

**PYTHON CODE**

import pandas as pd

import numpy as np

data=pd.read\_csv('LoadData\_in\_kW.csv')

from sklearn import preprocessing

data= preprocessing.normalize(data)

data= pd.DataFrame(data=data, columns=["X", "Y"])

data

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data.X, data.Y, test\_size = 0.10, random\_state = 4)

m = 1 #Initial value of slope

c = -1 #Initial value of intercept

lr = 0.01 #Learning Rate

delta\_m = 1 #Initialising Δm

delta\_c = 1 #Initialising Δc

max\_iters = 1000 #Maximum number of iterations

iters\_count = 0 #Counting Iterations

def deriv(m\_f, c\_f, datax, datay):

    m\_deriv = 0

    c\_deriv = 0

    for i in range(datax.shape[0]):

        x, y = datax.iloc[i], datay.iloc[i]

        m\_deriv += (y-m\_f\*x-c\_f)\*x

        c\_deriv += (y-m\_f\*x-c\_f)

        m\_deriv = -m\_deriv/len(datax)

        c\_deriv = -c\_deriv/len(datay)

    return m\_deriv, c\_deriv

while iters\_count < max\_iters:

    delta\_m, delta\_c = deriv(m, c, x\_train, y\_train)

    delta\_m = -lr \* delta\_m

    delta\_c = -lr \* delta\_c

    m += delta\_m

    c += delta\_c

    iters\_count += 1

    print(f"Iteration: {iters\_count}\tValue of m: {m}, \tValue of c: {c}")

print(f"\nThe local minima occurs at: {m}, {c}")

x\_train = np.array(x\_train)

y\_train = np.array(y\_train)

x\_test = np.array(x\_test)

y\_test = np.array(y\_test)

i=0

y\_pred=[]

while i<27:

    y\_predict=(m\*x\_train[i])-c

    y\_pred.append(y\_predict)

    i=i+1

i=0

yt\_pred=[]

while i<3:

    yt\_predict=(m\*x\_test[i])-c

    yt\_pred.append(yt\_predict)

    i=i+1

import math

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

def accuracy\_op(y\_train,y\_pred):

    mse = math.sqrt(mean\_squared\_error(y\_train,y\_pred))

    print('Root mean square error', mse)

    mse = (mean\_squared\_error(y\_train,y\_pred))

    print('Mean square error', mse)

    mae=mean\_absolute\_error(y\_train, y\_pred)

    print('Mean absolute error', mae)

print(accuracy\_op(y\_train,y\_pred))

print(accuracy\_op(y\_test,yt\_pred))