Mruganshi Gohel

B20CS014

Lab-3

| import pandas as pd import numpy as np import matplotlib.pyplot as plt |
| --- |

Question-1

| data2= pd.read\_excel('2\_col.xlsx')  data2 |
| --- |

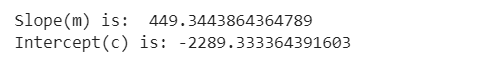
| dt2 = data2.values dt2 |
| --- |

| B=data2.values x,y=B[:,0],B[:,1] len(x) |
| --- |

| A=np.column\_stack((x,np.ones((len(x),1),dtype=float))) beta=np.dot(np.linalg.inv(np.dot(A.T,A)),np.dot(A.T,y.T)) |
| --- |

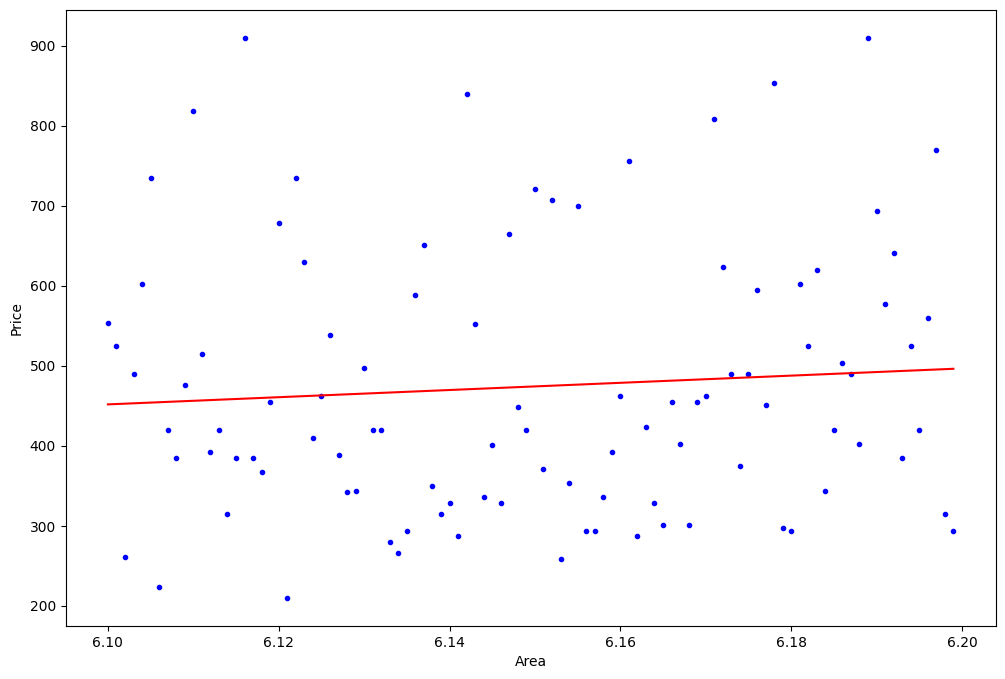
| plt.figure(figsize = (12, 8)) plt.plot(x, y, 'b.') plt.plot(x, beta[0]\*x + beta[1], 'r') plt.xlabel('Area') plt.ylabel('Price') plt.show() |
| --- |

| print("Slope(m) is: ", str(beta[0])) print("Intercept(c) is:", str(beta[1])) |
| --- |



| val = 4000; y = beta[0]\*val + beta[1] print("The predicted value for R = 4000 is:", y) |
| --- |



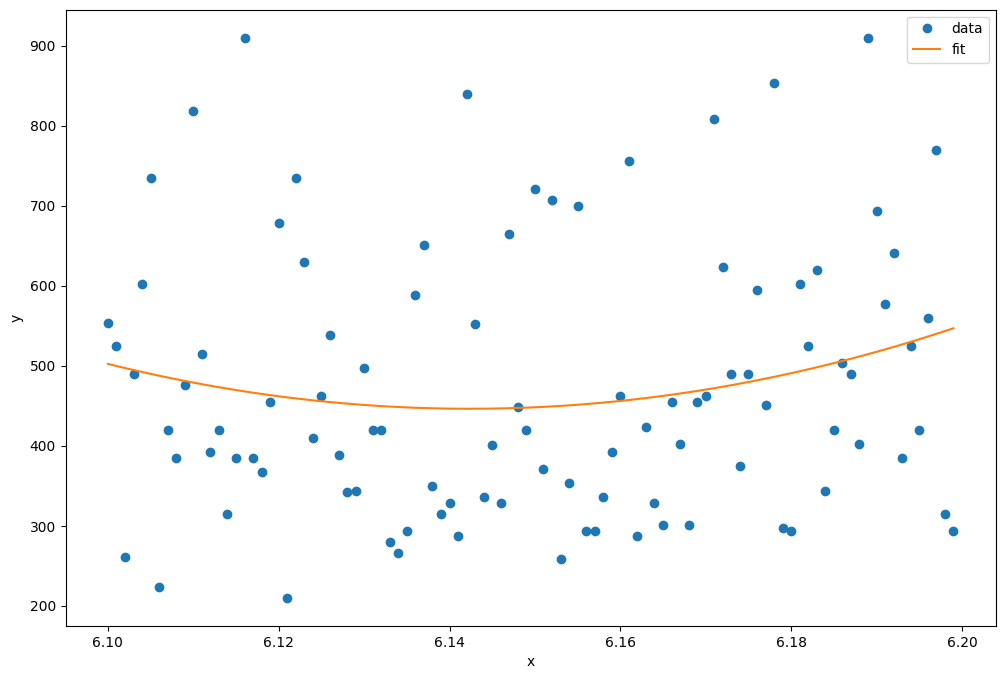


Question-2

Degree-2

| data2= pd.read\_excel('2-col.xlsx') Data2  B=data2.values |
| --- |

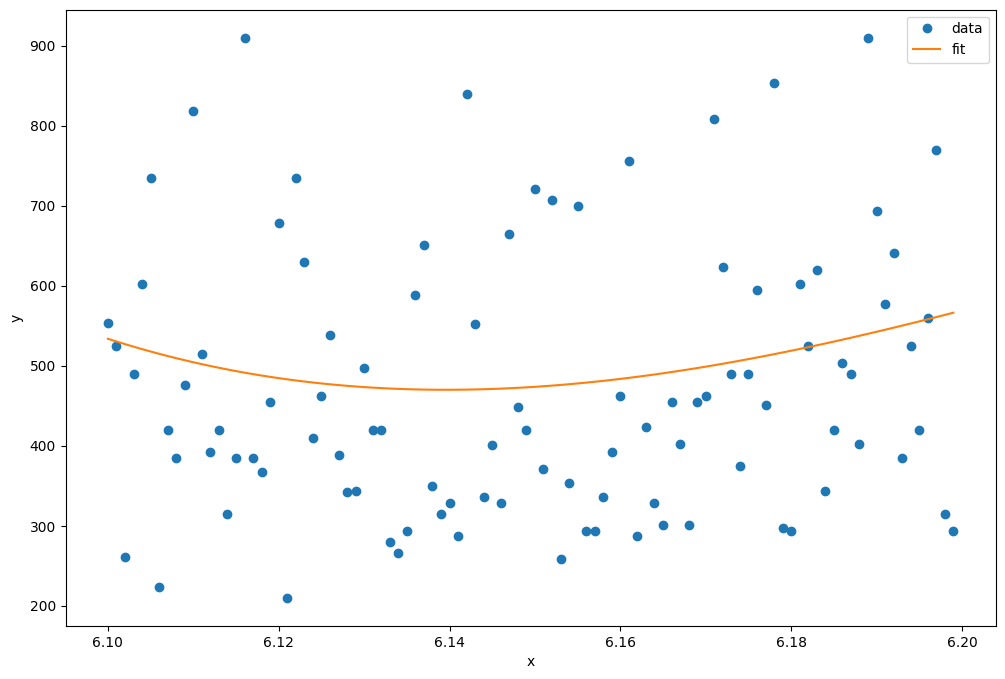
| x2 = B[:,0] y2 = B[:,1] A = np.column\_stack((x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*x2\*\*2 + beta[1]\*x2 + beta[2], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*2 + beta[1]\*val + beta[2] print("The predicted value for R = 4000 with degree 2 is:", y) |
| --- |





Degree-3

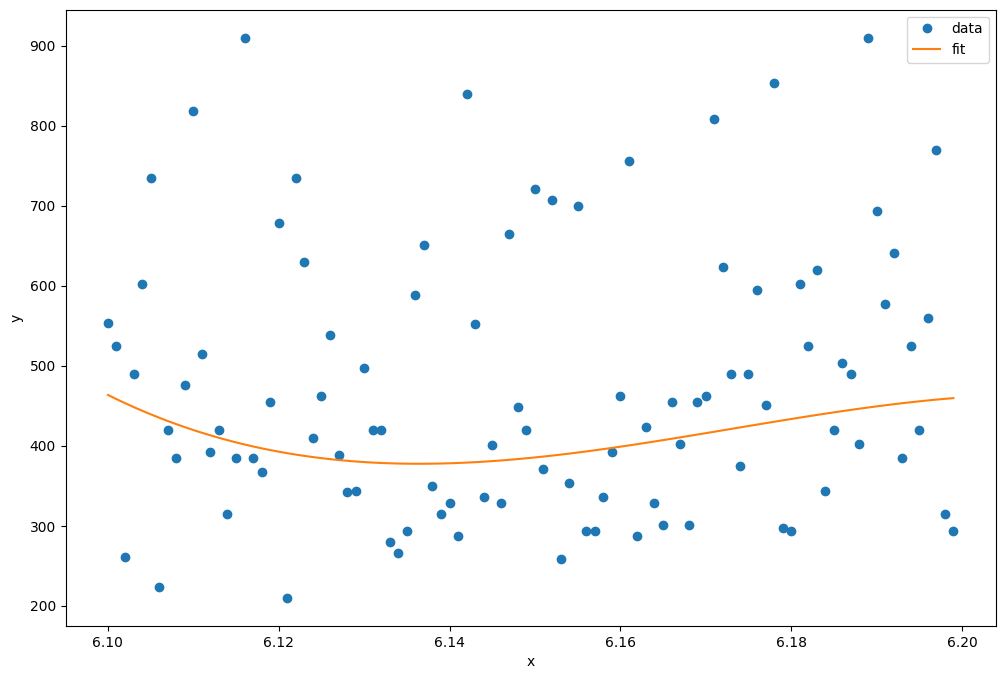
| x2=dt2[:,0] y2=dt2[:,1] A = np.column\_stack((x2\*\*3,x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*(x2\*\*3)+beta[1]\*(x2\*\*2) + beta[2]\*x2 + beta[3], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*3 + beta[1]\*val\*\*2 + beta[2]\*val + beta[3] print("The predicted value for R = 4000 with degree 3 is:", y) |
| --- |





Degree-4

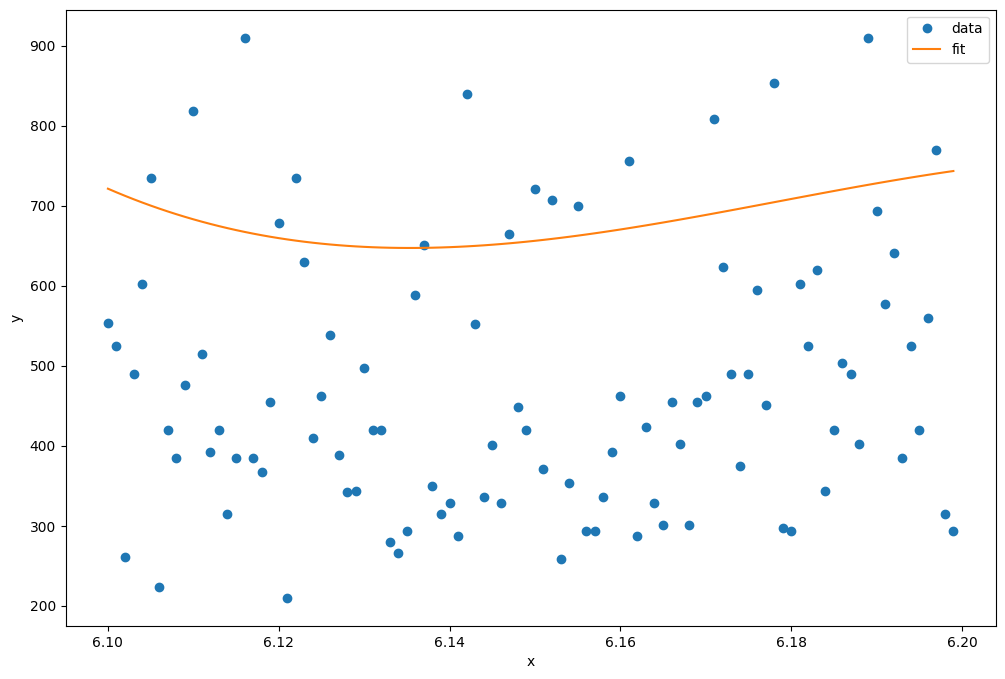
| x2=dt2[:,0] y2=dt2[:,1] A = np.column\_stack((x2\*\*4,x2\*\*3,x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*x2\*\*4 + beta[1]\*x2\*\*3 + beta[2]\*x2\*\*2 + beta[3]\*x2 + beta[4], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*4 + beta[1]\*val\*\*3 + beta[2]\*val\*\*2 + beta[3]\*val + beta[4] print("The predicted value for R = 4000 with degree 4 is:", y) |
| --- |





Degree-5

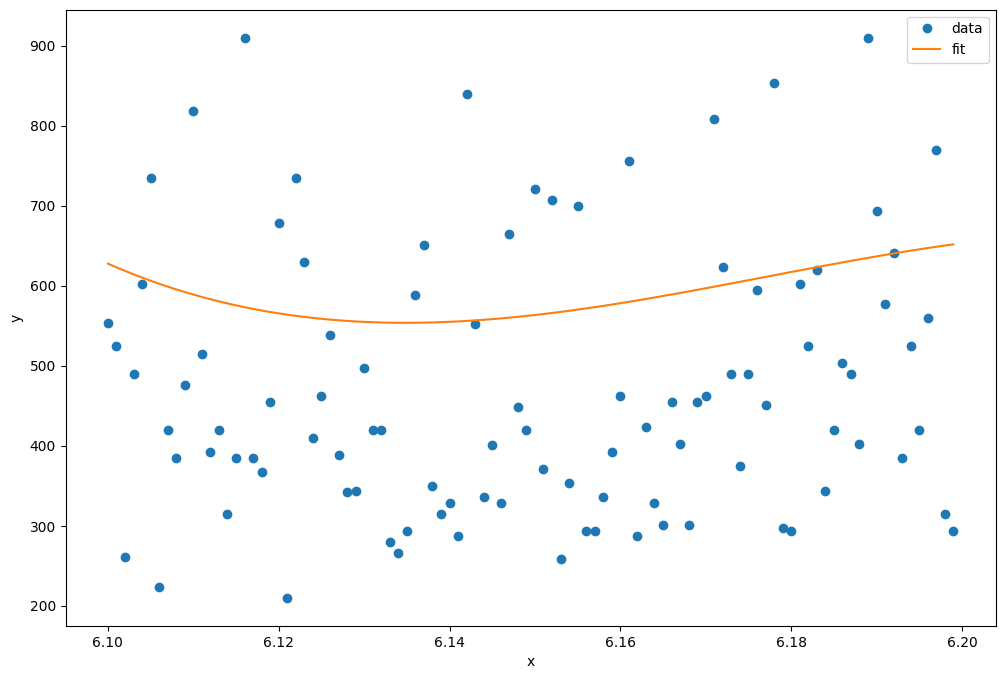
| x2=dt2[:,0] y2=dt2[:,1] A = np.column\_stack((x2\*\*5,x2\*\*4,x2\*\*3,x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*x2\*\*5+beta[1]\*x2\*\*4 + beta[2]\*x2\*\*3 + beta[3]\*x2\*\*2+ beta[4]\*x2+beta[5], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*5 + beta[1]\*val\*\*4 + beta[2]\*val\*\*3 + beta[3]\*val\*\*2 + beta[4]\*val + beta[5] print("The predicted value for R = 4000 with degree 5 is:", y) |
| --- |





