** | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| **14** | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| **15** | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| **16** | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| **17** | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| **18** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| **19** | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |

In [4]:
```
d4
```

Out[4]
```
array([[1, 1, 1, 1, 1, 1, 0, 1, 1],
       [1, 1, 1, 1, 1, 1, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1, 0, 0, 1, 1],
       [1, 1, 1, 1, 1, 0, 0, 0, 1],
```

```
       [1, 1, 0, 1, 1, 1, 0, 1, 0],
       [1, 1, 1, 0, 1, 1, 0, 0, 1],
       [1, 1, 1, 0, 1, 0, 0, 1, 1],
       [1, 1, 1, 0, 1, 0, 0, 0, 1],
       [0, 1, 1, 1, 1, 1, 0, 1, 1],
       [0, 1, 1, 1, 1, 1, 0, 0, 1],
       [1, 0, 1, 1, 1, 1, 0, 1, 0],
       [0, 1, 1, 1, 1, 0, 0, 1, 1],
       [1, 1, 0, 1, 0, 1, 0, 1, 0],
       [1, 0, 0, 1, 1, 1, 0, 1, 0],
       [1, 0, 0, 1, 0, 1, 1, 1, 0],
       [0, 1, 1, 1, 1, 0, 0, 0, 1],
       [1, 0, 1, 1, 1, 1, 1, 1, 0],
       [0, 1, 1, 0, 1, 1, 0, 1, 1]], dtype=int64)
```

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

In [6]:
```
#Reading testing data
test4=pd.read_csv("test4.csv",header=None)
t4=test4.to_numpy()
```

In [7]:
```
test4
```

Out[7]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **1** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| **3** | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

In [8]:
```
t4
```

Out[8]:
```
array([[0, 1, 1, 1, 1, 1, 1, 1],
       [1, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 0, 1, 0, 0, 0],
       [0, 1, 1, 1, 1, 0, 0, 0]], dtype=int64)
```

In [9]:
```
#Array with size of both Training and Testing data
euclid=np.zeros((d4.shape[0],t4.shape[0]))
euclid
```

Out[9]:
```
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., 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.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [10]:

```
#find distance using square and difference between training data and testing date
for i in range(0,t4.shape[0]):
    for j in range(0,d4.shape[0]):
        euclid[j,i]=np.sum(np.square(d4[j,0:8]-t4[i,:]))
euclid
```

Out[10]:

```
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 [11]:

```
#sort based on location using function argsort()
array_sort=np.argsort(euclid,axis=0)
array_sort
```

Out[11]:

```
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]], dtype=int64)
```

In [12]:

```
k=5
knn=np.zeros((k,t4.shape[0]))
for i in range(0,t4.shape[0]):
    for j in range(0,k):
        knn[j,i]=d4[array_sort[j,i],8]
knn
```

Out[12]:

```
array([[0., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 0., 1., 1.]])
```

In [13]:

```
#finding the class label
result=[]
for i in range(0,knn.shape[1]):
    b=knn[:,i]
    b=b.astype(int)
    counts=np.bincount(b)
    maxlabel=np.argmax(counts)
    result=np.append(result,maxlabel)
result
```

Out[13]:

```
array([1., 1., 1., 1.]
```

In [ ]: