CS428 (Advanced Operating Systems) / MA500 (Geometric Foundations of Data Analysis – Classical Techniques) – Assignment 1

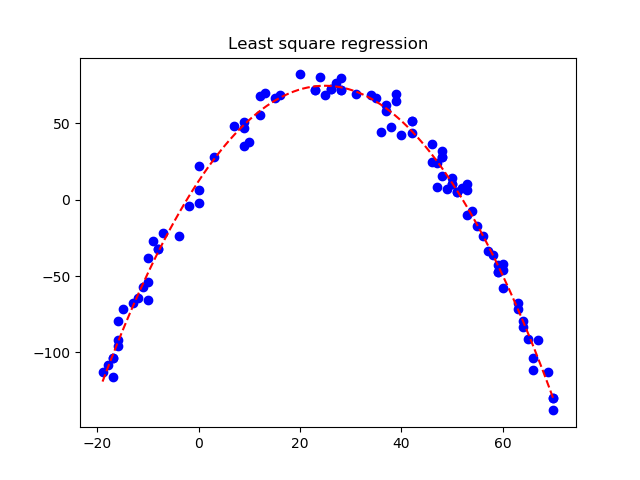
*Name: Taidgh Murray*

*Student ID: 15315901*

*Part 1 Outputs:*

Least Square estimator is:

For the model:



In general, the line of best fit is

With

And

SSR:

Define the Fitted Value as:

And the residual as:

These values are to also be used in Part 2.

My calculated b0 values are [12.34270521 5.00610241 -0.10068145]

SSR/SSTo = 0.5234696113226226

1 - SSE/SSTo = 0.52772458233925

Both these values are = to R^2. For the most part they are similar values

*Part 1 Code:*

1. #Student ID: 15315901
2. #Name: Taidgh Murray
3. #CS428/MA500 Homework 1
5. **import** math
6. **import** pylab
7. **import** numpy as np
8. **import** matplotlib.pyplot as plt
9. **from** scipy.optimize **import** curve\_fit
11. # Open the data file in a read format
12. f = open('data.txt', 'r')
14. # yi = b0+b1\*xi+b2\*xi^2+error(i)
16. # Initialise x & y arrays, along with error and beta value
18. Xs, Ys= [], []
19. b0 = [1.0, 1.0, 1.0]
21. **def** func(x, a, b, c):
22. **return** a +  b\*x + c\*x\*\*2

25. # For-loop
26. # Iterates through lines in file
27. # If the number is an x, it's added to the Xi array
28. # If the number is a y, it's added to the Yi Array
29. **for** l **in** f:
30. # Split the lines up by their = sign
31. # print(l)
32. sp = l.split('=')
34. # Splits the x value by its ',' character
35. xVal= sp[1].split(',')
37. # Adds corresponding x & y values in the line ot the arrays
38. Xs.append(float(xVal[0]))
39. Ys.append(float(sp[2]))
41. # Closes file
42. f.close()
44. # Changing X and Y data to numpy arrays
45. Xs = np.array(Xs)
46. Ys = np.array(Ys)
47. # Creating an arbitrary sigma array filled with ones to indicate error values
48. sigma = np.ones((100), dtype=float)

51. # Calculating the b0, b1 & b2 values
52. linear = np.linspace(Xs.min(), Xs.max(), 100)
53. B, \_ = curve\_fit(func, Xs, Ys, b0, sigma)
54. y = func(linear, \*B)

57. # Plotting & drawing the graph
59. #Plotting the points
60. plt.plot(Xs,Ys, 'bo', label='Data')
61. #Plotting the line of best fit
62. plt.plot(linear, y, 'r--', label='Fit')
63. plt.title('Least square regression')
64. #Drawing graph
65. plt.show()
67. **print**(B)
69. # Defining yHat - The fitted value
70. **def** yHat(i):
71. **return** b0[0] + b0[1]\*Xs[i]
73. # Defining yMean - The sample mean
74. yTotal = 0
76. **for** i **in** Ys:
77. yTotal += i
78. yMean = (1/len(Ys))\*(yTotal)
80. # Defining SSTO - The Total Sum of Squares
81. SSTO = 0
82. **for** i **in** Ys:
83. SSTO += (i - yMean)\*\*2
85. # Defining SSE - The Error Sum of Squares
86. SSE = 0
87. count = 0
88. **for** j **in** Ys:
89. SSE += (j - (yHat(count)))\*\*2
90. count+=1
92. # Defining SSR - The Regression Sum of Squares
93. count = 0
94. SSR = 0
95. **for** k **in** Ys:
96. SSR += ((yHat(count)) - yMean)\*\*2
97. count+=1
99. # Showcasing that r^2 = SSR/SSTO = (SSE/SSTO)-1 (Which, they largely are)
100. **print**('SSR/SSTo = ', SSR/SSTO)
101. **print**('1 - SSE/SSTo = ', (SSE/SSTO)-1)

