Міністерство освіти і науки України Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського» Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 4**

з дисципліни «Методи оптимізації та планування»

Тема: Проведення трьохфакторного експерименту

при використанні рівняння регресії з урахуванням ефекту взаємодії.

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-83 Коломієць Д.Ю. Залікова – 8313 Варіант-312

ПЕРЕВІРИВ:

Регіда П.Г.

Київ – 2020

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання на лабораторну роботу

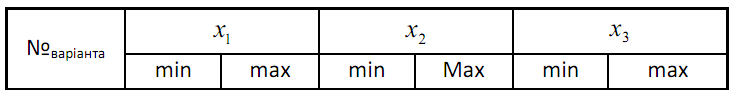
1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках

факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

1. Знайти коефіцієнти рівняння регресії і записати його.
2. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
3. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
4. Написати комп'ютерну програму, яка усе це моделює

Варіант:

312

****



**Код програми:**

**import** numpy **as** np  
**from** itertools **import** product, combinations  
  
  
np.set\_printoptions(formatter={**'float\_kind'**: **lambda** x: **"%.2f"** % x})  
  
*# Критерій Кохрена*gt = {1: 0.6798, 2: 0.5157, 3: 0.4377, 4: 0.3910, 5: 0.3595, 6: 0.3362, 7:  
 0.3185, 8: 0.3043, 9: 0.2926, 10: 0.2829}  
  
*# Критерій Стьюдента*tt = {16: 2.120, 24: 2.064, 32: 1.96} *# m = [3, 6]  
  
# Критерій Фішера*ft = {1: {16: 4.5, 24: 4.3, 32: 4.2},  
 2: {16: 3.6, 24: 3.4, 32: 3.3},  
 3: {16: 3.2, 24: 3.0, 32: 2.9},  
 4: {16: 3.0, 24: 2.8, 32: 2.7},  
 5: {16: 2.9, 24: 2.6, 32: 2.5},  
 6: {16: 2.7, 24: 2.5, 32: 2.4}}  
  
  
**def** make\_norm\_plan\_matrix(plan\_matrix, matrix\_of\_min\_and\_max\_x):  
 X0 = np.mean(matrix\_with\_min\_max\_x, axis=1)  
 interval\_of\_change = np.array([(matrix\_of\_min\_and\_max\_x[i, 1] - X0[i])  
 **for** i **in** range(len(plan\_matrix[0]))])  
 X\_norm = np.array(  
 [[round((plan\_matrix[i, j] - X0[j]) / interval\_of\_change[j], 3) **for** j  
 **in** range(len(plan\_matrix[i]))]  
 **for** i **in** range(len(plan\_matrix))])  
 **return** X\_norm  
  
  
**def** cochran\_check(Y\_matrix):  
 mean\_Y = np.mean(Y\_matrix, axis=1)  
 dispersion\_Y = np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0)  
 Gp = np.max(dispersion\_Y) / (np.sum(dispersion\_Y))  
 **return** Gp < gt[m - 1]  
  
**def** students\_t\_test(norm\_matrix, Y\_matrix):  
 mean\_Y = np.mean(Y\_matrix, axis=1)  
 dispersion\_Y = np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0)  
 mean\_dispersion = np.mean(dispersion\_Y)  
 sigma = np.sqrt(mean\_dispersion / (N \* m))  
 betta = np.mean(norm\_matrix.T \* mean\_Y, axis=1)  
 t = np.abs(betta) / sigma  
 **if** (m - 1) \* N > 32:  
 **return** np.where(t > 1.96)  
 **return** np.where(t > tt[(m - 1) \* N])  
  
  
**def** phisher\_criterion(Y\_matrix, d):  
 **if** d == N:  
 **return False** Sad = m / (N - d) \* np.mean(check1 - mean\_Y)  
 mean\_dispersion = np.mean(np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0))  
 Fp = Sad / mean\_dispersion  
 **if** (m - 1) \* N > 32:  
 **if** N - d > 6:  
 **return** Fp < ft[6][32]  
 **return** Fp < ft[N - d][32]  
 **if** N - d > 6:  
 **return** Fp < ft[6][(m - 1) \* N]  
 **return** Fp < ft[N - d][(m - 1) \* N]  
  
**"""  
x1min = 10  
x1max = 60  
  
x2min = -30  
x2max = 45  
  
x3min = -30  
x3max = 45  
  
"""**matrix\_with\_min\_max\_x = np.array([[10, 60], [-30, 45], [-30, 45]])  
m = 6  
N = 8  
norm\_matrix = np.array(list(product(**"01"**, repeat=3)), dtype=np.int)  
norm\_matrix[norm\_matrix == 0] = -1  
norm\_matrix = np.insert(norm\_matrix, 0, 1, axis=1)  
plan\_matrix = np.empty((8, 3))  
  
