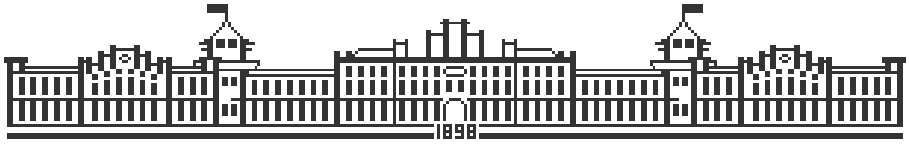

Міністерство освіти та науки України
Національний технічний університет України “Київський політехнічний інститут”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4
з дисципліни «Методи оптимізації та планування»
на тему: «Проведення трьохфакторного експерименту
при використанні рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:
студент групи ІВ-71
Мазан Я. В.
Залікова №7109
Перевірив:
асистент
Регіда П. Г.

Київ – 2019

Варіант:

x_{1min} = -30; x_{1max} = 0; x_{cpmax} = 10; x_{cpmin} = -22;

x_{2min} = -35; x_{2max} = 10; y_{jmax} = 210; y_{jmin} = 178;

x_{3min} = 0; x_{3max} = 20;

Код програми:

```
import numpy as np
import random as r
from functions import *
#                   1   2   3    12  13  23    123
norm_factors_table = [[-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]]
zero_factor = [+1]*8
factors_table = [[-30, -35, 0,       1050, 0, 0,          0],
                 [-30,  10, 20,      -300, -600,       -6000],
                 [0,   -35, 20,       0, 0, -700,          0],
                 [0,    10, 0,        0, 0, 0,             0],
                 [-30, -35, 20,       1050, -600, -700, 21000],
                 [-30,  10, 0,        -300, 0, 0,          0],
                 [0,   -35, 0,        0, 0, 0,             0],
                 [0,    10, 20,       0, 0, 200,           0]]
y_min = 178
y_max = 210
M = 5
N = 8
y_arr = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]
x1 = np.array(list(zip(*factors_table))[0])
x2 = np.array(list(zip(*factors_table))[1])
x3 = np.array(list(zip(*factors_table))[2])
yi = np.array([np.average(i) for i in y_arr])
coeffs = [[N,       m_ij(x1),    m_ij(x2),    m_ij(x3),    m_ij(x1*x2),    m_ij(x1*x3),    m_ij(x2*x3),    m_ij(x1*x2*x3)],
        [m_ij(x1), m_ij(x1**2), m_ij(x1*x2), m_ij(x1*x3), m_ij(x1**2*x2), m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1**2*x2*x3)],
        [m_ij(x2), m_ij(x1*x2), m_ij(x2**2), m_ij(x2*x3), m_ij(x1*x2**2), m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x1*x2**2*x3)],
        [m_ij(x3), m_ij(x1*x3), m_ij(x2*x3), m_ij(x3**2), m_ij(x1*x2*x3), m_ij(x1*x3**2), m_ij(x2*x3**2), m_ij(x1*x2*x3**2)],
        [m_ij(x1*x2), m_ij(x1**2*x2), m_ij(x1*x2**2), m_ij(x1*x2*x3), m_ij(x1**2*x2**2), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3),
m_ij(x1**2*x2**2*x3)],
        [m_ij(x1*x3), m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1*x3**2), m_ij(x1**2*x2*x3), m_ij(x1**2*x3**2), m_ij(x1*x2*x3**2),
m_ij(x1**2*x2*x3**2)],
        [m_ij(x2*x3), m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x2*x3**2), m_ij(x1*x2**2*x3), m_ij(x1*x2*x3**2), m_ij(x2**2*x3**2),
m_ij(x1*x2**2*x3**2)],
        [m_ij(x1*x2*x3), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3), m_ij(x1*x2*x3**2), m_ij(x1**2*x2**2*x3), m_ij(x1**2*x2*x3**2),
m_ij(x1*x2**2*x3**2), m_ij(x1**2*x2**2*x3**2)]]
free_vals = [m_ij(yi), m_ij(yi*x1), m_ij(yi*x2), m_ij(yi*x3), m_ij(yi*x1*x2), m_ij(yi*x1*x3), m_ij(yi*x2*x3), m_ij(yi*x1*x2*x3)]
natural_bi = np.linalg.solve(coeffs, free_vals)
natural_x1 =  np.array(list(zip(*norm_factors_table))[0])
natural_x2 =  np.array(list(zip(*norm_factors_table))[1])
natural_x3 =  np.array(list(zip(*norm_factors_table))[2])
norm_bi = [m_ij(yi),
         m_ij(yi*natural_x1),
         m_ij(yi*natural_x2),
         m_ij(yi*natural_x3),
         m_ij(yi*natural_x1*natural_x2),
         m_ij(yi*natural_x1*natural_x3),
         m_ij(yi*natural_x2*natural_x3),
         m_ij(yi*natural_x1*natural_x2*natural_x3)]
while not cochran_criteria(M, 4, y_arr):
    M += 1
    y_table = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]
print("Матриця планування:")
labels_table = list(map(lambda x: x.ljust(6), ["x1", "x2", "x3", "x12", "x13", "x23", "x123"] + ["y{}".format(i+1) for i in range(M)]))
rows_table = [list(factors_table[i]) + list(y_arr[i]) for i in range(M)]
rows_normalized_table = [factors_table[i] + list(y_arr[i]) for i in range(M)]
print((" ").join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+6}".format(j), rows_table[i])) for i in range(len(rows_table))]))
```

```
print("\t")
norm_factors_table_zero_factor = [[+1]+i for i in norm_factors_table]
importance = student_criteria(M, N, y_arr , norm_factors_table_zero_factor)
fisher_criteria(M, N, 1, factors_table, y_arr, natural_bi, importance)


import math

from _pydecimal import Decimal
from scipy.stats import f, t, ttest_ind, norm
from functools import reduce
from itertools import compress
import numpy as np
def cochran_criteria(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {} для таблиці y_table".format(m, N))
    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
    f2 = N
    p = 0.95
    q = 1-p
    gt = get_cochran_value(f1,f2, q)
    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: "with zero factor!"):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} "
          "для таблиці y_table та нормалізованих факторів".format(m, N))
    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
    q = 0.05
    t = get_student_value(f3, q)
    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 = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123"]
    importance_to_print = ["важливий" if i else "неважливий" for i in importance]
    to_print = map(lambda x: x[0] + " " + x[1], zip(beta_i, importance_to_print))
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23", "x123"], importance))#'[""] + list(compress(["x{}".format(i) for i in
range(N)], importance))[1:]
    betas_to_print = list(compress(coefficients_beta_s, importance))
    print(*to_print, sep = "; ")
    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(compress(row, importance)) for row in x_table]
    b_coefficients = list(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))
    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05
    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("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 = get_fisher_value(f3, f4, q)
    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
def m_ij(*arrays):
    return np.average(reduce(lambda accum, el: accum*el, arrays))
def get_cochran_value(f1, f2, q):
    partResult1 = q / f2 # (f2 - 1)
    params = [partResult1, f1, (f2 - 1) * f1]
    fisher = f.isf(*params)
    result = fisher/(fisher + (f2 - 1))
    return Decimal(result).quantize(Decimal('.0001')).__float__()
def get_student_value(f3, q):
    return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal('.0001')).__float__()
def get_fisher_value(f3,f4, q):
    return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal('.0001')).__float__()
```