Degree-6

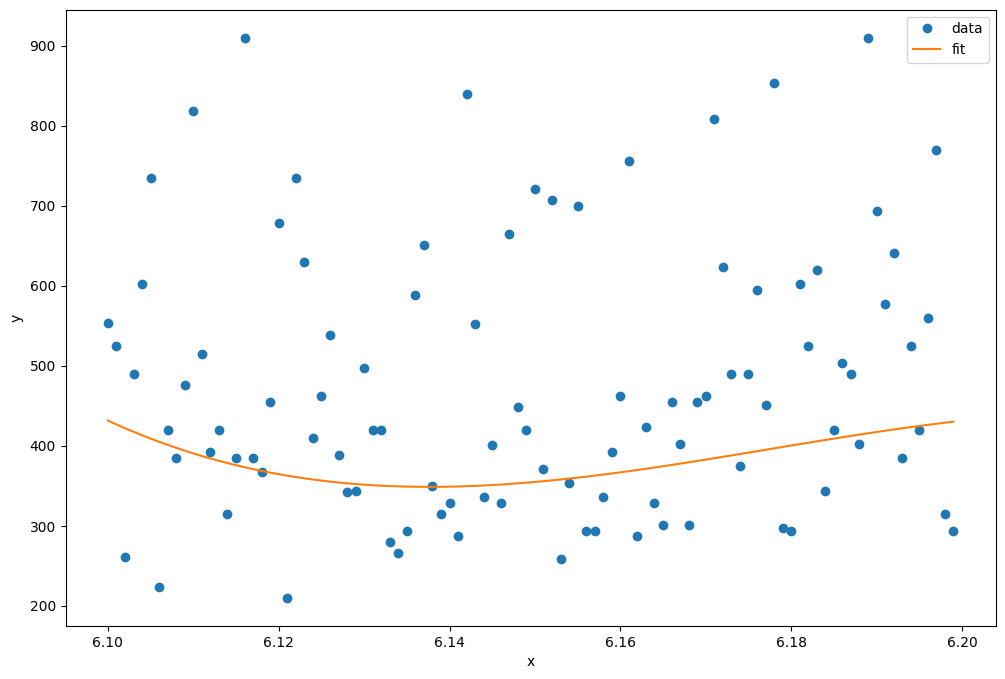
| x2=dt2[:,0] y2=dt2[:,1] A = np.column\_stack((x2\*\*6,x2\*\*5,x2\*\*4,x2\*\*3,x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*x2\*\*6+beta[1]\*x2\*\*5 + beta[2]\*x2\*\*4 + beta[3]\*x2\*\*3+ beta[4]\*x2\*\*2+beta[5]\*x2+beta[6], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*6 + beta[1]\*val\*\*5 + beta[2]\*val\*\*4 + beta[3]\*val\*\*3 + beta[4]\*val\*\*2 + beta[5]\*val + beta[6] print("The predicted value for R = 4000 with degree 6 is:", y) |
| --- |





