



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**НГТУ
НЭТИ** | **Факультет прикладной
математики и информатики**

Кафедра теоретической и прикладной информатики

Лабораторная работа №6
по дисциплине «Статистические методы анализа данных»

Студенты ИВАНОВ ВЛАДИСЛАВ (92)

ОБЕРШТ ЕЛЕНА (93)

Вариант 5

Преподаватель ПОПОВ АЛЕКСАНДР АЛЕКСАНДРОВИЧ

Новосибирск, 2022

1 Постановка задачи

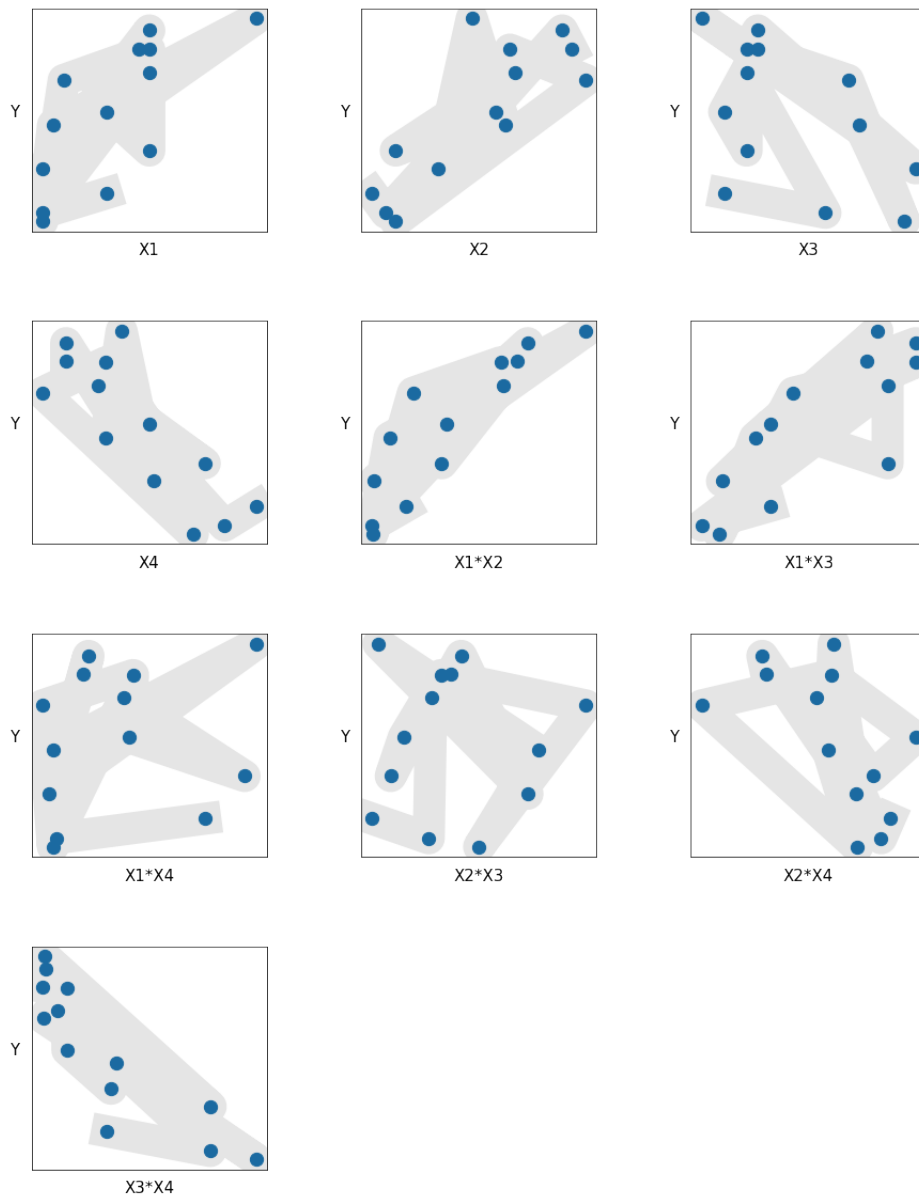
Модифицировать программу МНК-оценивания из ЛР №2 под реализацию алгоритма включения. Выбрать оптимальную модель для аппроксимации заданных экспериментальных данных.

