

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №5:

«Проведення трьохфакторного експерименту при використанні рівняння регресії з
урахуванням квадратичних членів
(центральний ортогональний композиційний план)»

Виконала:
студентка групи ІВ-81
Дрозд С. В.
Залікова книжка № 8111
Перевірив Регіда П. Г.

Київ 2020р.

Лабораторна робота №5

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів.
(центральний ортогональний композиційний план)

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Виконання:

Варіант – 111.

111	-9	1	-2	3	-2	4
-----	----	---	----	---	----	---

Завдання

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$

$$y_{i\min} = 200 + x_{cp\min}$$

$$\text{где } x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}, \quad x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Код програми:

```
1. import random
2. import math
3. from _pydecimal import Decimal
4. from scipy.stats import f, t, ttest_ind, norm
5. from functools import reduce
6. from itertools import compress
7. import numpy as np
8.
9.
10. def generate_factors_table(row_array):
11.
12.     return [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2], \
13.         row[0] * row[1] * row[2]] + list(map(lambda x: round(x ** 2, 5), \
14.         row))] for row in row_array]
15.
16. def cochrans_criteria(m, N, y_table, p=0.95):
17.     print("Перевірка рівномірності дисперсій за критерієм Кохрена: ")
18.     y_variations = [np.var(i) for i in y_table]
19.     max_y_variation = max(y_variations)
20.     gp = max_y_variation/sum(y_variations)
21.     f1 = m - 1
22.     f2 = N
```

```

23.     q = 1-p
24.     gt = get_cochran_value(f1,f2, q)
25.     print(f"Gp = {gp:.3f}; Gt = {gt:.3f}; f1 = {f1}; f2 = {f2}; q = {q:.3f}")
26.     if gp < gt:
27.         print("Gp < Gt => дисперсії рівномірні ")
28.         return True
29.     else:
30.         print("Gp > Gt => дисперсії нерівномірні ")
31.         return False
32.
33. def x_i(i, raw_factors_table):
34.     try:
35.         assert i <= 10
36.     except:
37.         raise AssertionError("i must be smaller or equal 10")
38.     with_null_factor = list(map(lambda x: [1] + x, generate_factors_table(raw_factors_table)))
39.     res = [row[i] for row in with_null_factor]
40.     return np.array(res)
41.
42. def m_ij(*arrays):
43.     return np.average(reduce(lambda accum, el: accum*el, arrays))
44.
45. def calculate_theoretical_y(x_table, b_coefficients, importance):
46.
47.     x_table = [list(compress(row, importance)) for row in x_table]
48.     b_coefficients = list(compress(b_coefficients, importance))
49.     y_vals = np.array([sum(map(lambda x, b: x*b, row, b_coefficients)) for row in x_table])
50.     return y_vals
51.
52. def get_cochran_value(f1, f2, q):
53.     partResult1 = q / f2 # (f2 - 1)
54.     params = [partResult1, f1, (f2 - 1) * f1]
55.     fisher = f.isf(*params)
56.     result = fisher/(fisher + (f2 - 1))
57.     return Decimal(result).quantize(Decimal('.0001'))
58.
59. def get_student_value(f3, q):
60.     return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal('.0001'))
61.
62. def get_fisher_value(f3,f4, q):
63.     return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal('.0001'))
64.
65.
66. #згідно варіанту
67. x1min, x1max = -9, 1
68. x2min, x2max = -2, 3
69. x3min, x3max = -2, 4
70. x_avr_min = (x1min + x2min + x3min) / 3
71. x_avr_max = (x1max + x2max + x3max) / 3
72. m = 3
73. N = 15
74. ymin = 200 + x_avr_min
75. ymax = 200 + x_avr_max
76. p = 0.95
77.
78. x0 = [(x1max+x1min)/2, (x2max+x2min)/2, (x3max+x3min)/2]
79. detx = [abs(x1min - x0[0]), abs(x2min-x0[1]), abs(x3min-x0[2])]
80. l=1.215
81.

