**SAMPLE-1**

**ITERATION-1**

**step:1** - [7.6,157],eta=0.01,m=1,c=-1

**step-2**  - de/dm|m = 1 ===> -(y-mx-c)\*(-x)

= 1143.04

de/dc|c = -1 ==> -(y-mx-c)

= -150.4

**step-3** - delta(m) = -eta(de/dm) = -(0.01)(1143.04)

= -11.430

delta(c) = -eta(de/dc) = -(0..01)(-150.4)

= 1.504

**step-4** - m = m+delta(m) = 1+(11.43) = -10.43

c = c+delta(c) = -1+(1.504) = 0.504

**ITERATION-2**

**step:1** - [7.6,157], eta = 0.01, m = -10.43, c = -0.504

**step-2** - de/dm|m = -10.43 ===> -(y-mx-c)\*(-x)

= 1794.955

de/dc|c = 0.504 ==> -(y-mx-c)

= -235.868

**step-3** - delta(m) = -eta(de/dm) = -17.949

delta(c) = -eta(de/dc) = 2.358

**step-4** - m = m+delta(m) = -28.379

c = c+delta(c) = 2.868

**SAMPLE-2**

**ITERATION-1**

**step:1** - [7.1,174], eta = 0.01, m = 1, c = -1

**step-2** - de/dm|m = 1 ===> -(y-mx-c)\*(-x)

= 1192.09

de/dc|c = -1 ==> -(y-mx-c)

= -167.9

**step-3** - delta(m) = -eta(de/dm) = -11.920

delta(c) = -eta(de/dc) = 1.679

**step-4** - m = m+delta(m) = 1+(11.43) = -10.920

c = c+delta(c) = -1+(1.504) = 0.679

**ITERATION-2**

**step:1** - [7.6,157], eta = 0.01, m = -10.920, c = -0.679

**step-2** - de/dm|m = -10.43 ===> -(y-mx-c)\*(-x)

= 1781.056

de/dc|c = 0.504 ==> -(y-mx-c)

= -250.853

**step-3** - delta(m) = -eta(de/dm) = -17.810

delta(c) = -eta(de/dc) = 2.508

**step-4** - m = m+delta(m) = -28.73

c = c+delta(c) = 3.187

**PYTHON CODE**

import pandas as pd

import numpy as np

data=pd.read\_csv('groundwater\_survey.csv')

from sklearn import preprocessing

data= preprocessing.normalize(data)

data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data = scaler.fit\_transform(data)

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 = 0.1 #Learning Rate

delta\_m = 1 #Initialising Δm

delta\_c = 1 #Initialising Δc

max\_iters = 1000 #Maximum number of iterations

iters\_count = 0 #Counting Iterations

#samples = 30

def deriv(m\_f, c\_f, x, y):

    m\_deriv = -1\*(y-m\_f\*x-c\_f)\*x

    c\_deriv = -1\*(y-m\_f\*x-c\_f)

    return m\_deriv, c\_deriv

while iters\_count < max\_iters:

    for i in range(x\_train.shape[0]):

        delta\_m, delta\_c = deriv(m, c, x\_train.iloc[i], y\_train.iloc[i])

        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<30:

    y\_predict=(m\*x\_train[i])-c

    y\_pred.append(y\_predict)

    i=i+1

i=0

yt\_pred=[]

while i<4:

    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))