МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №3
з дисципліни «Методи оптимізації та планування»
на тему: «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

Виконав:
студент 2-го курсу ФІОТ
групи ІВ-71
Мазан Я. В.

Перевірив:
асистент
Регіда П. Г.

Київ – 2019

Варіант:

№ у списку – 9

№ варіанту	x_1		x_2		x_3	
109	-20	15	10	60	15	35

$x_{cp\ max} = 110/3 \approx 37$

$x_{cp\ min} = 5/3 \approx 2$

$y_{max} = 237$

$y_{min} = 202$

Код програми:

```
import itertools
import math
import random
import numpy as np
import xlrd
y_min = 202
y_max = 237
factors_table = [[1, -1, -1, -1],
                 [1, -1, +1, +1],
                 [1, +1, -1, +1],
                 [1, +1, +1, -1]]
naturalized_factors_table = [np.array([-20, 10, 15]),
                             np.array([-20, 60, 35]),
                             np.array([+15, 10, 35]),
                             np.array([+15, 60, 15])]
def experiment(m):
    return [np.array([random.randint(202+i*4, 237) for _ in range(m)]) for i in range(4)]
def coefficients(m, natural_x_table, normalized_x_table, y_table):
    def m_i(array):
        return np.average(array)
    def a_ij(arr1, arr2):
        return np.average(arr1*arr2)
    def a_jj(array):
        return np.average(array**2)
    y_averages = [np.average(i) for i in y_table]
    x1 = np.array([i[0] for i in natural_x_table])
    x2 = np.array([i[1] for i in natural_x_table])
    x3 = np.array([i[2] for i in natural_x_table])
    mx1 = m_i(x1)
    mx2 = m_i(x2)
    mx3 = m_i(x3)
    my = m_i(y_averages)
    a11 = a_jj(x1)
    a22 = a_jj(x2)
    a33 = a_jj(x3)
    a12 = a_ij(x1, x2)
    a13 = a_ij(x1, x3)
    a23 = a_ij(x2, x3)
    a1 = a_ij(y_averages, x1)
    a2 = a_ij(y_averages, x2)
    a3 = a_ij(y_averages, x3)
    equations_sys_coefficients = [[1,   mx1, mx2, mx3],
                                  [mx1, a11, a12, a13],
                                  [mx2, a12, a22, a23],
                                  [mx3, a13, a23, a33]]
    equations_sys_free_members = [my, a1, a2, a3]
    b_coefficients = np.linalg.solve(equations_sys_coefficients,equations_sys_free_members)
    # normalized regression coefficients
    b_normalized_coefficients = np.array([np.average(y_averages),
                                  np.average(y_averages*np.array([i[1] for i in
normalized_x_table])),
                                  np.average(y_averages*np.array([i[2] for i in
normalized_x_table])),
                                  np.average(y_averages*np.array([i[3] for i in
normalized_x_table]))])
    return b_coefficients, b_normalized_coefficients
def cochran_criteria(m, N, y_table):
    # y_name = globals()
    # print(y_name["y_table"])
```

```
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {} для таблиці
y_table".format(m, N))
    cochran_table = xlrd.open_workbook("Cochran.xls").sheet_by_index(0)
    y_averages = [np.average(i) for i in y_table]
    y_variations = [np.var(i) for i in y_table]
    max_y_variation = max(y_variations)
    gp = max_y_variation/sum(y_variations)
    f1 = m - 1              # column
    f2 = N                  # row
    p = 0.95
    q = 1-p
    column = f1 if f1 <= 6 else 7 if f1 in [7,8] \
        else 8 if f1 in [9,10] \
        else 9 if f1 in range(10,17) \
        else 10 if f1 in range(17,37) \
        else 11
    row = f2-1               # 3rd row with value 4
    gt = cochran_table.row_values(row-1)[column-1]/math.pow(10,4)
    print("Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}".format(gp, gt, f1, f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні - все правильно")
        return True
    else:
        print("Gp > Gt => дисперсії нерівномірні - треба ще експериментів")
        return False
def student_criteria(m, N, y_table, normalized_x_table):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} для
таблиці y_table".format(m, N))
    student_table = xlrd.open_workbook("Student.xls").sheet_by_index(0)
    average_variation = np.average(list(map(np.var, y_table)))
    y_averages = np.array(list(map(np.average, y_table)))
    variation_beta_s = average_variation/N/m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    x_i = np.array([[el[i] for el in normalized_x_table] for i in range(len(normalized_x_table))])
    coefficients_beta_s = np.array([np.average(y_averages*x_i[i]) for i in range(len(x_i))])
    print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(str,coefficients_beta_s))))
    t_i = np.array([abs(coefficients_beta_s[i])/standard_deviation_beta_s for i in
range(len(coefficients_beta_s))])
    print("Коефіцієнти ts:         " + ", ".join(list(map(lambda i: "{:.2f}".format(i), t_i))))
    f3 = (m-1)*N
    p = 0.95
    q = 0.05
    t = float(student_table.col_values(3)[f3].replace(",", "."))
    importance = [True if el > t else False for el in list(t_i)]
    # print result data
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t))
    beta_i = ["β{}".format(i) for i in range(N)]
    importance_to_print = ["важливий" if i else "неважливий" for i in importance]
    to_print = list(zip(beta_i, importance_to_print))
    x_i_names = [""] + list(itertools.compress(["x{}".format(i) for i in range(N)], importance))[1:]
    betas_to_print = list(itertools.compress(coefficients_beta_s, importance))
    print("{0[0]} {0[1]}; {1[0]} {1[1]}; {2[0]} {2[1]}; {3[0]} {3[1]}".format(*to_print))
    # print(list(zip(list(map(lambda x: "{:+.2f}".format(x), betas_to_print)), x_i_names)))
    equation = " ".join(["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x),
betas_to_print)),x_i_names)])
    print("Рівняння регресії без незначимих членів: y = " + equation)
    return importance
def calculate_theoretical_y(x_table, b_coefficients, importance):
    x_table = [list(itertools.compress(row, importance)) for row in x_table]
    b_coefficients = list(itertools.compress(b_coefficients, importance))
    y_vals = np.array([sum(map(lambda x, b: x*b, row, b_coefficients)) for row in x_table])
    return y_vals
def fisher_criteria(m, N, d, naturalized_x_table, y_table, b_coefficients, importance):
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N = {} для таблиці
y_table".format(m, N))
    fisher_table = xlrd.open_workbook("Fisher.xls").sheet_by_index(0)
    f3 = (m - 1) * N
    f4 = N - d
    theoretical_y = calculate_theoretical_y(naturalized_x_table, b_coefficients, importance)
    theoretical_values_to_print = list(zip(map(lambda x: "x1 = {0[1]}, x2 = {0[2]}, x3 =
{0[3]}".format(x),naturalized_x_table),theoretical_y))
    # print(theoretical_values_to_print)
    print("Теоретичні значення y для різних комбінацій факторів:")
    print("\n".join(["{arr[0]}: y = {arr[1]}".format(arr = el) for el in
theoretical_values_to_print]))
    y_averages = np.array(list(map(np.average, y_table)))
    s_ad = m/(N-d)*(sum((theoretical_y-y_averages)**2))
    y_variations = np.array(list(map(np.var, y_table)))
```

```
    s_v = np.average(y_variations)
    f_p = float(s_ad/s_v)
    f_t = float((fisher_table.row_values(f3) if f3 <= 30 else fisher_table.row_values(30))
[f4].replace(",","."))
    print("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель неадекватна")
    return True if f_p < f_t else False
m = 4
y_table = experiment(m)
while not cochran_criteria(m, 4, y_table):
    m += 1
    y_table = experiment(m)
print("Матриця планування:")
labels_table = ["x1", "x2", "x3"] + ["y{}".format(i+1) for i in range(m)]
rows_table = [list(naturalized_factors_table[i]) + list(y_table[i]) for i in range(m)]
rows_normalized_table = [factors_table[i] + list(y_table[i]) for i in range(m)]
print((" "*4).join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j), rows_table[i])) for i in
range(len(rows_table))]))
print("\t")
print("Нормована матриця планування:")
labels_table = ["x{}".format(i) for i in range(4)] + ["y{}".format(i+1) for i in range(m)]
print((" "*4).join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j), rows_normalized_table[i])) for i in
range(len(rows_normalized_table))]))
print("\t")
naturalized_coefficients, normalized_coefficients = coefficients(m, naturalized_factors_table,
factors_table, y_table)
print("Рівняння регресії для нормованих факторів: y = {0} {1:+}*x1 {2:+}*x2
{3:+}*x3".format(*normalized_coefficients))
print("\nРівняння регресії для натуралізованих факторів: y = {0:.3f} {1:+.3f}*x1 {2:+.3f}*x2
{3:+.3f}*x3".format(*naturalized_coefficients))
importance = student_criteria(m, 4, y_table, factors_table)
factors_table = [np.array([1]+list(i)) for i in naturalized_factors_table]
correctness = fisher_criteria(m, 4, 1, factors_table, y_table, naturalized_coefficients, importance)
```