Degree-7

| x2=dt2[:,0] y2=dt2[:,1] A = np.column\_stack((x2\*\*7,x2\*\*6,x2\*\*5,x2\*\*4,x2\*\*3,x2\*\*2, x2, np.ones((len(x2), 1), dtype = float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y2))  plt.figure(figsize = (12, 8)) plt.plot(x2, y2, 'o', label = 'data') plt.plot(x2, beta[0]\*x2\*\*7+beta[1]\*x2\*\*6 + beta[2]\*x2\*\*5 + beta[3]\*x2\*\*4+ beta[4]\*x2\*\*3+beta[5]\*x2\*\*2+beta[6]\*x2+beta[7], label = 'fit') plt.xlabel('x') plt.ylabel('y') plt.legend() plt.show()  val = 4000; y = beta[0]\*val\*\*7 + beta[1]\*val\*\*6 + beta[2]\*val\*\*5 + beta[3]\*val\*\*4 + beta[4]\*val\*\*3 + beta[5]\*val\*\*2 + beta[6]\*val + beta[7] print("The predicted value for R = 4000 with degree 7 is:", y) |
| --- |





Question-3

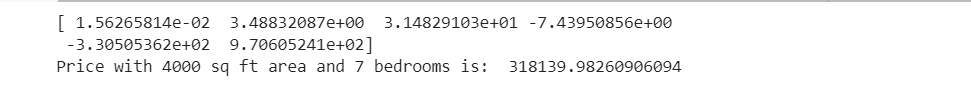
| data4= pd.read\_csv('6 columns - Copy.csv') data4 |
| --- |

| B=data4.values x,y,z=B[:,1],B[:,2],B[:,0] len(z) |
| --- |

| A = np.column\_stack((x, y, np.ones((len(x), 1), dtype=float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, z)) house\_area = 4000 house\_bedrooms = 7  # Predict the house price using the fitted model predicted\_price = beta[0] \* house\_area + beta[1] \* house\_bedrooms + beta[2] print("The predicted value for a house with", house\_area, "square feet and", house\_bedrooms, "bedrooms is:", predicted\_price) |
| --- |

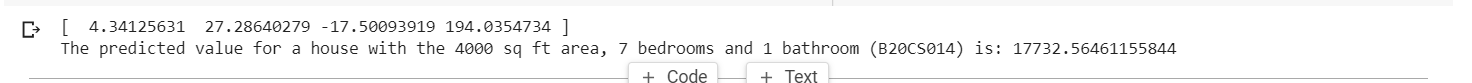
Question-4

| B=data4.values x,y,z=B[:,1],B[:,2],B[:,0] A = np.column\_stack((x\*x, x\*y,y\*y,x,y, np.ones((len(x), 1), dtype=float))) beta = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, z)) print(beta) pred\_area=4000 pred\_bedrooms = 7 price= beta[0]\*pred\_area\*\*2 + beta[1]\*pred\_area\*pred\_bedrooms + beta[2]\*pred\_bedrooms\*\*2 + beta[3]\*pred\_area + beta[4]\*pred\_bedrooms + beta[5] print("Price with 4000 sq ft area and 7 bedrooms is: ", price) |
| --- |



Question-5

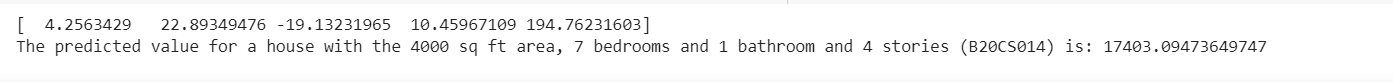
| B=data4.values x=data4.iloc[:,[1,2,3]] x=x.values y=B[:,0] A = np.column\_stack((x, np.ones((len(x), 1), dtype=float))) coefficients = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y) pred\_area = 4000 pred\_bedrooms=7 pred\_bathrooms=1 print(coefficients) predicted\_price = np.dot(coefficients[:-1], [pred\_area,pred\_bedrooms,pred\_bathrooms]) + coefficients[-1] print("The predicted value for a house with the 4000 sq ft area, 7 bedrooms and 1 bathroom (B20CS014) is:", predicted\_price) |
| --- |



