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

$[(x2-x1)^2 + (y2-y1)^2]^{1/2}$

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 : teena saj
@Description   : K-nearest neighbour algorithm without any packages
@Start Date    : 07-01-2020
@Last Edited   : 11-01-2020
@Python Version : Python 3.7



#Defining the train and test data
train=[[1,2,30.6],[3,4,40.2],[5,6,30.7]]
test=[[7,8,40.2]]
n=len(train)

#Initialising various lists
k=[]
diff=[]
dm=[]
dm1=[]

#k values
for i in range(1,n+1):
    k.append(i)



#Finding the difference matrix
target=test[0][2]
predict=[]
for i in range(len(train)):
    for j in range(len(train[0])-1):
        diff.append(test[0][j]-train[i][j])
```

```
for i in range(0,len(diff),2):
    dm.append(diff[i:i+2])
print("dm is:: ",dm)    #printing the difference matrix

#finding the distance
for i in range(len(dm)):
    s=0
    for j in range(len(dm[0])):
        s=s+dm[i][j]**2    #applying the Euclidean distance formula
    dm1.append(s**0.5)
dm1

#adding the distance value along with the train set
dm2={}
for i in range(len(dm1)):
    dm2[dm1[i]]=train[i]
dm2

#Sorting the distance
dm2=sorted(dm2.items())
dm2



#Finding the predicted value
s1=0
for i in range(len(dm2)):
    s1=s1+dm2[i][1][2]
    avg=s1/(i+1)
    predict.append(avg)
predict



#finding the error values
dm3=[]
s2=0
```

```
for i in range(len(predict)):
    s2=(target-predict[i])*(100)/(target)
    dm3.append(abs(s2))
dm3

#Printing the accurate value
print("The Accurate value is",predict[1])
```

OUTPUT:

```
dm is::  [[6, 6], [4, 4], [2, 2]]
The Accurate value is 35.45
```