Результати виконання:

```
/home/yan/PycharmProjects/Optimization&PlanningLab3/venv/bin/python /home/yan/PycharmProjects/Optimiz
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 4, N = 4 для таблиці y_table
Gp = 0.45774647887323944; Gt = 0.6841; f1 = 3; f2 = 4; q = 0.05
Gp < Gt => дисперсії рівномірні - все правильно
Матриця планування:
x1    x2    x3    y1    y2    y3    y4
-20   +10   +15   +219  +237  +210  +213
-20   +60   +35   +215  +211  +225  +233
+15   +10   +35   +225  +236  +226  +224
+15   +60   +15   +215  +225  +231  +223

Нормована матриця планування:
x0    x1    x2    x3    y1    y2    y3    y4
+1    -1    -1    -1    +219  +237  +210  +213
+1    -1    +1    +1    +215  +211  +225  +233
+1    +1    -1    +1    +225  +236  +226  +224
+1    +1    +1    -1    +215  +225  +231  +223

Рівняння регресії для нормованих факторів: y = 223.0 +2.625*x1 -0.75*x2 +1.375*x3

Рівняння регресії для натуралізованих факторів: y = 220.988 +0.150*x1 -0.030*x2 +0.137*x3

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 4, N = 4 для таблиці y_table
Оцінки коефіцієнтів βs: 223.0, 2.625, -0.75, 1.375
Коефіцієнти ts:         115.25, 1.36, 0.39, 0.71
f3 = 12; q = 0.05; tтабл = 2.1788
β0 важливий; β1 неважливий; β2 неважливий; β3 неважливий
Рівняння регресії без незначимих членів: y = +223.00

Перевірка адекватності моделі за критерієм Фішера: m = 4, N = 4 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -20, x2 = 10, x3 = 15: y = 220.98750000000027
x1 = -20, x2 = 60, x3 = 35: y = 220.98750000000027
x1 = 15, x2 = 10, x3 = 35: y = 220.98750000000027
x1 = 15, x2 = 60, x3 = 15: y = 220.98750000000027
Fp = 1.1924326204137452, Ft = 3.49
Fp < Ft => модель адекватна

Process finished with exit code 0
```