**K-Nearest Neighbour Algorithm**

**PROBLEM STATEMENT:**

An attempt to predict the Weight using KNN Algorithm without any inbuilt packages.

**IMPORTANT FORMULAS USED:**

Euclidean Distance Formula:

Distance between any two points (x1,y1) and (x2,y2) is given by

√[(x2-x1)2 + (y2-y1)2]

**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

**CODE:**

@Script Author : Richa Elizabath

@Description : K-Nearest neighbour algorithm without any packages

@Start Date : 07-01-2020

@Last Edited : 11-01-2020

@Python Version : Python 3.7.3

#Defining the train and test data

#initialising empty lists

f=[]

train=[[3,5,1],[6,8,2],[5,9,1],[3,9,5],[7,2,9]]

test=[7,4,9]

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

Tr=len(train)

Ts=len(test)

for i in range(Tr):

d=[]

for j in range(Ts-1):

diff=test[j]-train[i][j]

d.append(diff)

f.append(d)

f

# finding the euclidean distance

for i in range(len(f)):

for j in range(len(f[0])):

f[i][j]=f[i][j]\*f[i][j]

f

import math

f\_sum=[]

s=0

for i in range(len(f)):

for j in range(len(f[0])):

s=s+f[i][j]

s=math.sqrt(s)

f\_sum.append(s)

f\_sum

# mapping the distance to corresponding element in training data in dictionary

dict={f\_sum[0]:train[0],f\_sum[1]:train[1],f\_sum[2]:train[2],f\_sum[3]

:train[3],f\_sum[4]:train[4]}

dict

#sorting the distance in ascending order

dict1=sorted(dict.items())

dict1

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

s=0

cum\_sum=[]

for i in range(len(dict1)):

s=(s+dict1[i][1][2])

cum=s/(i+1)

cum\_sum.append(cum)

cum\_sum

#calculating the percentage error

perc\_error=[]

dict\_fin=[]

for i in range(len(test)):

di=test[2]-cum\_sum[i]

perc\_error.append(abs((di/test[2])\*100))

dict\_fin={perc\_error[0]:cum\_sum[0],perc\_error[1]:cum\_sum[1],perc\_err

or[2]:cum\_sum[2]}

dict\_fin

**OUTPUT:**

Percentage:cum sum

0.0: 9.0

44.44444444444444: 5.0

55.55555555555556: 4.0}