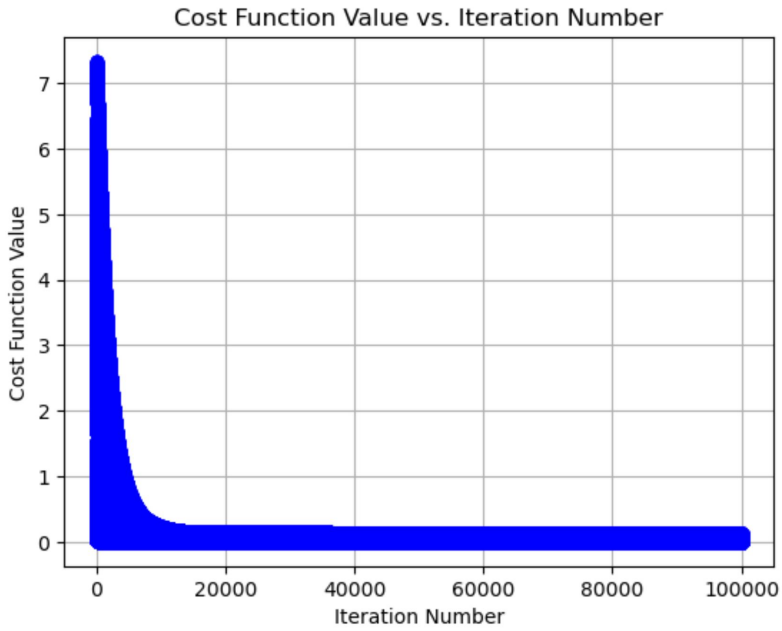
E04-ASSIGNMENT 4 LR USING GRM

NAME:HRISHIKESH S

ROLL NO:22B4217

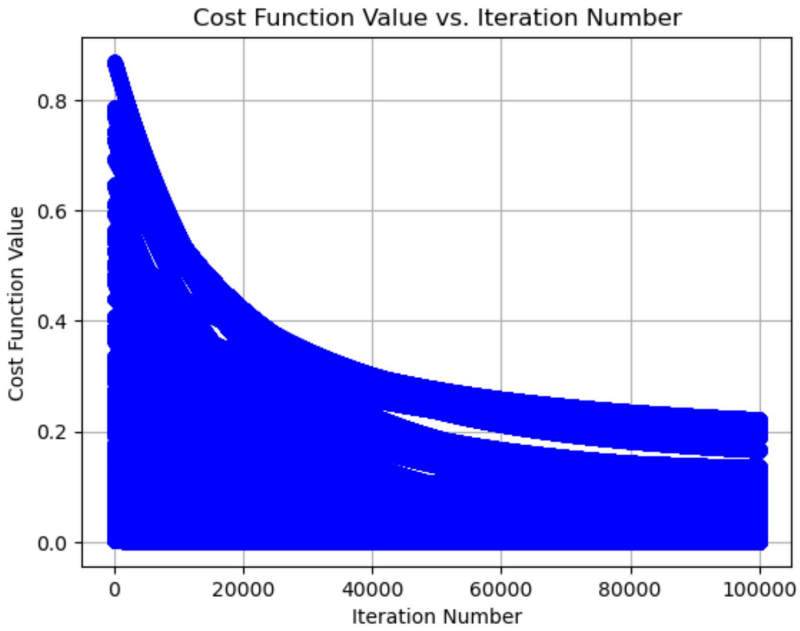
```
import numpy as np
import matplotlib.pyplot as plt
import math

data=np.genfromtxt('linear-data-set-for-regression.csv',delimiter=',')

x=data[:,1]
y=data[:,0]


def cost(x,y,w,b):
    m=x.shape[0]
    cost=0


    for i in range(m):
        f=w*x+b
        cost=cost+(f-y)**2

    total_cost=cost/(2*m)

    return total_cost

def compute_grad(x,y,w,b):
    m=x.shape[0]
    dj_dw=0
    dj_db=0

    for i in range(m):
        f=w*x[i]+b
        dj_dw_i=(f-y[i])*x[i]
        dj_db_i=(f-y[i])
        dj_dw=dj_dw+dj_dw_i
        dj_db=dj_db+dj_db_i

    dj_db1=dj_dw/m
    dj_db0=dj_db/m

    return dj_db0,dj_db1

def gradient_descent(x,y,w_in,b_in,n,iterations,cost,compute_grad):
    J_his=[]
    p_his=[]

    b=b_in
    w=w_in

    for i in range(iterations):
        #Calculating the Derivative terms
        dj_db,dj_dw=compute_grad(x,y,w,b)
```




```
#Different Test Cases
slope=0
intercept=0
n=0.0001
z=10000

slope_final,intercept_final,J_history,p_history=gradient_descent(x,y,slope,in
tercept,n,z,cost,compute_grad)

print(f"(Slope ,Intercept) found by gradient descent:
({slope_final:8.4f},{intercept_final:8.4f})")
```

```
(Slope ,Intercept) found by gradient descent: (  1.0298,  0.6211)
```