*Part 2 Outputs:*

b0, b1, b2: [3.36508341 4.06510383 4.72071285]

MSE:

11133.1

F\* value:

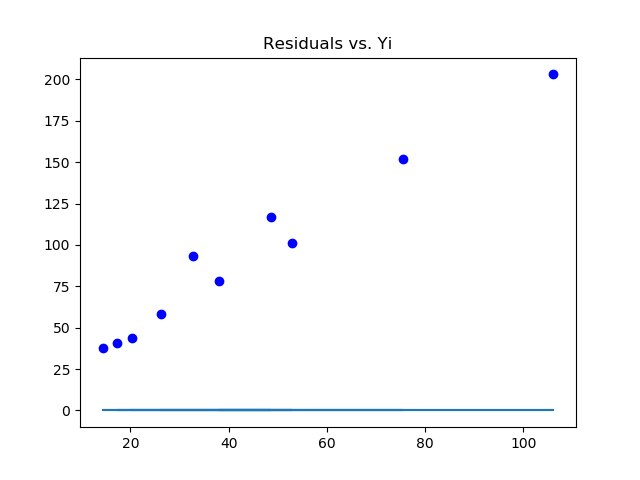
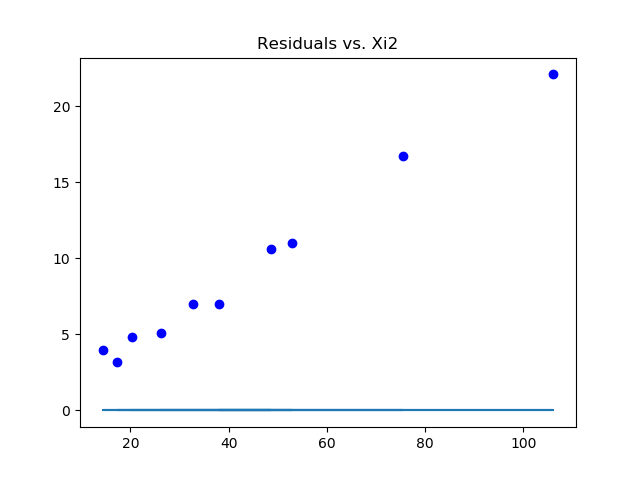
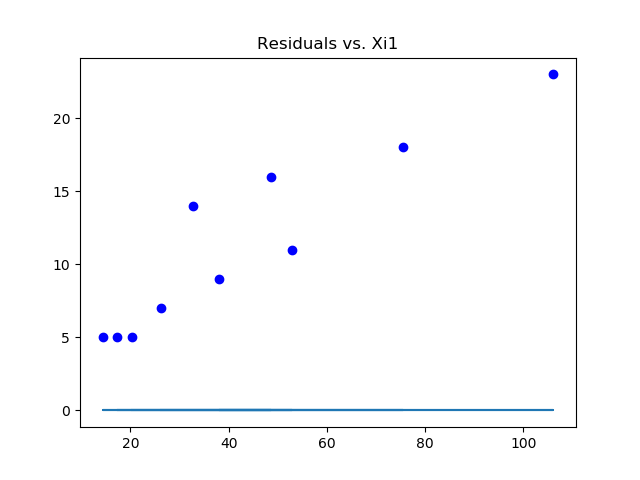
[-0.20144135 -1.38350205 -0.10066079 -0.51791505 -0.61085113 -0.08646888

-2.46765651 -0.36431901 -0.81971778 -0.11593057]

Define the Fitted Value as:

Define the residual as:

Estimated Covariance matrix:



To obtain the residuals, plot each , Xi1, & Xi2 against . The residuals seem to be independent (as there is no systematic deviation) and not dependent on the level of or the xi1 or xi2 values. You can conclude that all values are independent in N(0, ).

