1. **Nearest Neighbour Algorithm**

Name : M SAI RAJ

Uid no:19BDA7131

**ALGORITHM:**

**Step 1** − Load the training and test data.

**Step 2** − Choose the value of K i.e. the nearest data points. K can be any integer (preferably not 1, but any other odd value)

**Step 3** − For each point in the test data do the following −

* **3.1** − Calculate the distance between test data and each row of training data with Euclidean Distance Formula.
* **3.2** − Based on the distance value, sort them in ascending order.
* **3.3** − Next, it will choose the top K rows from the sorted array.
* **3.4** – Compute the average of sum of the preceding rows and calculate the percentage error. The predicted value corresponds to the value with the least percentage error.

**Step 4** − End

#Defining the train and test data

#initialising empty lists

import math

A=[[40,174,63,5.7,69],

[50,126,73,5.0,66],

[32,140,72,5.3,85],

[48,123,64,6.1,88],

[28,132,74,4.6,83],

[27,178,74,5.2,82],

[26,148,77,6.0,88],

[23,120,68,5.8,75],

[38,177,73,4.5,70],

[29,101,72,6.3,88]]

B=[50,130,70,5.0]

K=2

T=[]

V=[]

DM=[]

n=len(A)

# finding the difference and appending the difference into final list as lists

for i in range (0,n):

D=[]

for j in range(len(A[i])-1):

D.append((A[i][j]-B[j])\*\*2)

DM.append(D)

# finding the euclidean distance

for k in range(len(DM)):

s=math.sqrt(sum(DM[k]))

V.append(s)

for y in range(0,n):

A[y].append(V[y])

#sorting the distance in ascending order

srt=sorted(A,key=lambda x:x[4])

srt

#calculating the cumilative sum and hence the average using k values

#calculating the percentage error

distance\_column\_copy=A.copy()

distance\_indices=[A.index(index\_val) for index\_val in distance\_column\_copy]

distance\_indices=distance\_indices[0:k]

prediction= sum([A[i][len(A[i])-1] for i in distance\_indices])/k

prediction

**OUTPUT:**

Out[6]:

29.442575635020415

…………………