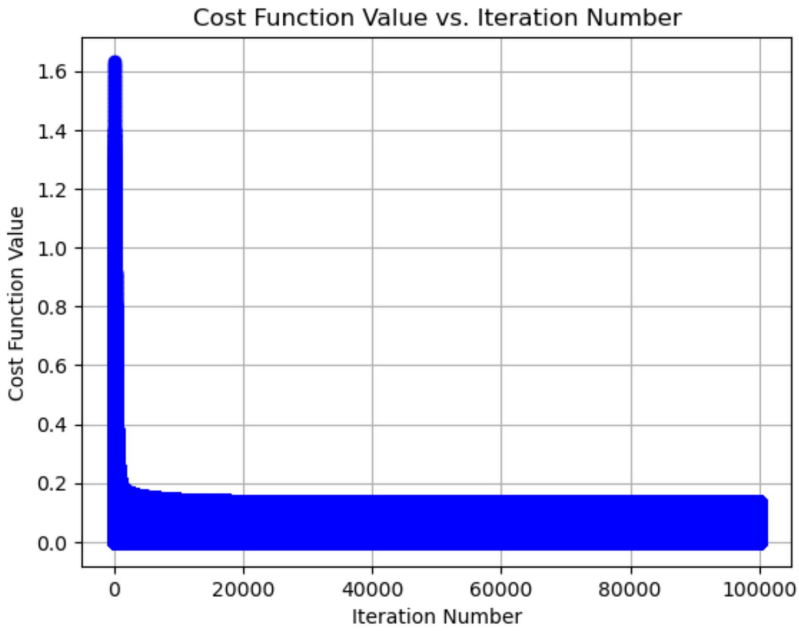
```
plt.plot(range(z), J_history, marker='o', linestyle='-', color='b')
plt.xlabel('Iteration Number')
plt.ylabel('Cost Function Value')
plt.title('Cost Function Value vs. Iteration Number')
plt.grid(True)
plt.show()
```




```
slope=0.80
intercept=0.05
n=0.001
z=100000

slope_final,intercept_final,J_history,p_history=gradient_descent(x,y,slope,in
tercept,n,z,cost,compute_grad)

print(f"(Slope ,Intercept) found by gradient descent:
({slope_final:8.4f},{intercept_final:8.4f})")
```

```
(Slope ,Intercept) found by gradient descent: (  1.1720,  0.5156)
```

```
plt.plot(range(z), J_history, marker='o', linestyle='-', color='b')
plt.xlabel('Iteration Number')
plt.ylabel('Cost Function Value')
plt.title('Cost Function Value vs. Iteration Number')
plt.grid(True)
plt.show()
```



Observations:

As the value of n increases,the resulting values do not converge,but diverges.

Thus we need to keep the value of n as small as possible,and we should note that as we keep making the value of n smaller,we will be required to increase the number of iterations as thenn only after the looping we would reach the minima!

So,we need to keep a balance between the value of n and the number of iterations in which the value of n must be as small as possible and the value of the number of iterations should be as large as possible.

Actually,in the gradient descent method,we are finding the nearest minima,and while we are implementing this Algorithm in Regression,we are actually getting the Global Minima itself and this is a consequence of the nature of the Cost function,which is Quadratic in nature.

Thus this is nothing but a great Algorithm indeed.