	x1	x2	x3	x4	y
1	7.0	26.0	6.0	60.0	78.5
2	1.0	29.0	15.0	52.0	74.3
3	11.0	56.0	8.0	20.0	104.3
4	11.0	31.0	8.0	47.0	87.6
5	7.0	52.0	6.0	33.0	95.9
6	11.0	55.0	9.0	22.0	109.2
7	3.0	71.0	17.0	6.0	102.7
8	1.0	31.0	22.0	44.0	72.5
9	2.0	54.0	18.0	22.0	93.1
10	21.0	47.0	4.0	26.0	115.9
11	1.0	40.0	23.0	34.0	83.8
12	11.0	66.0	9.0	12.0	113.3
13	10.0	68.0	8.0	12.0	109.4

2 Анализ взаимосвязей

Вычислим коэффициент корреляции Пирсона для каждой пары признаков и их взаимодействий, построим корреляционную матрицу и графики зависимостей:

	x1	x2	x3	x4	x1*x2	x1*x3	x1*x4	x2*x3	x2*x4	x3*x4	y
x1	1.00	0.23	-0.82	-0.25	0.93	0.82	0.78	-0.61	-0.10	-0.69	0.73
x2	0.23	1.00	-0.14	-0.97	0.51	0.49	-0.32	0.46	-0.81	-0.72	0.82
x3	-0.82	-0.14	1.00	0.03	-0.75	-0.68	-0.76	0.78	-0.11	0.73	-0.53
x4	-0.25	-0.97	0.03	1.00	-0.52	-0.50	0.33	-0.53	0.84	0.64	-0.82
x1*x2	0.93	0.51	-0.75	-0.52	1.00	0.86	0.51	-0.42	-0.32	-0.77	0.89
x1*x3	0.82	0.49	-0.68	-0.50	0.86	1.00	0.50	-0.32	-0.39	-0.76	0.81
x1*x4	0.78	-0.32	-0.76	0.33	0.51	0.50	1.00	-0.80	0.36	-0.37	0.21
x2*x3	-0.61	0.46	0.78	-0.53	-0.42	-0.32	-0.80	1.00	-0.60	0.16	-0.01
x2*x4	-0.10	-0.81	-0.11	0.84	-0.32	-0.39	0.36	-0.60	1.00	0.49	-0.60
x3*x4	-0.69	-0.72	0.73	0.64	-0.77	-0.76	-0.37	0.16	0.49	1.00	-0.87
y	0.73	0.82	-0.53	-0.82	0.89	0.81	0.21	-0.01	-0.60	-0.87	1.00



По корреляционной таблице и графикам можно сделать вывод о том, что все зависимости, кроме $X_1 \times X_4$ и $X_2 \times X_3$, близки к линейным. В качестве класса допустимых решений F возьмем следующий:

$$f(x) = 1 + x_1 + x_2 + x_3 + x_4 + x_1x_2 + x_1x_3 + x_1x_4$$

$$n = 13$$

$$m = 8$$

3 Используемые критерии

$$C_p = \frac{RSS_p}{\hat{\sigma}^2} + 2p - n \rightarrow \min$$

$$R_p^2 = \frac{\sum (\hat{y}_p - \bar{\hat{y}})^2}{\sum (y_i - \bar{y})^2} \rightarrow 1$$

$$E_p = \frac{RSS_p}{n(n-p)} \left(1 + n + \frac{p(n+1)}{n-p-2} \right) \rightarrow \min$$

