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

Лабораторна робота №6

з дисципліни

«Методи оптимізації та планування експерименту»

Тема: «Проведення трьофакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:

студент групи ІО-93
Комаровський Роман Сергійович
Номер у списку: 14

Перевірив:

ас. Регіда П. Г.

Київ 2021

Мета: провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи **рототабельний** композиційний план.

Завдання:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1; *+l; -l;* 0 для x_1, x_2, x_3
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

де f(x1, x2, x3) вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів
5. Зробити висновки по виконаній роботі.

Варіант завдання:

	X_1		X_2		X_3		Функція
314	-25	75	5	40	15	25	5,5+6,4*x1+0,6*x2+2,7*x3+1,9*x1*x1+0,4*x2*x2+0,7*x3*x3+1,8*x1*x2+0,2*x1*x3+6,0*x2*x3+4,8*x1*x2*x3

Лістинг програми:

```
import math
import random
from _decimal import Decimal
from itertools import compress
from scipy.stats import f, t
import numpy
from functools import reduce


def regression_equation(x1, x2, x3, coeffs, importance=[True] * 11):
    factors_array = [1, x1, x2, x3, x1 * x2, x1 * x3, x2 * x3, x1 * x2 * x3,
x1 ** 2, x2 ** 2, x3 ** 2]
    return sum([el[0] * el[1] for el in compress(zip(coeffs, factors_array),
importance)])


def func(x1, x2, x3):
    coeffs = [5.5, 6.4, 0.6, 2.7, 1.9, 0.4, 0.7, 1.8, 0.2, 6.0, 4.8]
    return regression_equation(x1, x2, x3, coeffs)

xmin = [-25, 5, 15]
xmax = [75, 40, 25]
x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]
norm_plan_raw = [[-1, -1, -1],
                 [-1, +1, +1],
                 [+1, -1, +1],
                 [+1, +1, -1],
                 [-1, -1, +1],
                 [-1, +1, -1],
```

```
                  [+1, -1, -1],
                  [+1, +1, +1],
                  [-1.73, 0, 0],
                  [+1.73, 0, 0],
                  [0, -1.73, 0],
                  [0, +1.73, 0],
                  [0, 0, -1.73],
                  [0, 0, +1.73]]

natur_plan_raw = [[xmin[0],           xmin[1],           xmin[2]],
                  [xmin[0],           xmin[1],           xmax[2]],
                  [xmin[0],           xmax[1],           xmin[2]],
                  [xmin[0],           xmax[1],           xmax[2]],
                  [xmax[0],           xmin[1],           xmin[2]],
                  [xmax[0],           xmin[1],           xmax[2]],
                  [xmax[0],           xmax[1],           xmin[2]],
                  [xmax[0],           xmax[1],           xmax[2]],
                  [-1.73*dx[0]+x0[0], x0[1],             x0[2]],
                  [1.73*dx[0]+x0[0],  x0[1],             x0[2]],
                  [x0[0],             -1.73*dx[1]+x0[1], x0[2]],
                  [x0[0],             1.73*dx[1]+x0[1],  x0[2]],
                  [x0[0],             x0[1],             -1.73*dx[2]+x0[2]],
                  [x0[0],             x0[1],             1.73*dx[2]+x0[2]],
                  [x0[0],             x0[1],             x0[2]]]


def generate_factors_table(raw_array):
    raw_list = [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2],
row[0] * row[1] * row[2]] + list(
        map(lambda x: x ** 2, row)) for row in raw_array]
    return list(map(lambda row: list(map(lambda el: round(el, 3), row)),
raw_list))


def generate_y(m, factors_table):
    return [[round(func(row[0], row[1], row[2]) + random.randint(-5, 5), 3)
for _ in range(m)] for row in factors_table]


def print_matrix(m, N, factors, y_vals, additional_text=":"):
    labels_table = list(map(lambda x: x.ljust(10),
                            ["x1", "x2", "x3", "x12", "x13", "x23", "x123",
"x1^2", "x2^2", "x3^2"] + [
                                "y{}".format(i + 1) for i in range(m)]))
    rows_table = [list(factors[i]) + list(y_vals[i]) for i in range(N)]
    print("\nМатриця планування" + additional_text)
    print(" ".join(labels_table))
    print("\n".join([" ".join(map(lambda j: "{:<+10}".format(j),
rows_table[i])) for i in range(len(rows_table))]))
    print("\t")


