import numpy as np

# Generate 1000 random numbers (uniform distribution)

theta\_0 = np.random.uniform(0, 1, 1000)

theta\_1 = np.random.uniform(0, 1, 1000)

# Plot Θ⁰ vs Θ¹ graph

import matplotlib.pyplot as plt

plt.scatter(theta\_0, theta\_1)

plt.title("Θ⁰ vs Θ¹")

plt.xlabel("Θ⁰")

plt.ylabel("Θ¹")

plt.show()

# Plot J(Θ⁰, Θ¹ ) vs Θ⁰ vs Θ¹ graph

import numpy as np

# Calculate J(Θ⁰, Θ¹ )

def cost\_function(x, y, theta\_0, theta\_1):

  m = len(x)

  total\_error = 0.0

  for i in range(m):

    total\_error += (y[i] - (theta\_0 + theta\_1 \* x[i]))\*\*2

  return total\_error / m

# Generate some sample data

x = np.linspace(0, 10, 100)

y = 2 + 0.5 \* x

# Generate 1000 random numbers (uniform distribution)

theta\_0 = np.random.uniform(0, 1, 1000)

theta\_1 = np.random.uniform(0, 1, 1000)

# Calculate J(Θ⁰, Θ¹ )

cost = np.zeros(shape=(1000, 1000))

for i in range(1000):

  for j in range(1000):

    cost[i,j] = cost\_function(x, y, theta\_0[i], theta\_1[j])

# Plot J(Θ⁰, Θ¹ ) vs Θ⁰ vs Θ¹ graph

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.plot\_surface(theta\_0, theta\_1, cost, cmap=&#39;viridis&#39;, linewidth=0.2)ax.set\_xlabel('Θ⁰')

ax.set\_ylabel('Θ¹')

ax.set\_zlabel('J(Θ⁰, Θ¹ )')

plt.title("J(Θ⁰, Θ¹ ) vs Θ⁰ vs Θ¹")

plt.show()

#The minimum of j(Θ⁰, Θ¹ ) will be the best fitting model.

min\_index = np.argmin(cost)

i, j = np.unravel\_index(min\_index, cost.shape)

theta\_0\_opt = theta\_0[i]

theta\_1\_opt = theta\_1[j]

print("The best fitting model is f(x) = {} + {}x".format(theta\_0\_opt, theta\_1\_opt))



