**Code:**

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()

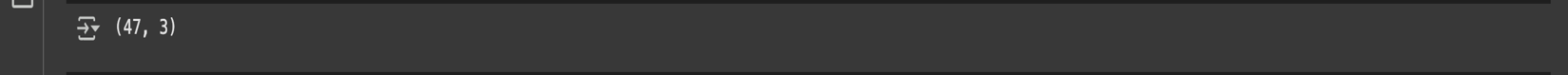


areas=dataset.iloc[0:dataset.shape[0],0:1]

romms=dataset.iloc[0:dataset.shape[0],1:2]

prices=dataset.iloc[0:dataset.shape[0],2:3]

dataset.shape



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

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

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")

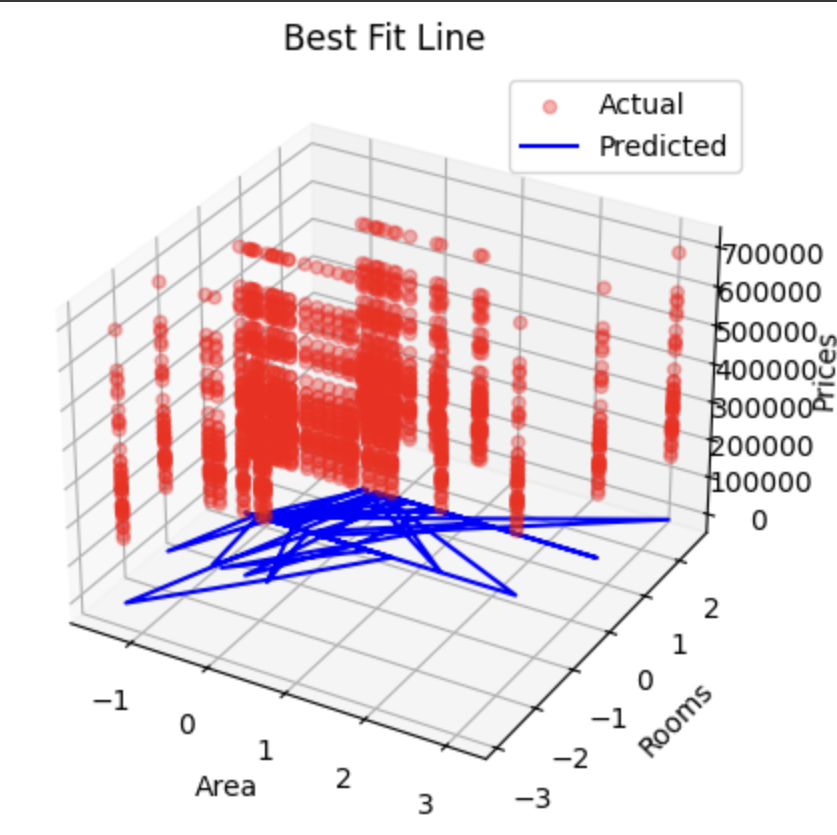
ax.set\_ylabel("Rooms")

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')

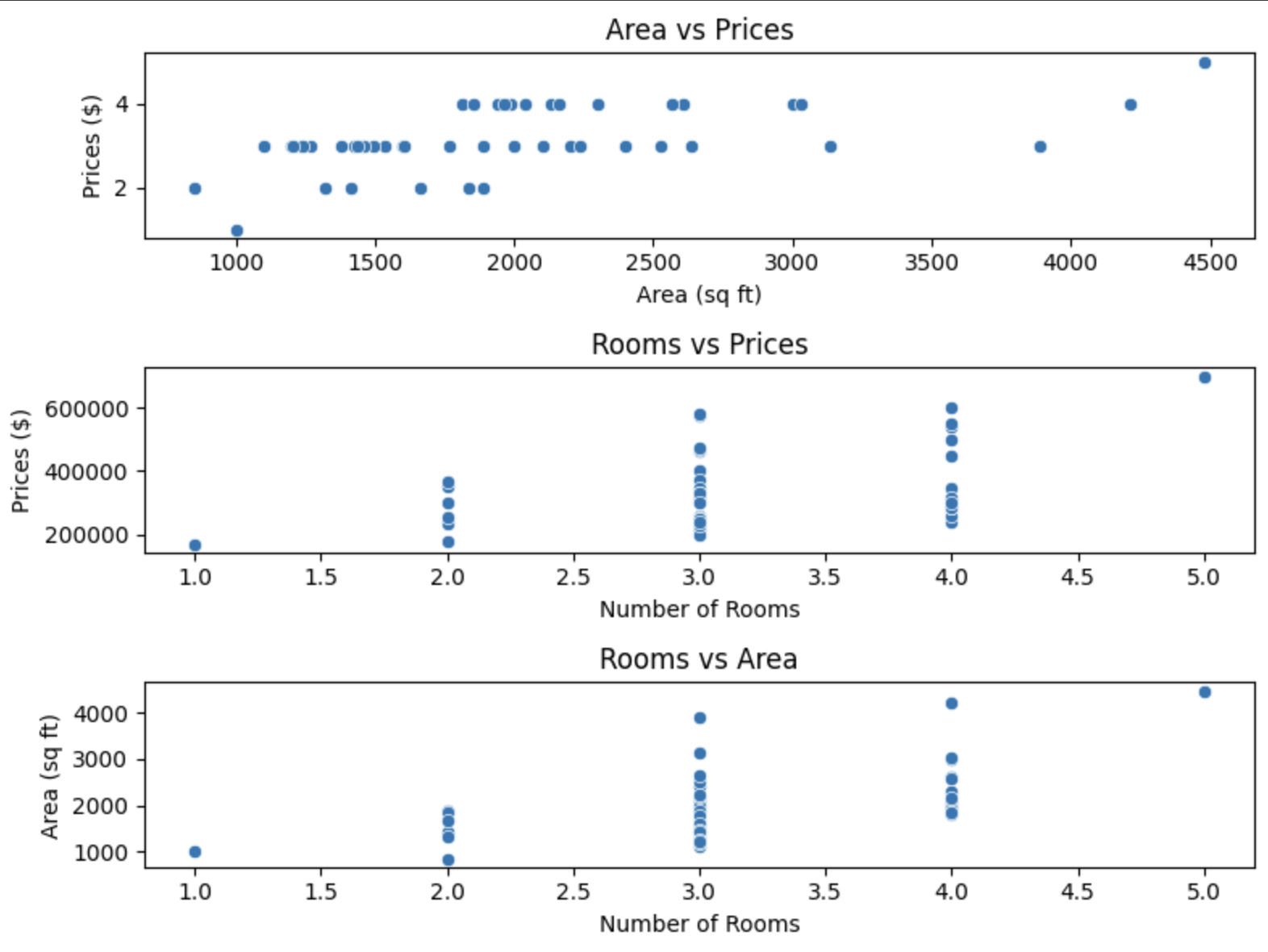
plt.title('Rooms vs Area')

plt.xlabel('Number of Rooms')

plt.ylabel('Area (sq ft)')

plt.tight\_layout()

plt.show()

****