**Program 6: KNN**

* Without Normalization

import pandas as pd

import math

dS = pd.read\_csv('resident.csv')

mainAttr = list(dS.columns[:-1])

lastCol = dS.columns[-1]

# dS.iloc[1,-1]

def takeIp(dS):

    ip = {}

    k = input("k :")

    for attr in dS.columns[:-1]:

        # print(attr)

        ip[attr] = float(input(f"for {attr}: "))

    return ip,k

def euclidion(dS,iD={}):

    ans = {}

    array = dS.to\_numpy()

    rowNo = 0

    for row in array:

        cal = 0

        i = 0

        for attr in mainAttr:

            cal += math.pow((int(row[i]) - int(iD[attr])),2)

            i += 1

        ans[rowNo] = cal

        rowNo += 1

    lst = ans.items()

    slst = sorted(lst,key = lambda x : x[1])

    return slst

def takeDicision(k,distance,dS):

    valueDic = {}

    newlst = distance[:k]

    for value in newlst:

        valueDic[dS.iloc[value[0],-1]] = valueDic.get(dS.iloc[value[0],-1],0) + 1

    return max(valueDic,  key=valueDic.get)

iD, k = takeIp(dS)

eulst = euclidion(dS, iD)

ans = takeDicision(int(k),eulst,dS)

print(ans)

Output:

k :3

for age: 48

for loan: 142000

y

* With Normalization

import pandas as pd

import math

dS = pd.read\_csv('resident.csv')

mainAttr = list(dS.columns[:-1])

lastCol = dS.columns[-1]

# dS.iloc[1,-1]

def minmax(dS):

    for attr in dS.columns[:-1]:

        minimum = min(dS[attr])

        maximum = max(dS[attr])

        for i in range(len(dS)):

            value = dS.loc[i, attr]

            newValue = (value - minimum)/(maximum - minimum)

            dS.loc[i, attr] = newValue.\_\_round\_\_(3)

    return dS

def takeIp(dS):

    ip = {}

    k = input("k :")

    for attr in dS.columns[:-1]:

        # print(attr)

        ip[attr] = float(input(f"for {attr}: "))

    return ip,k

def euclidion(dS,iD={}):

    ans = {}

    array = dS.to\_numpy()

    rowNo = 0

    for row in array:

        cal = 0

        i = 0

        for attr in mainAttr:

            cal += math.pow((int(row[i]) - int(iD[attr])),2)

            i += 1

        ans[rowNo] = cal

        rowNo += 1

    lst = ans.items()

    slst = sorted(lst,key = lambda x : x[1])

    return slst

def takeDicision(k,distance,dS):

    valueDic = {}

    newlst = distance[:k]

    for value in newlst:

        valueDic[dS.iloc[value[0],-1]] = valueDic.get(dS.iloc[value[0],-1],0) + 1

    return max(valueDic,  key=valueDic.get)

dS = minmax(dS)

iD, k = takeIp(dS)

eulst = euclidion(dS, iD)

ans = takeDicision(int(k),eulst,dS)

print(ans)

Output:

k :3

for age: 0.7

for loan: 0.61

n

Dataset:

age,loan,default

25,40000,n

35,60000,n

45,80000,n

20,20000,n

35,120000,n

52,18000,n

23,95000,y

40,62000,y

60,100000,y

48,220000,y

33,150000,y