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

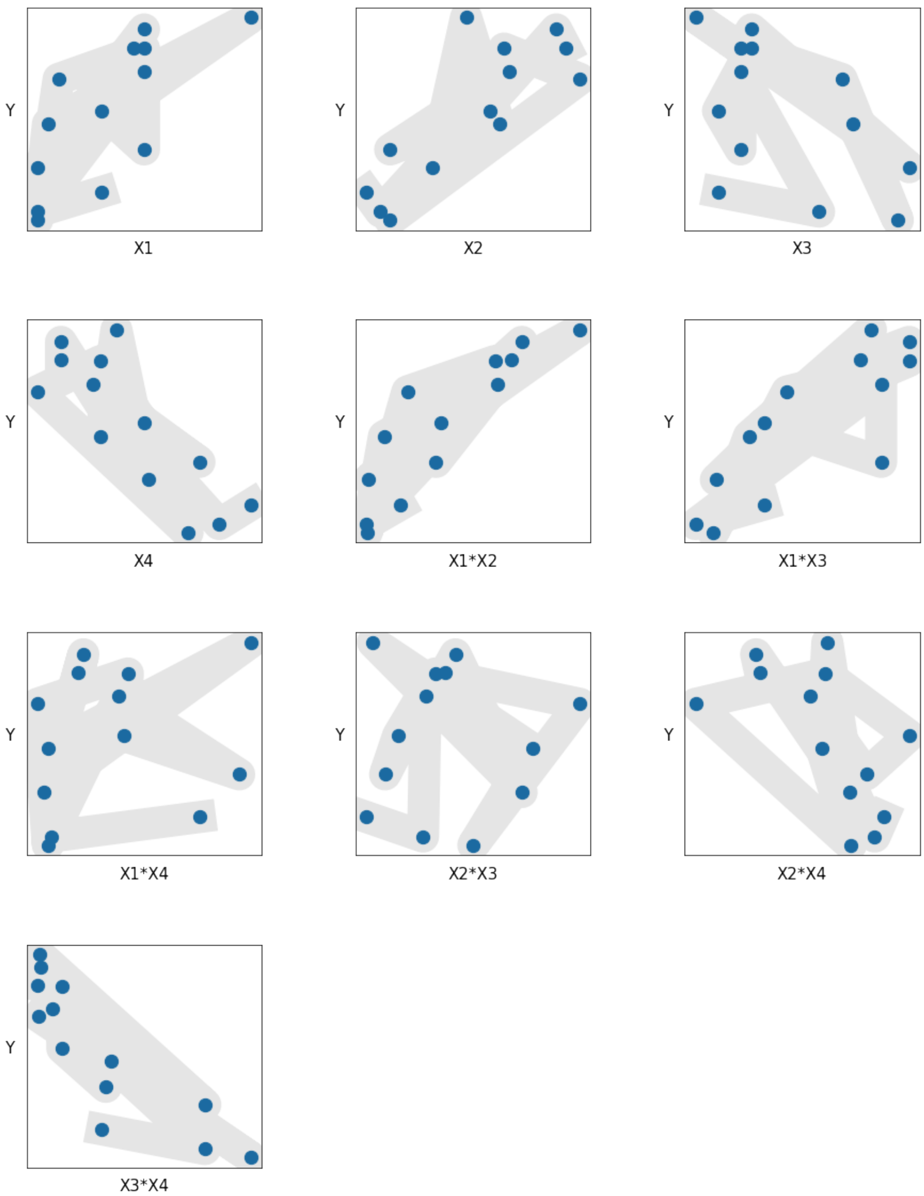
Модифицировать программу МНК-оценивания из ЛР №2 под реализацию алгоритма включения. Выбрать оптимальную модель для аппроксимации заданных экспериментальных данных.