Question-6

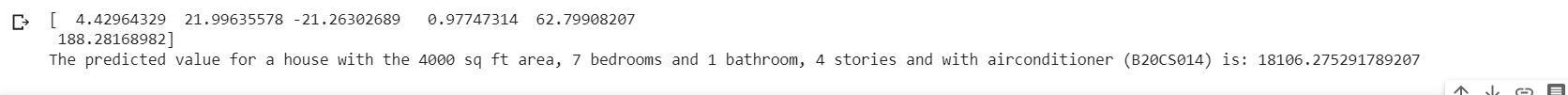
5-Column

| B=data4.values x=data4.iloc[:,[1,2,3,4]] x=x.values y=B[:,0] A = np.column\_stack((x, np.ones((len(x), 1), dtype=float))) coefficients = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y) pred\_area = 4000 pred\_bedrooms=7 pred\_bathrooms=1 pred\_stories=4 #assumed as not given what to take print(coefficients) predicted\_price = np.dot(coefficients[:-1], [pred\_area,pred\_bedrooms,pred\_bathrooms,pred\_stories]) + coefficients[-1] print("The predicted value for a house with the 4000 sq ft area, 7 bedrooms and 1 bathroom and 4 stories (B20CS014) is:", predicted\_price) |
| --- |



6-Columns

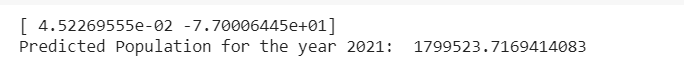
| B=data4.values x=data4.iloc[:,[1,2,3,4,5]] x=x.values y=B[:,0] A = np.column\_stack((x, np.ones((len(x), 1), dtype=float))) coefficients = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y) pred\_area = 4000 pred\_bedrooms=7 pred\_bathrooms=1 pred\_stories=4 #assumed as not given what to take pred\_airconditioning=1 #again assumed as not given print(coefficients) predicted\_price = np.dot(coefficients[:-1], [pred\_area,pred\_bedrooms,pred\_bathrooms,pred\_stories,pred\_airconditioning]) + coefficients[-1] print("The predicted value for a house with the 4000 sq ft area, 7 bedrooms and 1 bathroom, 4 stories and with airconditioner (B20CS014) is:", predicted\_price) |
| --- |



Question-7

| data5= pd.read\_csv('Census data (Chandigarh).csv') data5 |
| --- |

| B=data5.values x,y=B[:,0],B[:,1] z = np.log(y) A = np.column\_stack((x, np.ones((len(x), 1), dtype=float))) coefficients = np.dot(np.linalg.inv(np.dot(A.T,A)),np.dot(A.T,z.T)) print(coefficients) pred\_year=2021 m=coefficients[0] c=coefficients[1] pred\_ln\_population = m\*pred\_year + c pred\_population = np.exp(pred\_ln\_population) print("Predicted Population for the year 2021: ", pred\_population) |
| --- |



Question-8

| x,y=B[:,0],B[:,1] from sklearn.metrics import mean\_absolute\_error as mae  def polynomial\_fitting\_mplementation(x,y,p):  A=[]  i=p  while(i>-1):  A.append(pow(x.T,i))  i-=1  A=np.array(A)  A=A.T  beta=np.dot(np.linalg.inv(np.dot(A.T,A)),np.dot(A.T,y.T))  a=[]  i=p  while(i>-1):  a.append(pow(x,i))  i-=1  a=np.array(a)  ans=np.dot(beta,a)  return mae(y,ans) loss=[] for i in range(1,21):  loss.append(polynomial\_fitting\_mplementation(x,y,i)) plt.plot(loss) plt.show() print("Lowest average loss occurs at degree ", loss.index(min(loss)) , "polynomial and the avg loss is", min(loss)) |
| --- |

