import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

data = pd.read\_csv('C:/Users/Aastha Kanaujia/Downloads/AIML\_Datasets/student.csv')

print(data.shape)

data.head()

math = data['Math'].values

read = data['Reading'].values

write = data['Writing'].values

fig = plt.figure()

ax = Axes3D(fig)

ax.scatter(math, read, write, color='#ef1234')

# plt.legend()

plt.show()

m = len(math)

x0 = np.ones(m)

X = np.array([x0, math, read]).T

B = np.array([0, 0, 0])

Y = np.array(write)

alpha = 0.0001

def cost\_function(X, Y, B):

m = len(Y)

J = np.sum((X.dot(B) - Y) \*\* 2)/(2 \* m)

return J

inital\_cost = cost\_function(X, Y, B)

print("Initial Cost")

print(inital\_cost)

def gradient\_descent(X, Y, B, alpha, iterations):

cost\_history = [0] \* iterations

m = len(Y)

for iteration in range(iterations):

# Hypothesis Values

h = X.dot(B)

# Difference b/w Hypothesis and Actual Y

loss = h - Y

# Gradient Calculation

gradient = X.T.dot(loss) / m

# Changing Values of B using Gradient

B = B - alpha \* gradient

# New Cost Value

cost = cost\_function(X, Y, B)

cost\_history[iteration] = cost

return B, cost\_history

# 100000 Iterations

newB, cost\_history = gradient\_descent(X, Y, B, alpha, 100000)

# New Values of B

print("New Coefficients")

print(newB)

# Final Cost of new B

print("Final Cost")

print(cost\_history[-1])

def rmse(Y, Y\_pred):

rmse = np.sqrt(sum((Y - Y\_pred) \*\* 2) / len(Y))

return rmse

def r2\_score(Y, Y\_pred):

mean\_y = np.mean(Y)

ss\_tot = sum((Y - mean\_y) \*\* 2)

ss\_res = sum((Y - Y\_pred) \*\* 2)

r2 = 1 - (ss\_res / ss\_tot)

return r2

Y\_pred = X.dot(newB)

#print("RMSE")

#print(rmse(Y, Y\_pred))

print("R2 Score")

print(r2\_score(Y, Y\_pred))