Результати виконання:

```
/home/yan/PycharmProjects/Optimisation&PlanningLab4/venv/bin/python /home/yan/PycharmProjects/Optimisation&PlanningLab4/main.py
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 5, N = 4 для таблиці y_table
Gp = 0.320460704607046; Gt = 0.6287; f1 = 4; f2 = 4; q = 0.05
Gp < Gt => дисперсії рівномірні - все правильно
Матриця планування:
x1     x2     x3     x12    x13    x23    x123   y1     y2     y3     y4     y5
-30    -35    +0     +1050  +0     +0     +0     +187   +181   +179   +178   +190
-30    +10    +20    -300   -600   -6000  +206   +186   +207   +180   +210
+0     -35    +20    +0     +0     -700   +0     +191   +185   +207   +190   +207
+0     +10    +0     +0     +0     +0     +0     +185   +182   +194   +203   +198
-30    -35    +20    +1050  -600   -700   +21000 +202   +191   +209   +194   +184
-30    +10    +0     -300   +0     +0     +0     +201   +192   +188   +190   +191
+0     -35    +0     +0     +0     +0     +0     +205   +209   +208   +202   +193
+0     +10    +20    +0     +0     +200   +0     +180   +189   +186   +188   +195


Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 5, N = 8 для таблиці y_table та нормалізованих факторів
Оцінки коефіцієнтів βs: 193.575, 1.275, -1.025, 0.775, -3.825, -3.825, -0.625, 1.275
Коефіцієнти ts:         159.33, 1.05, 0.84, 0.64, 3.15, 3.15, 0.51, 1.05
f3 = 32; q = 0.05; tтабл = 2.0369
β0 важливий; β1 неважливий; β2 неважливий; β3 неважливий; β12 важливий; β13 важливий; β23 неважливий; β123 неважливий
Рівняння регресії без незначимих членів: y = +193.57 -3.83x12 -3.83x13

Перевірка адекватності моделі за критерієм Фішера: m = 5, N = 8 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -35, x2 = 0, x3 = 1050: y = -155.28763529026136
x1 = 10, x2 = 20, x3 = -300: y = -1846.6632994424417
x1 = -35, x2 = 20, x3 = 0: y = -206.73511047945283
x1 = 10, x2 = 0, x3 = 0: y = 0.0
x1 = -35, x2 = 20, x3 = 1050: y = -281.3831772408705
x1 = 10, x2 = 0, x3 = -300: y = -155.28763529026136
x1 = -35, x2 = 0, x3 = 0: y = 0.0
x1 = 10, x2 = 20, x3 = 0: y = 59.06717442270081
Fp = 59283.72620339062, Ft = 2.3127
Fp > Ft => модель неадекватна

Process finished with exit code 0
```