$$AEV_p = \frac{p \cdot RSS_p}{n(n-p)} \rightarrow \min$$

$$F_{ij} = \frac{v_2}{v_1} \cdot \frac{RSS_{ij} - RSS_j}{RSS_{i,j}}$$

$$RSS_p = (y - \hat{y}_p)^T (y - \hat{y}_p)$$

$$\hat{\sigma}^2 = \frac{RSS}{n-m}$$

$$v_1 = 1$$

$$v_2 = n - m$$

4 Алгоритм включения

$$f_1(x) = (1)^T$$

$$p = 1$$

	F11	F12	F13	F14	F15	F16	F17
0	5.73	9.98	2.0	10.36	18.63	9.58	0.23

$$f_2(x) = (1, x_1 x_2)^T$$

$$p = 2$$

	F11	F12	F13	F14	F16	F17
0	2.83	27.74	1.04	27.78	0.19	2.98

$$f_3(x) = (1, x_4, x_1 x_2)^T$$

$$p = 3$$

	F11	F12	F13	F16	F17
0	1.34	0.57	1.85	0.21	2.21

$$f_4(x) = (1, x_4, x_1 x_2, x_1 x_4)^T$$

$$p = 4$$

	F11	F12	F13	F16
0	0.34	1.05	1.01	0.0

$$f_5(x) = (1, x_2, x_4, x_1x_2, x_1x_4)^T$$

$$p = 5$$

	F11	F13	F16
0	0.47	0.01	0.01

$$f_6(x) = (1, x_1, x_2, x_4, x_1x_2, x_1x_4)^T$$

$$p = 6$$

	F13	F16
0	0.01	0.18

$$f_7(x) = (1, x_1, x_2, x_4, x_1x_2, x_1x_3, x_1x_4)^T$$

$$p = 7$$

	F13
0	0.18

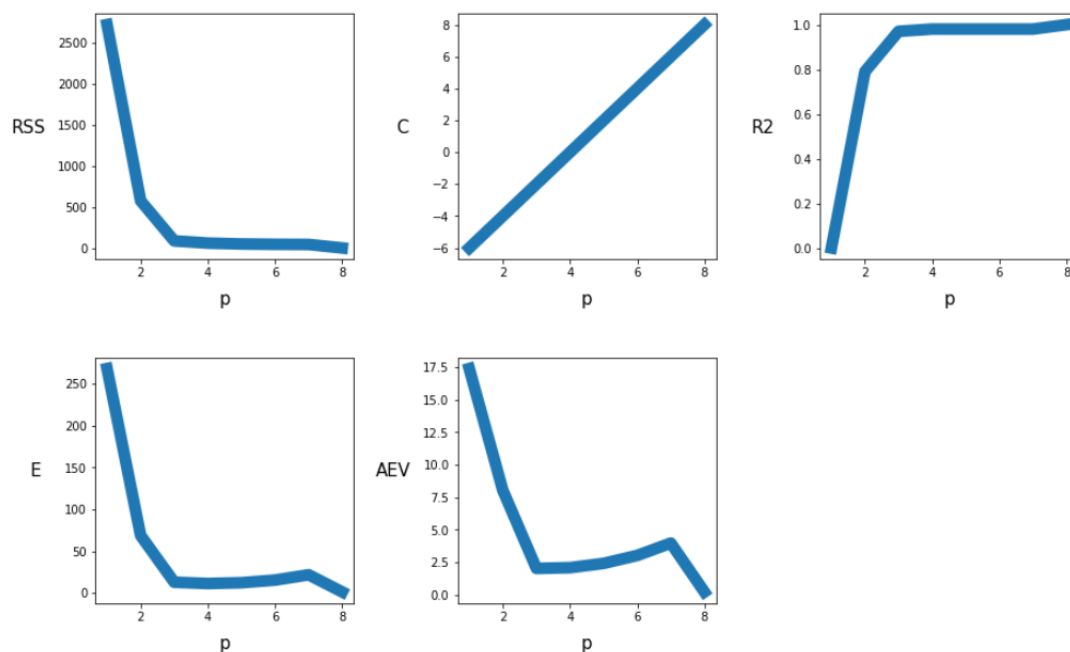
$$f_8(x) = (1, x_1, x_2, x_3, x_4, x_1x_2, x_1x_3, x_1x_4)^T$$

5 Исследование

Таблица значений критериев:

	RSS	C	R2	E	AEV
1	2715.76	-6.0	0.00	268.09	17.41
2	574.56	-4.0	0.79	68.75	8.04
3	87.63	-2.0	0.97	12.98	2.02
4	60.74	0.0	0.98	11.42	2.08
5	50.19	2.0	0.98	12.39	2.41
6	45.88	4.0	0.98	15.53	3.02
7	44.29	6.0	0.98	21.86	3.97
8	1.73	8.0	1.00	1.37	0.21

Графики зависимости значений критериев от шага:



Выберем лучшую модель:

$$f_4(x) = (1, x_4, x_1x_2, x_1x_4)^T$$

$$\hat{\theta} = (101.62, -0.58, 0.02, 0.01)^T$$

Сравним её предсказания с целевой переменной:

	y	y_hat	y - y_hat
1	78.5	77.032927	1.467073
2	74.3	72.619850	1.680150
3	104.3	106.708551	-2.408551
4	87.6	89.613044	-2.013044
5	95.9	93.813034	2.086966
6	109.2	105.645076	3.554924
7	102.7	103.006694	-0.306694
8	72.5	77.222237	-4.722237
9	93.1	91.772587	1.327413
10	115.9	116.352012	-0.452012
11	83.8	83.115939	0.684061
12	113.3	112.391284	0.908716
13	109.4	111.206767	-1.806767