1. *Part 2 Code:* # Assignment 1 - Question 2
3. #Student ID: 15315901
4. #Name: Taidgh Murray
5. #CS428/MA500 Homework 1
7. **import** math
8. **import** pylab
9. **import** numpy as np
10. **import** matplotlib.pyplot as plt
11. **from** scipy.optimize **import** curve\_fit, leastsq
12. **from** scipy.stats **import** t
14. # Defining data as np arrays
15. Xi1=np.array([7, 18, 5, 14, 11, 5, 23, 9, 16, 5])
16. Xi2=np.array([5.11, 16.70, 3.20, 7.00, 11.00, 4.00, 22.10, 7.00, 10.60, 4.80])
17. Yi=np.array([58, 152, 41, 93, 101, 38, 203, 78, 117, 44])
18. b0=np.array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
19. bZ = np.array([0.0,0.0,0.0])
21. # Model definition
22. **def** model(Beta, Xi1, Xi2, Error, i):
23. **return** Beta[0] + Beta[1]\*Xi1[i] + Beta[2]\*Xi2[i] + Error[i]
25. # Defining the least square estimator
26. **def** leastSquareEstimator(b,x, x2):
27. **return** b[0] + b[1]\*x[1] + b[2]\*x2[2]
29. # Defining the least square estimator
30. **def** func(X, a, b, c):
31. Xi1, Xi2 = X
32. **return** a + b\*Xi1 + c\*Xi2

35. # Calculating the b0, b1 & b2 values
36. B = (curve\_fit(func, (Xi1, Xi2), Yi, p0=bZ))
37. **print**('b0, b1, b2: ', B[0])
38. B0 = B[0]

41. # T value with 12 degrees of freedom & a significange level of 0.05
42. T = t.ppf(0.975, 12)
43. P = 3
45. #MSR = (1/p-1)(yTranspose\*y - bTranspose \* xTranspose \* y)

48. # Defining yHat - The fitted value
49. **def** yHat(i):
50. **return** B0[0] + B0[1]\*Xi1[i]
52. # Defining e(i) - The residual value
53. **def** e(i):
54. **return** (Yi[i] - yHat(i))
56. # Defining yMean - The sample mean
57. yTotal = 0
59. **for** i **in** Yi:
60. yTotal += i
62. yMean = (1/len(Yi))\*(yTotal)
64. # Defining SSTO - The Total Sum of Squares
65. SSTO = 0
66. **for** i **in** Yi:
67. SSTO += (i - yMean)\*\*2
69. # Defining SSE - The Error Sum of Squares
70. SSE = 0
71. count = 0
72. **for** j **in** Yi:
73. SSE += (j - float(yHat(count)))\*\*2
74. count+=1
76. # Defining SSR - The Regression Sum of Squares
77. count = 0
78. SSR = 0
79. **for** k **in** Yi:
80. SSR += (float(yHat(count)) - yMean)\*\*2
81. count+=1
83. # Defining MSR
84. MSR = (1/P-1) \* (Yi.transpose()\*Yi - b0.transpose()\*Xi1.transpose()\*Yi)
86. # Obtaining residuals
87. resVals=[]
88. **for** i **in** range(len(Yi)):
89. resVals.append(e(i))
91. # Plotting the residuals
92. plt.plot(resVals, Yi, 'bo', label='Yi')
93. plt.plot(resVals, b0)
94. plt.title('Residuals vs. Yi')
95. plt.show()
97. plt.plot(resVals, Xi1, 'bo', label='Yi')
98. plt.plot(resVals, b0)
99. plt.title('Residuals vs. Xi1')
100. plt.show()
102. plt.plot(resVals, Xi2, 'bo', label='Yi')
103. plt.plot(resVals, b0)
104. plt.title('Residuals vs. Xi2')
105. plt.show()
107. # Printing MSE
108. **print**('MSE:')
109. MSE = SSE/len(Yi)-P
110. **print**(MSE)
112. # Printing Fstar value as found in the notes
113. **print**('F\* value:')
114. FStar = MSR/MSE
115. **print**(FStar)