```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
colnames=['areas','rooms','prices']
dataset = pd.read_csv("https://raw.githubusercontent.com/nishithkotak/machine-learning/refs/heads/master/ex1data2.txt",
                names=colnames)
```

```
dataset.describe()
```

	areas	rooms	prices
count	47.000000	47.000000	47.000000
mean	2000.680851	3.170213	340412.659574
std	794.702354	0.760982	125039.899586
min	852.000000	1.000000	169900.000000
25%	1432.000000	3.000000	249900.000000
50%	1888.000000	3.000000	299900.000000
75%	2269.000000	4.000000	384450.000000
max	4478.000000	5.000000	699900.000000

```
areas=dataset.iloc[0:dataset.shape[0],0:1]
romms=dataset.iloc[0:dataset.shape[0],1:2]
prices=dataset.iloc[0:dataset.shape[0],2:3]
```

```
dataset.shape
```

```
(47, 3)
```

```
from posixpath import splitdrive
#function normalization
def feature_normalization(x):
  mean=np.mean(x,axis=0)
  std=np.std(x,axis=0)
  x_normalized=(x-mean)/std
  return x_normalized,mean,std
```

```
data_norm = dataset.values
m = data_norm.shape[0]
#taking features vectors
x2 = data_norm[:, 0:2].reshape(m, 2)
x2_norm, mean, std = feature_normalization(x2)

y2 = data_norm[:, 2:3].reshape(m, 1)
```

```
x2_norm
```

```
array([[ 1.31415422e-01, -2.26093368e-01],
       [-5.09640698e-01, -2.26093368e-01],
       [ 5.07908699e-01, -2.26093368e-01],
       [-7.43677059e-01, -1.55439190e+00],
       [ 1.27107075e+00,  1.10220517e+00],
       [-1.99450507e-02,  1.10220517e+00],
       [-5.93588523e-01, -2.26093368e-01],
       [-7.29685755e-01, -2.26093368e-01],
       [-7.89466782e-01, -2.26093368e-01],
       [-6.44465993e-01, -2.26093368e-01],
       [-7.71822042e-02,  1.10220517e+00],
       [-8.65999486e-04, -2.26093368e-01],
       [-1.40779041e-01, -2.26093368e-01],
       [ 3.15099326e+00,  2.43050370e+00],
       [-9.31923697e-01, -2.26093368e-01],
       [ 3.80715024e-01,  1.10220517e+00],
       [-8.65782986e-01, -1.55439190e+00],
       [-9.72625673e-01, -2.26093368e-01],
       [ 7.73743478e-01,  1.10220517e+00],
       [ 1.31050078e+00,  1.10220517e+00],
       [-2.97227261e-01, -2.26093368e-01],
       [-1.43322915e-01, -1.55439190e+00],
       [-5.04552951e-01, -2.26093368e-01],
       [-4.91995958e-02,  1.10220517e+00],
       [ 2.40309445e+00, -2.26093368e-01],
       [-1.14560907e+00, -2.26093368e-01],
```

```
           [-6.90255715e-01, -2.26093368e-01],
           [ 6.68172729e-01, -2.26093368e-01],
           [ 2.53521350e-01, -2.26093368e-01],
           [ 8.09357707e-01, -2.26093368e-01],
           [-2.05647815e-01, -1.55439190e+00],
           [-1.27280274e+00, -2.88269044e+00],
           [ 5.00114703e-02,  1.10220517e+00],
           [ 1.44532608e+00, -2.26093368e-01],
           [-2.41262044e-01,  1.10220517e+00],
           [-7.16966387e-01, -2.26093368e-01],
           [-9.68809863e-01, -2.26093368e-01],
           [ 1.67029651e-01,  1.10220517e+00],
           [ 2.81647389e+00,  1.10220517e+00],
           [ 2.05187753e-01,  1.10220517e+00],
           [-4.28236746e-01, -1.55439190e+00],
           [ 3.01854946e-01, -2.26093368e-01],
           [ 7.20322135e-01,  1.10220517e+00],
           [-1.01841540e+00, -2.26093368e-01],
