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

$\sqrt{[(x2-x1)^2 + (y2-y1)^2]}$

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 : Thomas Thomas
@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
d,f,diff=[],[],[]

#training data
train=[[40,174,63,5.70,69],[50,126,73,5.00,66],[32,140,72,5.30,85],[48,123,64,6.10,88],[28,132,74,4.60,83],[27,178,74,5.20,82],[26,148,77,6.00,88],[23,120,68,5.80,75],[38,177,73,4.50,70],[29,101,72,6.30,88]]

#testing data
test=[50,130,70,5.00,80]

# finding the difference and appending the difference into final list as lists
for i in range(len(train)):
    diff=[]
    for j in range(len(test)-1):
        x=test[j]-train[i][j]
        diff.append(x)
    f.append(diff)

# finding the euclidean distance
for i in range(n):
    s=0
    for j in range(len(test)-1):
        s=s+f[i][j]**2
    d.append(s**0.5)
d

# mapping the distance to corresponding element in training data in dictionary
dict1={}
for m in range(len(d)):
    dict1[d[m]]=train[m]
dict1

#sorting the distance in ascending order
q=sorted(dict1.items())
q

#calculating the cumilative sum and hence the average using k values
predicted=[]
sums=0
for u in range(n):
    sums=sums+q[u][1][2]
    avg=sums/(u+1)
    print(sums)
    print(avg)
    predicted.append(avg)
predicted

#calculating the percentage error
z=0
error=[]
for s in range(n):
    z=abs(((test[2]-predicted[s])*100)/(test[2]))
    error.append(z)
error
```

```
#defining a dictionary which maps percentage error to predicted
value

dict2={}

for w in range(n):

    dict2[error[w]]=predicted[w]

dict2


#printing the predicting value

print("predicted value is ",dict2[min(error)])


#printing the actual value

print("actual value is ",test[len(test)-1])


#printing the minimum percentage error

print("percentage error is ",min(error))
```

OUTPUT:

```
predicted value is  80.25
actual value is  80
percentage error is  31.25
```