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.

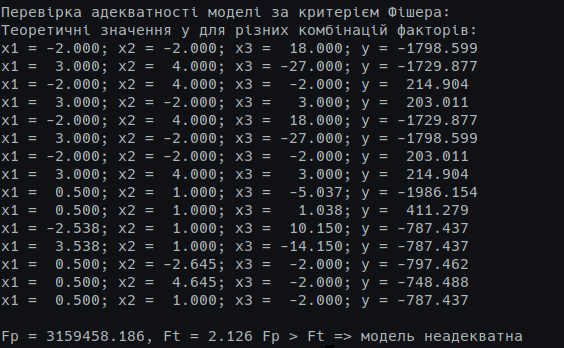
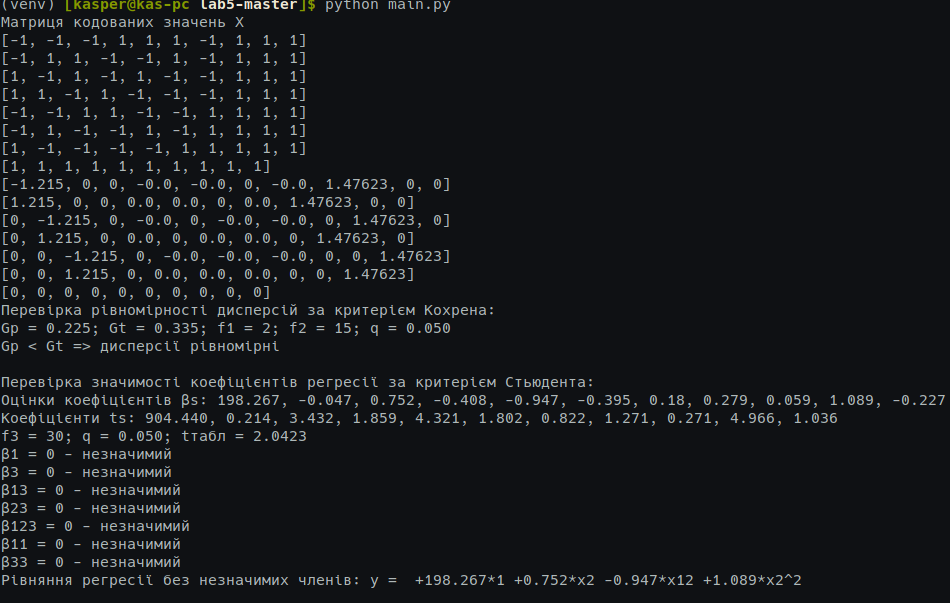