	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 X1*X4 и X2*X3, 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

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

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

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

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

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

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

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

$$v_1=1$$

$$v_2=n-m$$

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

$$f_2(x)=(1,x_1x_2)^T$$

$$p=2$$

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

$$f_3(x)=(1,x_4,x_1x_2)^T$$

$$p=3$$

	F11	F12	F13	F16	F17
0	1.34	0.57	1.85	0.21	2.21

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

$$p = 4$$

	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

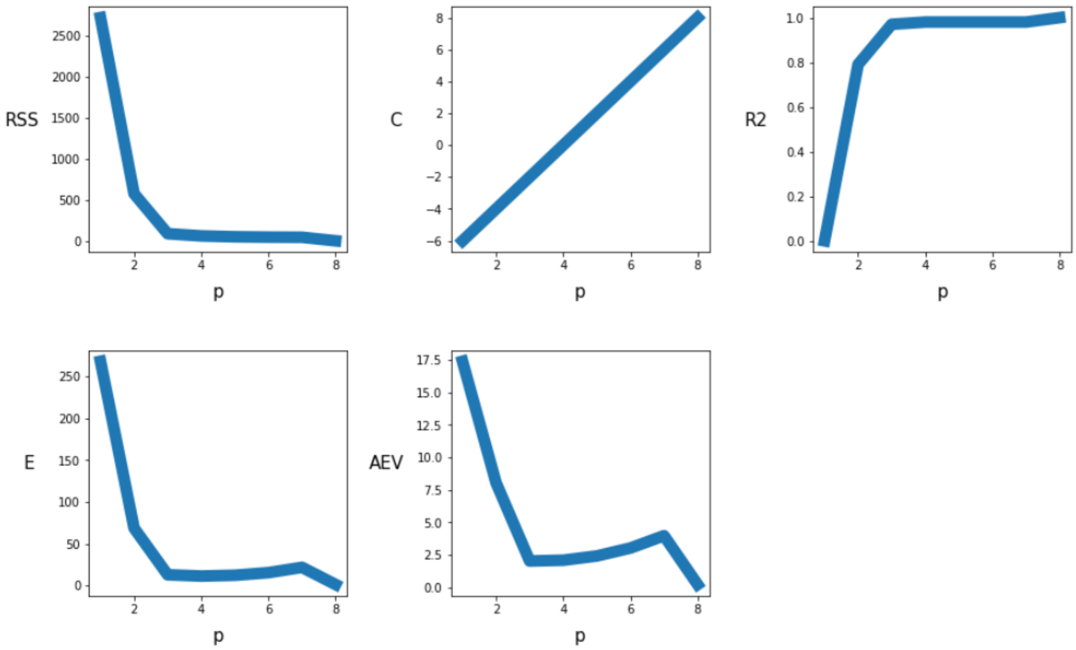
$$f_8(x) = (1, x_1, x_2, x_3, x_4, x_1x_2, x_1x_3, x_1x_4)^T$$

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

```python
import pandas as pd
import numpy as np
import random
import scipy.stats
from matplotlib import pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
np.set_printoptions(suppress=True)
random.seed(42)

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]
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]
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]
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]
y = [78.5,74.3,104.3,87.6,95.9,109.2,102.7,72.5,93.1,115.9,83.8,113.3,109.4]

df = pd.DataFrame(list(zip(x1, x2, x3, x4, y)), columns=['x1', 'x2', 'x3', 'x4', 'y'])
df.index += 1
print(df)

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'])
corr = df2.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)

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)]
factors_str = ['X1', 'X2', 'X3', 'X4', 'X1*X2', 'X1*X3', 'X1*X4', 'X2*X3', 'X2*X4', 'X3*X4']

fig = plt.figure(figsize=(15,20))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
for i in range(1, len(factors)+1):
    ax = fig.add_subplot(4, 3, i)
    ax.scatter(factors[i-1], y, s=150)
    ax.plot(factors[i-1], y, 'k', lw=30, alpha=0.1)
    ax.set(xlabel=factors_str[i-1], ylabel='Y')
    ax.yaxis.label.set_size(15)
    ax.xaxis.label.set_size(15)
    ax.yaxis.label.set_rotation(0)
```

```
    ax.yaxis.labelpad = 15
    ax.xaxis.labelpad = 10
    plt.tick_params(which='both', bottom=False, left=False,
     ↪  labelbottom=False, labelleft=False)

RSS_list, C_list, R2_list, E_list, AEV_list = [], [], [], [], []

n = len(y)
m = 8
nu1 = 1
nu2 = n - m
x0 = np.full((n,), 1.0)

p = 1 # !!!
F_list = []
X = np.array([x0]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

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)]: # !!!
    X = np.array([x0, i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F14', 4: 'F15', 5:
 ↪  'F16', 6: 'F17'}, inplace=True) # !!!
print(df)

p = 2 # !!!
F_list = []
X = np.array([x0, np.array(x1)*np.array(x2)]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
```

```
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

for i in [x1, x2, x3, x4, np.array(x1)*np.array(x3),
 ↪  np.array(x1)*np.array(x4)]: # !!!
    X = np.array([x0, np.array(x1)*np.array(x2), i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F14', 4: 'F16', 5:
 ↪  'F17'}, inplace=True) # !!!
print(df)

p = 3 # !!!
F_list = []
X = np.array([x0, x4, np.array(x1)*np.array(x2)]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

for i in [x1, x2, x3, np.array(x1)*np.array(x3),
 ↪  np.array(x1)*np.array(x4)]: # !!!
    X = np.array([x0, x4, np.array(x1)*np.array(x2), i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