6 Код программы

```
1 import pandas as pd
2 import numpy as np
3 import random
4 import scipy.stats
5 from matplotlib import pyplot as plt
6 from statsmodels.stats.outliers_influence import
  → variance_inflation_factor
7 np.set_printoptions(suppress=True)
8 random.seed(42)
9
10 x1 = [7.0,1.0,11.0,11.0,7.0,11.0,3.0,1.0,2.0,21.0,1.0,11.0,10.0]
11 x2 = [26.0,29.0,56.0,31.0,52.0,55.0,71.0,31.0,54.0,47.0,40.0,66.0,68.0]
12 x3 = [6.0,15.0,8.0,8.0,6.0,9.0,17.0,22.0,18.0,4.0,23.0,9.0,8.0]
13 x4 = [60.0,52.0,20.0,47.0,33.0,22.0,6.0,44.0,22.0,26.0,34.0,12.0,12.0]
14 y = [78.5,74.3,104.3,87.6,95.9,109.2,102.7,72.5,93.1,115.9,83.8,113.3,1
  → 09.4]
15
16 df = pd.DataFrame(list(zip(x1, x2, x3, x4, y)), columns=['x1', 'x2',
  → 'x3', 'x4', 'y'])
17 df.index += 1
18 print(df)
19
20 df2 = pd.DataFrame(list(zip(x1, x2, x3, x4, np.array(x1)*np.array(x2),
  → np.array(x1)*np.array(x3), np.array(x1)*np.array(x4),
  → np.array(x2)*np.array(x3), np.array(x2)*np.array(x4),
  → np.array(x3)*np.array(x4), y)), columns=['x1', 'x2', 'x3', 'x4',
  → 'x1*x2', 'x1*x3', 'x1*x4', 'x2*x3', 'x2*x4', 'x3*x4', 'y']))
21 corr = df2.corr()
22 corr.style.background_gradient(cmap='coolwarm').set_precision(2)
23
24 factors = [x1, x2, x3, x4, np.array(x1)*np.array(x2),
  → np.array(x1)*np.array(x3), np.array(x1)*np.array(x4),
  → np.array(x2)*np.array(x3), np.array(x2)*np.array(x4),
  → np.array(x3)*np.array(x4)]
25 factors_str = ['X1', 'X2', 'X3', 'X4', 'X1*X2', 'X1*X3', 'X1*X4',
  → 'X2*X3', 'X2*X4', 'X3*X4']
26
27 fig = plt.figure(figsize=(15,20))
28 fig.subplots_adjust(hspace=0.4, wspace=0.4)
29 for i in range(1, len(factors)+1):
30     ax = fig.add_subplot(4, 3, i)
31     ax.scatter(factors[i-1], y, s=150)
32     ax.plot(factors[i-1], y, 'k', lw=30, alpha=0.1)
33     ax.set(xlabel=factors_str[i-1], ylabel='Y')
34     ax.yaxis.label.set_size(15)
35     ax.xaxis.label.set_size(15)
36     ax.yaxis.label.set_rotation(0)
```

```

37     ax.yaxis.labelpad = 15
38     ax.xaxis.labelpad = 10
39     plt.tick_params(which='both', bottom=False, left=False,
    ↪     labelbottom=False, labelleft=False)
40
41 RSS_list, C_list, R2_list, E_list, AEV_list = [], [], [], [], []
42
43 n = len(y)
44 m = 8
45 nu1 = 1
46 nu2 = n - m
47 x0 = np.full((n,), 1.0)
48
49 p = 1 # !!!
50 F_list = []
51 X = np.array([x0]).T # !!!
52 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
53
54 y_hat = np.dot(X, theta)
55 RSS = sum((y - y_hat)**2)
56 sigma2 = RSS / nu2
57 C = (RSS / sigma2) + 2 * p - n
58 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
59 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
60 AEV = (p * RSS) / (n * (n - p))
61 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
    ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
    ↪ AEV_list.append(round(AEV, 2))
62
63 for i in [x1, x2, x3, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x3), np.array(x1)*np.array(x4)]: # !!!
64     X = np.array([x0, i]).T # !!!
65     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
66     RSS_new = sum((y - np.dot(X, theta))**2)
67
68     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
69     F_list.append(round(F, 2))
70
71 df = pd.DataFrame(F_list).T
72 df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F14', 4: 'F15', 5:
    ↪ 'F16', 6: 'F17'}, inplace=True) # !!!
73 print(df)
74
75 p = 2 # !!!
76 F_list = []
77 X = np.array([x0, np.array(x1)*np.array(x2)]).T # !!!
78 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
79
80 y_hat = np.dot(X, theta)