```

```

82.
83. raw_natur_table = [[x1min, x2min, x3min],
84.                     [x1min, x2max, x3max],
85.                     [x1max, x2min, x3max],
86.                     [x1max, x2max, x3min],
87.
88.                     [x1min, x2min, x3max],
89.                     [x1min, x2max, x3min],
90.                     [x1max, x2min, x3min],
91.                     [x1max, x2max, x3max],
92.
93.                     [-1*detx[0]+x0[0], x0[1], x0[2]],
94.                     [ 1*detx[0]+x0[0], x0[1], x0[2]],
95.                     [x0[0], -1*detx[1]+x0[1], x0[2]],
96.                     [x0[0], 1*detx[1]+x0[1], x0[2]],
97.                     [x0[0], x0[1], -1*detx[2]+x0[2]],
98.                     [x0[0], x0[1], 1*detx[2]+x0[2]],
99.                     [x0[0], x0[1], x0[2]]]
100.
101. raw_factors_table = [[-1, -1, -1],
102.                      [-1, +1, +1],
103.                      [+1, -1, +1],
104.                      [+1, +1, -1],
105.
106.                      [-1, -1, +1],
107.                      [-1, +1, -1],
108.                      [+1, -1, -1],
109.                      [+1, +1, +1],
110.
111.                      [-1.215, 0, 0],
112.                      [+1.215, 0, 0],
113.                      [0, -1.215, 0],
114.                      [0, +1.215, 0],
115.                      [0, 0, -1.215],
116.                      [0, 0, +1.215],
117.                      [0, 0, 0]]
118.
119.
120. factors_table = generate_factors_table(raw_factors_table)
121. print("Матриця кодованих значень X")
122. for row in factors_table:
123.     print(row)
124.
125. natur_table = generate_factors_table(raw_natur_table)
126. with_null_factor = list(map(lambda x: [1] + x, natur_table))
127.
128. y_arr = [[random.random()*(ymax-ymin) + ymin for i in range(m)] for j in range(N)]
129. while not cochrans_criteria(m, N, y_arr):
130.     m+=1
131.     y_arr = [[random.random()*(ymax - ymin) + ymin for i in range(m)] for j in range(N)]
132. y_i = np.array([np.average(row) for row in y_arr])
133. coefficients = [[m_ij(x_i(column, raw_factors_table)*x_i(row, raw_factors_table)) for column in range(11)] for
row in range(11)]
134. free_values = [m_ij(y_i, x_i(i, raw_factors_table)) for i in range(11)]
135. beta_coef = np.linalg.solve(coefficients, free_values)
136.
137. # -----Критерій Стюдента-----
138. y_table = y_arr
139. print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стюдента: ")

```

```

140. average_variation = np.average(list(map(np.var, y_table)))
141. y_averages = np.array(list(map(np.average, y_table)))
142. variation_beta_s = average_variation/N/m
143. standard_deviation_beta_s = math.sqrt(variation_beta_s)
144. x_vals = [x_i(i, raw_factors_table) for i in range(11)]
145. t_i = np.array([abs(beta_coef[i])/standard_deviation_beta_s for i in range(len(beta_coef))])
146. f3 = (m-1)*N
147. q = 1-p
148. t = get_student_value(f3, q)
149. importance = [True if el > t else False for el in list(t_i)]
150.
151. print("Оцінки коефіцієнтів  $\beta$ s: " + ", ".join(list(map(lambda x: str(round(float(x), 3)), beta_coef))), \
152.       "\nКоефіцієнти ts: " + ", ".join(list(map(lambda i: f"{i:.3f}", t_i))), \
153.       f"\nf3 = {f3}; q = {q:.3f}; табл = {t}")
154.
155. beta_i = [" $\beta_0$ ", " $\beta_1$ ", " $\beta_2$ ", " $\beta_3$ ", " $\beta_{12}$ ", " $\beta_{13}$ ", " $\beta_{23}$ ", " $\beta_{123}$ ", " $\beta_{11}$ ", " $\beta_{22}$ ", " $\beta_{33}$ "]
156. x_i_names = list(compress(["1", "x1", "x2", "x3", "x12", "x13", "x23", "x123", "x1^2", "x2^2", "x3^2"],
importance))
157. betas_to_print = list(compress(beta_coef, importance))
158.
159. for i in range(len(importance)):
160.     if not importance[i]:
161.         print(f"{beta_i[i]} = 0 - незначимий")
162. print("Рівняння регресії без незначимих членів: y = ", end="")
163. for i in range(len(betas_to_print)):
164.     print(f" {betas_to_print[i]:+.3f}*{x_i_names[i]}", end="")
165.
166. #-----критерій Фішера-----
167. d = len(list(filter(None, importance)))
168. y_table = y_arr
169. f3 = (m - 1) * N
170. f4 = N - d
171. q = 1-p
172.
173. theor_y = calculate_theoretical_y(natur_table, beta_coef, importance)
174. y_averages = np.array(list(map(np.average, y_table)))
175. s_ad = m/(N-d)*(sum((theor_y-y_averages)**2))
176. y_variations = np.array(list(map(np.var, y_table)))
177. s_v = np.average(y_variations)
178. f_p = float(s_ad/s_v)
179. f_t = get_fisher_value(f3, f4, q)
180.
181. print("\n\nПеревірка адекватності моделі за критерієм Фішера:", \
182.       "\nТеоретичні значення у для різних комбінацій факторів:")
183.
184. for i in range(len(natur_table)):
185.     print(f"x1 = {natur_table[i][1]:>6.3f}; x2 = {natur_table[i][2]:>6.3f}; "
186.           f"x3 = {natur_table[i][3]:>7.3f}; y = {theor_y[i]:>8.3f}")
187. print(f"\nFp = {f_p:.3f}, Ft = {f_t:.3f}", "\nFp < Ft => модель адекватна" if f_p < f_t else \
188.       "Fp > Ft => модель неадекватна")

