import pandas as pd

import numpy as np

import matplotlib.pyplot as graph

import math

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LinearRegression

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

""" LOADING DATA """

def openCSV():

data = pd.read\_csv("exams.csv")

exam\_1 = data['exam\_1']

exam\_2 = data['exam\_2']

passorNOT = data['admitted']

exam\_1 = (exam\_1 - exam\_1.mean()) / exam\_1.std()

exam\_1 = np.c\_[np.ones(exam\_1.shape[0]),exam\_1]

exam\_2 = (exam\_2 - exam\_2.mean()) / exam\_2.std()

passorNOT = (passorNOT - passorNOT.mean()) / passorNOT.std()

return exam\_1,exam\_2,passorNOT;

""" Visualization part 1. """

def firstplot():

exam\_1, exam\_2, passorNOT = openCSV();

firstvar = exam\_1[:,1]

secondvar = exam\_2

graph.xlabel('First Exam Score')

#graph.scatter(firstvar, secondvar, c = 'b')

graph.scatter(exam\_1[:,1],exam\_2, color="red")

#graph.scatter(exam\_1[:,1],exam\_2, color = "blue")

graph.ylabel('Second Exam Score')

graph.title('First vs Second')

graph.show()

""" SIGMA FUNCTION """

def sigmofunc(exam\_1):

exam\_1 = 4;

func = float(1 / (1 +math.exp(-exam\_1)))

return func

""" COST FUNCTION """

def costfunc(exam\_1,exam\_2):

exam\_1, exam\_2, passorNOT = openCSV();

m = len(exam\_1)

for i in range(0,2):

cost = -1/m \* (exam\_2[i]\*np.log(sigmofunc(exam\_1[i]))+(1-exam\_2[i])\*np.log(1-sigmofunc(exam\_1[i])))

#print("The cost function in iteration %d is %.3f" % (i,cost))

return cost

""" GRADIENT FUNCTION """

def graddescent(exam\_1, exam\_2,theta, learningrate, numofiter):

exam\_1, exam\_2, passorNOT = openCSV();

m = len(exam\_1)

for i in range(1, m):

theta[0] = theta[0] - learningrate\*(sigmofunc(exam\_1[i]) - exam\_2[i]) \* exam\_1[i]

theta[1] = theta[1] - learningrate\*(sigmofunc(exam\_1[i]) - exam\_2[i]) \* exam\_1[i]

return theta

""" PLOTTING EXAM 1 EXAM 2 AND ADMIITTED """

def plottingall(exam\_1, exam\_2,passorNOT):

exam\_1, exam\_2, passorNOT = openCSV();

graph.scatter(exam\_1[:,1],exam\_2, color="red")

graph.plot(passorNOT)

graph.show()

""" TESTING PART """

def testing():

exam\_1, exam\_2, passorNOT = openCSV();

#data1 = [55,70,1]

#data2 = [40,60,0]

data1 = np.random.rand(100)

data2 = np.random.rand(100)

exam\_1 = 50

exam\_2 = 60

priceprediction = data1 \* exam\_2 + data1 \* exam\_1 + data1

pred = priceprediction \* exam\_2 + exam\_2

actdata = exam\_2 \* exam\_2 + exam\_2

print("This is for test number 1.")

print("Predicted values I got is:")

print(pred)

print("Actual data is:",(actdata))

print("\nThis is for test number 2.\n")

test1 = [55,70,1]

test2 = [40, 60, 0]

print(accuracy\_score(test2, test1))

print(accuracy\_score(test2, test1, normalize=False))

""" LINEAR REGRESSION FUNCTION """

def LinearRegro():

linearreg = LinearRegression()

exam\_1, exam\_2, passorNOT = openCSV();

exam\_1Train, exam\_1Test, exam\_2Train, exam\_2Test = train\_test\_split(exam\_1, exam\_2, test\_size = 1/3, random\_state = 0)

linearreg.fit(exam\_1Test, exam\_2Test)

exam\_2Predict = linearreg.predict(exam\_1Train)

graph.scatter(exam\_1Test[:,1], exam\_2Test, color = 'black')

graph.plot(exam\_1Test, linearreg.predict(exam\_1Test), color = 'red')

graph.title("Linear Regression Test ")

graph.xlabel("Exam #1")

graph.ylabel("Exam #2")

graph.show()

""" EVERYTHING PRINTED HERE """

def forPrinting():

theta = [0,0]

numofiter = 100

learningrate = 0.01

exam\_1, exam\_2, passorNOT = openCSV(); #GETTING DATA HERE

m = len(exam\_1)

firstplot() #Visualization 1

print("The value we get from sigma function is",sigmofunc(5))

for num in range(0, numofiter):

costfinal = costfunc(exam\_1,exam\_2) #COST FUNCTION PRINTED HERE

theta = graddescent(exam\_1, exam\_2, theta, learningrate, numofiter) #GRADIENT PRINTED HERE

#print("Cost function in iteration %d is %s" % (num, costfinal))

#print("First theta is equal to %s and the second theta equal to %s" %(theta[0],theta[1]))

plottingall(exam\_1,exam\_2, passorNOT) #Visualization PART 2

testing() #TESTING PART

LinearRegro() #LINEAR REGRESSION

forPrinting() #PRINT FUNCTION CALLED