import numpy as np

import matplotlib.pyplot as plt

#Get 1000 random inputs for Theta0 and Theta1

Theta0 = np.random.rand(1000)

Theta1 = np.random.rand(1000)

#Plot Theta0 vs Theata1

plt.scatter(Theta0, Theta1)

plt.title('Theta0 vs Theta1')

plt.xlabel('Theta0')

plt.ylabel('Theta1')

plt.show()

#Plot J(Theta0,Theta1) vs Theta0 vs Theata1

def cost\_function(x, y, theta0, theta1):

    cost = 0

    for i in range(len(x)):

        cost += (theta0 + theta1\*x[i] - y[i])\*\*2

    return cost

x = np.random.rand(1000)

y = np.random.rand(1000)

cost\_values = np.array([[cost\_function(x, y, theta0, theta1) for theta0 in Theta0] for theta1 in Theta1])

plt.contour(Theta0, Theta1, cost\_values, np.logspace(-2, 3, 20))

plt.xlabel('Theta0')

plt.ylabel('Theta1')

plt.title('J(theta0, theta1)')

plt.show()

#Best fitting

The minimum of J(Theta0 , Theta1) will be the best fitting model because it represents the point on the graph with the least error. Therefore, the combination of Theta0  and Theta1 that gives the lowest J(Theta0,Theata1) value is the best fitting model.