```



```

81 RSS = sum((y - y_hat)**2)
82 sigma2 = RSS / nu2
83 C = (RSS / sigma2) + 2 * p - n
84 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
85 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
86 AEV = (p * RSS) / (n * (n - p))
87 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
  ↳ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
  ↳ AEV_list.append(round(AEV, 2))
88
89 for i in [x1, x2, x3, x4, np.array(x1)*np.array(x3),
  ↳ np.array(x1)*np.array(x4)]: # !!!
90     X = np.array([x0, np.array(x1)*np.array(x2), i]).T # !!!
91     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
92     RSS_new = sum((y - np.dot(X, theta))**2)
93
94     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
95     F_list.append(round(F, 2))
96
97 df = pd.DataFrame(F_list).T
98 df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F14', 4: 'F16', 5:
  ↳ 'F17'}, inplace=True) # !!!
99 print(df)
100
101 p = 3 # !!!
102 F_list = []
103 X = np.array([x0, x4, np.array(x1)*np.array(x2)]).T # !!!
104 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
105
106 y_hat = np.dot(X, theta)
107 RSS = sum((y - y_hat)**2)
108 sigma2 = RSS / nu2
109 C = (RSS / sigma2) + 2 * p - n
110 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
111 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
112 AEV = (p * RSS) / (n * (n - p))
113 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
  ↳ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
  ↳ AEV_list.append(round(AEV, 2))
114
115 for i in [x1, x2, x3, np.array(x1)*np.array(x3),
  ↳ np.array(x1)*np.array(x4)]: # !!!
116     X = np.array([x0, x4, np.array(x1)*np.array(x2), i]).T # !!!
117     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
118     RSS_new = sum((y - np.dot(X, theta))**2)
119
120     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
121     F_list.append(round(F, 2))
122

```

```

123 df = pd.DataFrame(F_list).T
124 df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F16', 4: 'F17'},
125           ↪ inplace=True) # !!!
126 print(df)
127
128 p = 4 # !!!
129 F_list = []
130 X = np.array([x0, x4, np.array(x1)*np.array(x2),
131           ↪ np.array(x1)*np.array(x4)]).T # !!!
132 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
133
134 y_hat = np.dot(X, theta)
135 RSS = sum((y - y_hat)**2)
136 sigma2 = RSS / nu2
137 C = (RSS / sigma2) + 2 * p - n
138 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
139 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
140 AEV = (p * RSS) / (n * (n - p))
141 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
142           ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
143           ↪ AEV_list.append(round(AEV, 2))
144
145 for i in [x1, x2, x3, np.array(x1)*np.array(x3)]: # !!!
146     X = np.array([x0, x4, np.array(x1)*np.array(x2),
147           ↪ np.array(x1)*np.array(x4), i]).T # !!!
148     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
149     RSS_new = sum((y - np.dot(X, theta))**2)
150
151     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
152     F_list.append(round(F, 2))
153
154 df = pd.DataFrame(F_list).T
155 df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F16'},
156           ↪ inplace=True) # !!!
157 print(df)
158
159 p = 5 # !!!
160 F_list = []
161 X = np.array([x0, x2, x4, np.array(x1)*np.array(x2),
162           ↪ np.array(x1)*np.array(x4)]).T # !!!
163 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
164
165 y_hat = np.dot(X, theta)
166 RSS = sum((y - y_hat)**2)
167 sigma2 = RSS / nu2
168 C = (RSS / sigma2) + 2 * p - n
169 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
170 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
171 AEV = (p * RSS) / (n * (n - p))

