# Experiment 8

**AIM : k Nearest Neighbors**

***To peform classification using k Nearest Neighbors***

### DESCRIPTION

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

### *ALGORITHM*

Let m be the number of training data samples. Let p be an unknown point.

```
1. Store the training samples in an array of data points arr[]. This means each
 element of this array represents a tuple (x, y).
2. for i=0 to m:
      Calculate Euclidean distance d(arr[i], p).

          $d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2}$

3. Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point.
4. Return the majority label among S.
```

### CODE and OUTPUT:

In [1]:

```python
# Implementation of k Nearest Neighbors Aglorithm
# Libraries Needed
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import math
```

In [2]:

```
# Loading Data set
iris = datasets.load_iris()
X = iris.data
y = iris.target
X.shape,y.shape
```

Out[2]:

```
((150, 4), (150,))
```

In [3]:

```
# Splitting dataset in Training and Testing data
i = np.random.randn(len(y))<float(2/3)
XTrain = X[i]
yTrain = y[i]
XTest = X[~i]
yTest = y[~i]

XTrain.shape,yTrain.shape,XTest.shape,yTest.shape
```

Out[3]:

```
((108, 4), (108,), (42, 4), (42,))
```

In [4]:

```
# Euclidean Distance between 2 datapoints
def euclideanDistance(a1,a2,length):
    distance = 0
    for i in range(length):
        distance = distance + (a1[i]-a2[i])**2
    distance = math.sqrt(distance)
    return distance
```

In [5]:

```
# Finding k Nearest Neighbors of test instance in Training data
def kNearestNeighbors(trainX,trainy,instance,k):
    dists = []
    n = len(instance)
    for i in range(len(trainy)):
        dist = euclideanDistance(trainX[i],instance,n)
        dists.append((trainX[i],trainy[i],dist))
    dists = sorted(dists, key = lambda x:x[2])
    neighbors = dists[:k]
    return neighbors
```

In [6]:

```
# Getting Majority Class in Neighbors
def getClass(neighbors):
    classes = dict()
    for neighbor in neighbors:
        classes[neighbor[1]] = classes.get(neighbor[1],0) +1
    classes = sorted(classes.items(), key = lambda x:x[1],reverse = True)
    return classes[0][0]
```

In [7]:

```
# Predicting Classes for test data
prediction = []
k=3
for x in XTest:
    neighbors = kNearestNeighbors(XTrain,yTrain,x,k)
    res = getClass(neighbors)
    prediction.append(res)
```

In [8]:

```
# Checking accuracy in predicted data
correct = 0
for x in range(len(yTest)):
    if yTest[x] == prediction[x]:
        correct += 1

(correct/float(len(yTest)))*100.0
```

Out[8]:

```
95.23809523809523
```

**LEARNING OUTCOMES :**

***In this Experiment we learned how we can classify data using k Nearest Neighbors Algorithm***