**Код програми:**

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. def generate\_factors\_table(raw\_array):
9. return [row + [row[0] \* row[1], row[0] \* row[2], row[1] \* row[2], \
10. row[0] \* row[1] \* row[2]]+list(map(lambda x: round(x \*\* 2, 5),\
11. row))for row in raw\_array]
12. def cochran\_criteria(m, N, y\_table, p=0.95):
13. print("Перевірка рівномірності дисперсій за критерієм Кохрена: ")
14. y\_variations = [np.var(i) for i in y\_table]
15. max\_y\_variation = max(y\_variations)
16. gp = max\_y\_variation/sum(y\_variations)
17. f1 = m - 1
18. f2 = N
19. q = 1-p
20. gt = get\_cochran\_value(f1,f2, q)
21. print(f"Gp = {gp:.3f}; Gt = {gt:.3f}; f1 = {f1}; f2 = {f2}; q = {q:.3f}")
22. if gp < gt:
23. print("Gp < Gt => дисперсії рівномірні ")
24. return True
25. else:
26. print("Gp > Gt => дисперсії нерівномірні ")
27. return False
28. def x\_i(i, raw\_factors\_table):
29. try:
30. assert i <= 10
31. except:
32. raise AssertionError("i must be smaller or equal 10")
33. with\_null\_factor = list(map(lambda x: [1] + x, generate\_factors\_table(raw\_factors\_table)))
34. res = [row[i] for row in with\_null\_factor]
35. return np.array(res)
36. def m\_ij(\*arrays):
37. return np.average(reduce(lambda accum, el: accum\*el, arrays))
38. def calculate\_theoretical\_y(x\_table, b\_coefficients, importance):
39. x\_table = [list(compress(row, importance)) for row in x\_table]
40. b\_coefficients = list(compress(b\_coefficients, importance))
41. y\_vals = np.array([sum(map(lambda x, b: x\*b, row, b\_coefficients)) for row in x\_table])
42. return y\_vals
43. def get\_cochran\_value(f1, f2, q):
44. partResult1 = q / f2 # (f2 - 1)
45. params = [partResult1, f1, (f2 - 1) \* f1]
46. fisher = f.isf(\*params)
47. result = fisher/(fisher + (f2 - 1))
48. return Decimal(result).quantize(Decimal('.0001'))
49. def get\_student\_value(f3, q):
50. return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal('.0001'))
51. def get\_fisher\_value(f3,f4, q):
52. return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal('.0001'))
53. #згідно варіанту
54. x1min, x1max = -9, 1
55. x2min, x2max = -2, 3
56. x3min, x3max = -2, 4
57. x\_avr\_min = (x1min + x2min + x3min) / 3
58. x\_avr\_max = (x1max + x2max + x3max) / 3
59. m = 3
60. N = 15
61. ymin = 200 + x\_avr\_min
62. ymax = 200 + x\_avr\_max
63. p = 0.95
64. x0 = [(x1max+x1min)/2, (x2max+x2min)/2, (x3max+x3min)/2]
65. detx = [abs(x1min - x0[0]), abs(x2min-x0[1]), abs(x3min-x0[2])]
66. l=1.215
67. raw\_natur\_table = [[x1min, x2min, x3min],
68. [x1min, x2max, x3max],
69. [x1max, x2min, x3max],
70. [x1max, x2max, x3min],
71. [x1min, x2min, x3max],
72. [x1min, x2max, x3min],
73. [x1max, x2min, x3min],
74. [x1max, x2max, x3max],
75. [-l\*detx[0]+x0[0], x0[1], x0[2]],
76. [ l\*detx[0]+x0[0], x0[1], x0[2]],
77. [x0[0], -l\*detx[1]+x0[1], x0[2]],
78. [x0[0], l\*detx[1]+x0[1], x0[2]],
79. [x0[0], x0[1], -l\*detx[2]+x0[2]],
80. [x0[0], x0[1], l\*detx[2]+x0[2]],
81. [x0[0], x0[1], x0[2]]]
82. raw\_factors\_table = [[-1, -1, -1],
83. [-1, +1, +1],
84. [+1, -1, +1],
85. [+1, +1, -1],
86. [-1, -1, +1],
87. [-1, +1, -1],
88. [+1, -1, -1],
89. [+1, +1, +1],
90. [-1.215, 0, 0],
91. [+1.215, 0, 0],
92. [0, -1.215, 0],
93. [0, +1.215, 0],
94. [0, 0, -1.215],
95. [0, 0, +1.215],
96. [0, 0, 0]]
97. factors\_table = generate\_factors\_table(raw\_factors\_table)
98. print("Матриця кодованих значень Х")
99. for row in factors\_table:
100. print(row)
101. natur\_table = generate\_factors\_table(raw\_natur\_table)
102. with\_null\_factor = list(map(lambda x: [1] + x, natur\_table))
103. y\_arr = [[random.random()\*(ymax-ymin) + ymin for i in range(m)] for j in range(N)]
104. while not cochran\_criteria(m, N, y\_arr):
105. m+=1
106. y\_arr = [[random.random()\*(ymax - ymin) + ymin for i in range(m)] for j in range(N)]
107. y\_i = np.array([np.average(row) for row in y\_arr])
108. 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)]
109. free\_values = [m\_ij(y\_i, x\_i(i, raw\_factors\_table)) for i in range(11)]
110. beta\_coef = np.linalg.solve(coefficients, free\_values)
111. # -----------------------------Критерій Стьюдента--------------------------------------
112. y\_table = y\_arr
113. print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: ")
114. average\_variation = np.average(list(map(np.var, y\_table)))
115. y\_averages = np.array(list(map(np.average, y\_table)))
116. variation\_beta\_s = average\_variation/N/m
117. standard\_deviation\_beta\_s = math.sqrt(variation\_beta\_s)
118. x\_vals = [x\_i(i, raw\_factors\_table) for i in range(11)]
119. t\_i = np.array([abs(beta\_coef[i])/standard\_deviation\_beta\_s for i in range(len(beta\_coef))])
120. f3 = (m-1)\*N
121. q = 1-p
122. t = get\_student\_value(f3, q)
123. importance = [True if el > t else False for el in list(t\_i)]
124. print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x: str(round(float(x), 3)), beta\_coef))),\
125. "\nКоефіцієнти ts: " + ", ".join(list(map(lambda i: f"{i:.3f}", t\_i))), \
126. f"\nf3 = {f3}; q = {q:.3f}; tтабл = {t}")
127. beta\_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11", "β22", "β33"]
128. x\_i\_names = list(compress(["1", "x1", "x2", "x3", "x12", "x13", "x23", "x123", "x1^2", "x2^2", "x3^2"], importance))
129. betas\_to\_print = list(compress(beta\_coef, importance))
130. for i in range(len(importance)):
131. if not importance[i]:
132. print(f"{beta\_i[i]} = 0 - незначимий")
133. print("Рівняння регресії без незначимих членів: y = ", end="")
134. for i in range(len(betas\_to\_print)):
135. print(f" {betas\_to\_print[i]:+.3f}\*{x\_i\_names[i]}", end="")
136. #--------------------------------критерій Фішера-------------------------------------
137. d = len(list(filter(None, importance)))
138. y\_table = y\_arr
139. f3 = (m - 1) \* N
140. f4 = N - d
141. q = 1-p
142. theor\_y = calculate\_theoretical\_y(natur\_table, beta\_coef, importance)
143. y\_averages = np.array(list(map(np.average, y\_table)))
144. s\_ad = m/(N-d)\*(sum((theor\_y-y\_averages)\*\*2))
145. y\_variations = np.array(list(map(np.var, y\_table)))
146. s\_v = np.average(y\_variations)
147. f\_p = float(s\_ad/s\_v)
148. f\_t = get\_fisher\_value(f3, f4, q)
149. print("\n\nПеревірка адекватності моделі за критерієм Фішера:",\
150. "\nТеоретичні значення y для різних комбінацій факторів:")
151. for i in range(len(natur\_table)):
152. print(f"x1 = {natur\_table[i][1]:>6.3f}; x2 = {natur\_table[i][2]:>6.3f}; "
153. f"x3 = {natur\_table[i][3]:>7.3f}; y = {theor\_y[i]:>8.3f}")
154. print(f"\nFp = {f\_p:.3f}, Ft = {f\_t:.3f}","\nFp < Ft => модель адекватна" if f\_p < f\_t else \
155. "Fp > Ft => модель неадекватна")

**Результат виконання роботи програми:**

****

**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.