```

```
df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F16', 4: 'F17'}, inplace=True) # !!!
print(df)

p = 4 # !!!
F_list = []
X = np.array([x0, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4)]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

for i in [x1, x2, x3, np.array(x1)*np.array(x3)]: # !!!
    X = np.array([x0, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4), i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F11', 1: 'F12', 2: 'F13', 3: 'F16'}, inplace=True) # !!!
print(df)

p = 5 # !!!
F_list = []
X = np.array([x0, x2, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4)]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
```

```
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))

for i in [x1, x3, np.array(x1)*np.array(x3)]: # !!!
    X = np.array([x0, x2, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4), i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F11', 1: 'F13', 2: 'F16'}, inplace=True) # !!!
print(df)

p = 6 # !!!
F_list = []
X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4)]).T # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

for i in [x3, np.array(x1)*np.array(x3)]: # !!!
    X = np.array([x0, x1, x2, x4, np.array(x1)*np.array(x2), np.array(x1)*np.array(x4), i]).T # !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F13', 1: 'F16'}, inplace=True) # !!!
print(df)

p = 7 # !!!
F_list = []
```

```
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 # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = (RSS / (n * (n - p))) * (1 + n + (p * (n + 1) / (n - p - 2)))
AEV = (p * RSS) / (n * (n - p))
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))

for i in [x3]: # !!!
    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 #
        !!!
    theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
    RSS_new = sum((y - np.dot(X, theta))**2)

    F = (nu2 / nu1) * (RSS - RSS_new) / RSS_new
    F_list.append(round(F, 2))

df = pd.DataFrame(F_list).T
df.rename(columns={0: 'F13'}, inplace=True) # !!!
print(df)

p = 8 # !!!
F_list = []
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 # !!!
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))

y_hat = np.dot(X, theta)
RSS = sum((y - y_hat)**2)
sigma2 = RSS / nu2
C = (RSS / sigma2) + 2 * p - n
R2 = sum((y_hat - np.mean(y_hat))**2) / sum((y - np.mean(y))**2)
E = RSS / (n * (n - p)) * (1 + n + p * (n + 1) / (n - p - 2))
AEV = (p * RSS) / (n * (n - p))
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))

df = pd.DataFrame(list(zip(RSS_list, C_list, R2_list, E_list,
    AEV_list)), columns=['RSS', 'C', 'R2', 'E', 'AEV'])
```

```python
df.index += 1
print(df)

lists = [RSS_list, C_list, R2_list, E_list, AEV_list]
lists_str = ['RSS', 'C', 'R2', 'E', 'AEV']
p_list = [1,2,3,4,5,6,7,8]

fig = plt.figure(figsize=(15,20))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
for i in range(1, len(lists)+1):
    ax2 = fig.add_subplot(4, 3, i)
    ax2.plot(p_list, lists[i-1], lw=10, alpha=1)
    ax2.set(xlabel='p', ylabel=lists_str[i-1])
    ax2.yaxis.label.set_size(15)
    ax2.xaxis.label.set_size(15)
    ax2.yaxis.label.set_rotation(0)
    ax2.yaxis.labelpad = 25
    ax2.xaxis.labelpad = 10

X_best = np.array([x0, x4, np.array(x1)*np.array(x2),
 ↪  np.array(x1)*np.array(x4)]).T
theta_best = np.dot(np.linalg.inv(np.dot(X_best.T, X_best)),
 ↪  np.dot(X_best.T, y))
y_hat_best = np.dot(X_best, theta_best)
residuals_best = y - y_hat_best

print('theta_best =', theta_best)
df = pd.DataFrame(list(zip(y, y_hat_best, residuals_best)),
 ↪  columns=['y', 'y_hat', 'y - y_hat'])
df.index += 1
df
```