**for** i **in** range(len(norm\_matrix)):  
 **for** j **in** range(1, len(norm\_matrix[i])):  
 **if** j == 1:  
 **if** norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j - 1] = 10  
 **elif** norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j - 1] = 60  
 **elif** j == 2:  
 **if** norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j - 1] = -30  
 **elif** norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j - 1] = 45  
 **elif** j == 3:  
 **if** norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j - 1] = -30  
 **elif** norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j - 1] = 45  
  
plan\_matr = np.insert(plan\_matrix, 0, 1, axis=1)  
Y\_matrix = np.random.randint(200 + np.mean(matrix\_with\_min\_max\_x, axis=0)[0],  
 200 + np.mean(matrix\_with\_min\_max\_x, axis=0)[1], size=(N, m))  
mean\_Y = np.mean(Y\_matrix, axis=1)  
combination = list(combinations(range(1, 4), 2))  
**for** i **in** combination:  
 plan\_matr = np.append(plan\_matr, np.reshape(plan\_matr[:, i[0]] \*  
 plan\_matr[:, i[1]], (8, 1)), axis=1)  
 norm\_matrix = np.append(norm\_matrix, np.reshape(norm\_matrix[:, i[0]] \*  
 norm\_matrix[:, i[1]], (8, 1)), axis=1)  
plan\_matr = np.append(plan\_matr, np.reshape(plan\_matr[:, 1] \* plan\_matr[:, 2]  
 \* plan\_matr[:, 3], (8, 1)), axis=1)  
norm\_matrix = np.append(norm\_matrix, np.reshape(norm\_matrix[:, 1] \* norm\_matrix[:, 2] \* norm\_matrix[:, 3], (8, 1)), axis=1)  
  
**if** cochran\_check(Y\_matrix):  
 b\_natura = np.linalg.lstsq(plan\_matr, mean\_Y, rcond=**None**)[0]  
 b\_norm = np.linalg.lstsq(norm\_matrix, mean\_Y, rcond=**None**)[0]  
 check1 = np.sum(b\_natura \* plan\_matr, axis=1)  
 check2 = np.sum(b\_norm \* norm\_matrix, axis=1)  
 indexes = students\_t\_test(norm\_matrix, Y\_matrix)  
  
 print(**"\nМатриця плану експерименту: \n"**, plan\_matr)  
 print(**"\nНормована матриця: \n"**, norm\_matrix)  
 print(**"\nМатриця відгуків: \n"**, Y\_matrix)  
 print(**"\nСередні значення У: "**, mean\_Y)  
 print(**"Натуралізовані коефіціенти: "**, b\_natura)  
 print(**"Нормовані коефіціенти: "**, b\_norm)  
 print(**"Перевірка 1: "**, check1)  
 print(**"Перевірка 2: "**, check2)  
 print(**"Індекси коефіціентів, які задовольняють критерію Стьюдента: "**,  
 np.array(indexes)[0])  
 print(**"Критерій Стьюдента: "**, np.sum(b\_natura[indexes] \*  
 np.reshape(plan\_matr[:,  
  
 indexes], (N, np.size(indexes))), axis=1))  
 **if** phisher\_criterion(Y\_matrix, np.size(indexes)):  
 print(**"\nРівняння регресії адекватно оригіналу."**)  
 **else**:  
 print(**"\nРівняння регресії неадекватно оригіналу."**)  
**else**:  
 print(**"\n\nДисперсія неоднорідна!"**)

# Результат роботи програми:

"D:\Kolomiets Dmitro Python\venv\Scripts\python.exe" C:/Users/Samsung/.PyCharmCE2018.2/config/scratches/MOPE/Lab4.py

Матриця плану експерименту:

[[1.00 10.00 -30.00 -30.00 -300.00 -300.00 900.00 9000.00]

[1.00 10.00 -30.00 45.00 -300.00 450.00 -1350.00 -13500.00]

[1.00 10.00 45.00 -30.00 450.00 -300.00 -1350.00 -13500.00]

[1.00 10.00 45.00 45.00 450.00 450.00 2025.00 20250.00]

[1.00 60.00 -30.00 -30.00 -1800.00 -1800.00 900.00 54000.00]

[1.00 60.00 -30.00 45.00 -1800.00 2700.00 -1350.00 -81000.00]

[1.00 60.00 45.00 -30.00 2700.00 -1800.00 -1350.00 -81000.00]

[1.00 60.00 45.00 45.00 2700.00 2700.00 2025.00 121500.00]]

Нормована матриця:

[[ 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]]

Матриця відгуків:

[[214 246 240 219 221 225]

[233 240 232 231 203 227]

[184 244 219 244 204 242]

[233 210 233 201 228 200]

[206 224 208 237 220 190]

[212 207 210 233 198 206]

[204 187 206 203 209 232]

[228 239 193 206 221 217]]

Середні значення У: [227.50 227.67 222.83 217.50 214.17 211.00 206.83 217.33]

Натуралізовані коефіціенти: [227.35 -0.25 -0.10 -0.04 0.00 0.00 -0.00 0.00]

Нормовані коефіціенти: [218.10 -5.77 -1.98 0.27 1.73 1.56 1.02 2.40]

Перевірка 1: [227.50 227.67 222.83 217.50 214.17 211.00 206.83 217.33]

Перевірка 2: [227.50 227.67 222.83 217.50 214.17 211.00 206.83 217.33]

Індекси коефіціентів, які задовольняють критерію Стьюдента: [0 1]

Критерій Стьюдента: [224.82 224.82 224.82 224.82 212.15 212.15 212.15 212.15]

Рівняння регресії адекватно оригіналу.

Process finished with exit code 0

**Висновок:**

Я провів повний трьохфакторний експеримент. Знайшов рівняння регресії адекватне об'єкту за допомогою мови програмування Python.