           [-1.46104938e+00, -1.55439190e+00],
           [-1.89112638e-01,  1.10220517e+00],
           [-1.01459959e+00, -2.26093368e-01]])
```

```
theta_array=np.zeros((3,1))
```

```
def Hypothesis(theta_array , x1 , x2) :
  return theta_array[0] + theta_array[1]*x1 + theta_array[2]*x2
```

```
def Cost_Function(theta_array,x1,x2,y,m):
  total_cost = 0
  for i in range(m):
    total_cost += (Hypothesis(theta_array,x1[i] , x2[i]) - y[i])**2
    return total_cost/(2*m)
```

```
def Gradient_Descent(theta_array , x1, x2, y , m ,alpha) :
  summation_0 = 0
  summation_1 = 0
  summation_2 = 0
  for i in range(m):
    summation_0 += (Hypothesis(theta_array,x1[i] , x2[i]) - y[i])
    summation_1 += ((Hypothesis(theta_array,x1[i] , x2[i]) - y[i])*x1[i])
    summation_2 += ((Hypothesis(theta_array,x1[i] , x2[i]) - y[i])*x2[i])
    new_theta0 = theta_array[0] - (alpha/m)*summation_0
    new_theta1 = theta_array[1] - (alpha/m)*summation_1
    new_theta2 = theta_array[2] - (alpha/m)*summation_2
    new_theta = [new_theta0 , new_theta1 , new_theta2]
    return new_theta
```

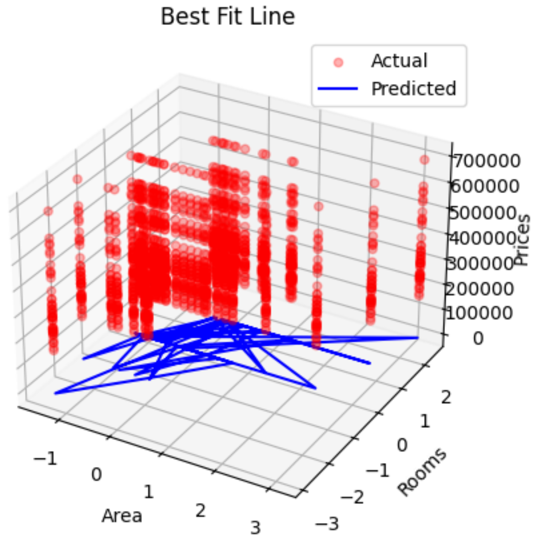
```
def Training(x1, x2, y, alpha, iters):
  theta_0 = 0
  theta_1 = 0
  theta_2 = 0
  theta_array = [theta_0, theta_1 ,theta_2]
  m = len(x1)
  cost_values = []
  for i in range(iters):
    theta_array = Gradient_Descent(theta_array, x1 ,x2, y, m, alpha)
    loss = Cost_Function(theta_array, x1 ,x2, y, m)
    cost_values.append(loss)
    y_new = theta_array[0] + theta_array[1]*x1 + theta_array[2]*x2
    return theta_array , cost_values
```

```
alpha = 0.01
iters = 500

area_norm = x2_norm[:, 0]
room_norm = x2_norm[:, 1]
price_norm = y2
theta_array, cost_per_itr = Training(area_norm, room_norm, price_norm, alpha, iters)
predicted_price = theta_array[0] + theta_array[1]*area_norm + theta_array[2]*room_norm

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(area_norm, room_norm, price_norm, alpha=0.3, c='#FF0000', label="Actual")
ax.plot(area_norm, room_norm, predicted_price, c="#0000FF", label="Predicted")
ax.set_xlabel("Area")
ax.set_ylabel("Rooms")
ax.set_zlabel("Prices")
ax.set_title("Best Fit Line")
plt.legend()
```

```
plt.show()
```




```
plt.figure(figsize=(8, 6))
plt.subplot(3, 1, 1)
sns.scatterplot(x='areas', y='rooms', data=dataset,palette='prices')
plt.title('Area vs Prices')
plt.xlabel('Area (sq ft)')
plt.ylabel('Prices ($)')
plt.subplot(3, 1, 2)
sns.scatterplot(x='rooms', y='prices', data=dataset, palette='viridis')
plt.title('Rooms vs Prices')
plt.xlabel('Number of Rooms')
plt.ylabel('Prices ($)')
plt.subplot(3, 1, 3)
sns.scatterplot(x='rooms', y='areas', data=dataset, palette='viridis')
plt.title('Rooms vs Area')
plt.xlabel('Number of Rooms')
plt.ylabel('Area (sq ft)')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-196695342.py:3: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(x='areas', y='rooms', data=dataset,palette='prices')
/tmp/ipython-input-196695342.py:8: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(x='rooms', y='prices', data=dataset, palette='viridis')
/tmp/ipython-input-196695342.py:13: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.scatterplot(x='rooms', y='areas', data=dataset, palette='viridis')
```