```

```

165 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
    ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
    ↪ AEV_list.append(round(AEV, 2))
166
167 for i in [x1, x3, np.array(x1)*np.array(x3)]: # !!!
168     X = np.array([x0, x2, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x4), i]).T # !!!
169     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
170     RSS_new = sum((y - np.dot(X, theta))**2)
171
172     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
173     F_list.append(round(F, 2))
174
175 df = pd.DataFrame(F_list).T
176 df.rename(columns={0: 'F11', 1: 'F13', 2: 'F16'}, inplace=True) # !!!
177 print(df)
178
179 p = 6 # !!!
180 F_list = []
181 X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x4)]).T # !!!
182 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
183
184 y_hat = np.dot(X, theta)
185 RSS = sum((y - y_hat)**2)
186 sigma2 = RSS / nu2
187 C = (RSS / sigma2) + 2 * p - n
188 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
189 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
190 AEV = (p * RSS) / (n * (n - p))
191 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
    ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
    ↪ AEV_list.append(round(AEV, 2))
192
193 for i in [x3, np.array(x1)*np.array(x3)]: # !!!
194     X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x4), i]).T # !!!
195     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
196     RSS_new = sum((y - np.dot(X, theta))**2)
197
198     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
199     F_list.append(round(F, 2))
200
201 df = pd.DataFrame(F_list).T
202 df.rename(columns={0: 'F13', 1: 'F16'}, inplace=True) # !!!
203 print(df)
204
205 p = 7 # !!!
206 F_list = []

```

```

207 X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x3), np.array(x1)*np.array(x4)]).T # !!!
208 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
209
210 y_hat = np.dot(X, theta)
211 RSS = sum((y - y_hat)**2)
212 sigma2 = RSS / nu2
213 C = (RSS / sigma2) + 2 * p - n
214 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
215 E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
216 AEV = (p * RSS) / (n * (n - p))
217 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
    ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
    ↪ AEV_list.append(round(AEV, 2))
218
219 for i in [x3]: # !!!
220     X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x3), np.array(x1)*np.array(x4), i]).T #
    ↪ !!!
221     theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
222     RSS_new = sum((y - np.dot(X, theta))**2)
223
224     F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
225     F_list.append(round(F, 2))
226
227 df = pd.DataFrame(F_list).T
228 df.rename(columns={0: 'F13'}, inplace=True) # !!!
229 print(df)
230
231 p = 8 # !!!
232 F_list = []
233 X = np.array([x0, x1, x2, x3, x4, np.array(x1)*np.array(x2),
    ↪ np.array(x1)*np.array(x3), np.array(x1)*np.array(x4),
    ↪ np.array(x2)*np.array(x3), np.array(x2)*np.array(x4),
    ↪ np.array(x3)*np.array(x4)]).T # !!!
234 theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
235
236 y_hat = np.dot(X, theta)
237 RSS = sum((y - y_hat)**2)
238 sigma2 = RSS / nu2
239 C = (RSS / sigma2) + 2 * p - n
240 R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
241 E = RSS / (n * (n - p)) * (1 + n + p * (n + 1) / (n - p - 2))
242 AEV = (p * RSS) / (n * (n - p))
243 RSS_list.append(round(RSS, 2)), C_list.append(round(C, 2)),
    ↪ R2_list.append(round(R2, 2)), E_list.append(round(E, 2)),
    ↪ AEV_list.append(round(AEV, 2))
244
245 df = pd.DataFrame(list(zip(RSS_list, C_list, R2_list, E_list,
    ↪ AEV_list)), columns=['RSS', 'C', 'R2', 'E', 'AEV'])

```

```

246 df.index += 1
247 print(df)
248
249 lists = [RSS_list, C_list, R2_list, E_list, AEV_list]
250 lists_str = ['RSS', 'C', 'R2', 'E', 'AEV']
251 p_list = [1,2,3,4,5,6,7,8]
252
253 fig = plt.figure(figsize=(15,20))
254 fig.subplots_adjust(hspace=0.4, wspace=0.4)
255 for i in range(1, len(lists)+1):
256     ax2 = fig.add_subplot(4, 3, i)
257     ax2.plot(p_list, lists[i-1], lw=10, alpha=1)
258     ax2.set(xlabel='p', ylabel=lists_str[i-1])
259     ax2.yaxis.label.set_size(15)
260     ax2.xaxis.label.set_size(15)
261     ax2.yaxis.label.set_rotation(0)
262     ax2.yaxis.labelpad = 25
263     ax2.xaxis.labelpad = 10
264
265 X_best = np.array([x0, x4, np.array(x1)*np.array(x2),
266     ↪ np.array(x1)*np.array(x4)]).T
267 theta_best = np.dot(np.linalg.inv(np.dot(X_best.T, X_best)),
268     ↪ np.dot(X_best.T, y))
269 y_hat_best = np.dot(X_best, theta_best)
270 residuals_best = y - y_hat_best
271
272 print('theta_best =', theta_best)
273 df = pd.DataFrame(list(zip(y, y_hat_best, residuals_best)),
274     ↪ columns=['y', 'y_hat', 'y - y_hat'])
275 df.index += 1
276 df

```