**K-Nearest Neighbour Algorithm**

**PROBLEM STATEMENT:**

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

**FORMULAS USED:**

Euclidean Distance Formula:

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

[(x2-x1)2 + (y2-y1)2]1/2

**ALGORITHM:**

**Step 1** − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

**Step 2** − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

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

**Step 4-** Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

**Step 5 -** Now, based on the distance value, sort them in ascending order.

**Step 6-**  Next, it will choose the top K rows from the sorted array.

**Step 7-** Now, it will assign a class to the test point based on most frequent class of these rows.

**Step-8** End

**CODE:**

@Script Author : Devika M R

@Description   : K-nearest neighbour algorithm without any packages

@Start Date    : 07-01-2020

@Last Edited   : 11-01-2020

@Python Version : Python 3.7

import math

train=[[35,164,63,5.7,69],

[60,196,73,5.0,66],

[32,10,72,5.3,85],

[28,13,64,6.1,88]]

test=[30,70,120,7.0]

K=2

T=[]

s1=[]

dif1=[]

n=len(train)

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

for i in range (0,n):

dif=[]

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

dif.append((train[i][j]-test[j])\*\*2)

dif1.append(dif)

# finding the euclidean distance

for k in range(len(dif1)):

s=math.sqrt(sum(dif1[k]))

s1.append(s)

for y in range(0,n):

train[y].append(s1[y])

#sorting the distance in ascending order

sort=sorted(train,key=lambda x:x[4])

sort

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

#calculating the percentage error

dic=train.copy()

di=[train.index(index\_val) for index\_val in dic]

di=di[0:k]

prediction= sum([train[i][len(train[i])-1] for i in di])/k

prediction

**OUTPUT**

108.24533891549652