```

Результат виконання роботи програми:

```
(venv) [kasper@kas-pc lab5-master]$ python main.py
Матриця кодованих значень X
[-1, -1, -1, 1, 1, 1, -1, 1, 1, 1]
[-1, 1, 1, -1, -1, 1, -1, 1, 1, 1]
[1, -1, 1, -1, 1, -1, -1, 1, 1, 1]
[1, 1, -1, 1, -1, -1, -1, 1, 1, 1]
[-1, -1, 1, 1, -1, -1, 1, 1, 1, 1]
[-1, 1, -1, -1, 1, -1, 1, 1, 1, 1]
[1, -1, -1, -1, -1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[-1.215, 0, 0, -0.0, -0.0, 0, -0.0, 1.47623, 0, 0]
[1.215, 0, 0, 0.0, 0.0, 0, 0.0, 1.47623, 0, 0]
[0, -1.215, 0, -0.0, 0, -0.0, -0.0, 0, 1.47623, 0]
[0, 1.215, 0, 0.0, 0, 0.0, 0.0, 0, 1.47623, 0]
[0, 0, -1.215, 0, -0.0, -0.0, -0.0, 0, 0, 1.47623]
[0, 0, 1.215, 0, 0.0, 0.0, 0.0, 0, 0, 1.47623]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Перевірка рівномірності дисперсій за критерієм Кохрена:
Gr = 0.225; Gt = 0.335; f1 = 2; f2 = 15; q = 0.050
Gr < Gt => дисперсії рівномірні

Перевірка значимості коефіцієнтів регресії за критерієм Стюдента:
Оцінки коефіцієнтів  $\beta$ s: 198.267, -0.047, 0.752, -0.408, -0.947, -0.395, 0.18, 0.279, 0.059, 1.089, -0.227
Коефіцієнти ts: 904.440, 0.214, 3.432, 1.859, 4.321, 1.802, 0.822, 1.271, 0.271, 4.966, 1.036
f3 = 30; q = 0.050; tтабл = 2.0423
 $\beta_1 = 0$  - незначимий
 $\beta_3 = 0$  - незначимий
 $\beta_{13} = 0$  - незначимий
 $\beta_{23} = 0$  - незначимий
 $\beta_{123} = 0$  - незначимий
 $\beta_{11} = 0$  - незначимий
 $\beta_{33} = 0$  - незначимий
Рівняння регресії без незначимих членів:  $y = +198.267 \cdot x_1 + 0.752 \cdot x_2 - 0.947 \cdot x_{12} + 1.089 \cdot x_2^2$ 

Перевірка адекватності моделі за критерієм Фішера:
Теоретичні значення y для різних комбінацій факторів:
x1 = -2.000; x2 = -2.000; x3 = 18.000; y = -1798.599
x1 = 3.000; x2 = 4.000; x3 = -27.000; y = -1729.877
x1 = -2.000; x2 = 4.000; x3 = -2.000; y = 214.904
x1 = 3.000; x2 = -2.000; x3 = 3.000; y = 203.011
x1 = -2.000; x2 = 4.000; x3 = 18.000; y = -1729.877
x1 = 3.000; x2 = -2.000; x3 = -27.000; y = -1798.599
x1 = -2.000; x2 = -2.000; x3 = -2.000; y = 203.011
x1 = 3.000; x2 = 4.000; x3 = 3.000; y = 214.904
x1 = 0.500; x2 = 1.000; x3 = -5.037; y = -1986.154
x1 = 0.500; x2 = 1.000; x3 = 1.038; y = 411.279
x1 = -2.538; x2 = 1.000; x3 = 10.150; y = -787.437
x1 = 3.538; x2 = 1.000; x3 = -14.150; y = -787.437
x1 = 0.500; x2 = -2.645; x3 = -2.000; y = -797.462
x1 = 0.500; x2 = 4.645; x3 = -2.000; y = -748.488
x1 = 0.500; x2 = 1.000; x3 = -2.000; y = -787.437
Fp = 3159458.186, Ft = 2.126 Fp > Ft => модель неадекватна
```

Висновок: у ході виконання лабораторної роботи №5 було проведено трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Було знайдено рівняння регресії, яке буде адекватним для опису об'єкту, результати наведені вище. Результати співпадають із калькулятором. Кінцева мета роботи досягнута.