def print_equation(coeffs, importance=[True] * 11):
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23",
"x123", "x1^2", "x2^2", "x3^2"], importance))
    coefficients to print = list(compress(coeffs, importance))
    equation = " ".join(
        ["".join(i) for i in zip(list(map(lambda x: "{:+.2f}".format(x),
coefficients_to_print)), x_i_names)])
    print("Рівняння регресії: y = " + equation)


def set_factors_table(factors_table):
    def x_i(i):
```

```
        with_null_factor = list(map(lambda x: [1] + x,
generate_factors_table(factors_table)))
        res = [row[i] for row in with_null_factor]
        return numpy.array(res)

    return x_i


def m_ij(*arrays):
    return numpy.average(reduce(lambda accum, el: accum * el, list(map(lambda
el: numpy.array(el), arrays))))


def find_coefficients(factors, y_vals):
    x_i = set_factors_table(factors)
    coeffs = [[m_ij(x_i(column), x_i(row)) for column in range(11)] for row
in range(11)]
    y_numpy = list(map(lambda row: numpy.average(row), y_vals))
    free_values = [m_ij(y_numpy, x_i(i)) for i in range(11)]
    beta_coefficients = numpy.linalg.solve(coeffs, free_values)
    return list(beta_coefficients)


def cochran_criteria(m, N, y_table):
    def get_cochran_value(f1, f2, q):
        partResult1 = q / f2
        params = [partResult1, f1, (f2 - 1) * f1]
        fisher = f.isf(*params)
        result = fisher / (fisher + (f2 - 1))
        return Decimal(result).quantize(Decimal('.0001')).__float__()

    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N
= {}".format(m, N))
    y_variations = [numpy.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, beta_coefficients):
    def get_student_value(f3, q):
        return Decimal(abs(t.ppf(q / 2,
f3))).quantize(Decimal('.0001')).__float__()

    print("\nПеревірка значимості коефіцієнтів регресії за критерієм
Стьюдента: m = {}, N = {} ".format(m, N))
    average_variation = numpy.average(list(map(numpy.var, y_table)))
    variation_beta_s = average_variation / N / m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = [abs(beta_coefficients[i]) / standard_deviation_beta_s for i in
range(len(beta_coefficients))]
    f3 = (m - 1) * N
```

```
    q = 0.05
    t_our = get_student_value(f3, q)
    importance = [True if el > t_our else False for el in list(t_i)]

    print("Оцінки коефіцієнтів βs: " + ", ".join(list(map(lambda x:
str(round(float(x), 3)), beta_coefficients))))
    print("Коефіцієнти ts: " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), t_i))))
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t_our))
    beta_i = ["β0", "β1", "β2", "β3", "β12", "β13", "β23", "β123", "β11",
"β22", "β33"]
    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))
    print(*to_print, sep="; ")
    print_equation(beta_coefficients, importance)

    return importance


def fisher_criteria(m, N, d, x_table, y_table, b_coefficients, importance):
    def get_fisher_value(f3, f4, q):
        return Decimal(abs(f.isf(q, f4,
f3))).quantize(Decimal('.0001')).__float__()

    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05
    theoretical_y = numpy.array([regression_equation(row[0], row[1], row[2],
b_coefficients) for row in x_table])
    average_y = numpy.array(list(map(lambda el: numpy.average(el), y_table)))
    s_ad = m / (N - d) * sum((theoretical_y - average_y) ** 2)
    y_variations = numpy.array(list(map(numpy.var, y_table)))
    s_v = numpy.average(y_variations)
    f_p = float(s_ad / s_v)
    f_t = get_fisher_value(f3, f4, q)
    theoretical_values_to_print = list(
        zip(map(lambda x: "x1 = {0[1]:<10} x2 = {0[2]:<10} x3 =
{0[3]:<10}".format(x), x_table), theoretical_y))
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N =
{} для таблиці y_table".format(m, N))
    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]))
    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 = 3
N = 15
natural_plan = generate_factors_table(natur_plan_raw)
y_arr = generate_y(m, natur_plan_raw)
while not cochran criteria(m, N, y arr):
    m += 1
    y_arr = generate_y(m, natural_plan)

print_matrix(m, N, natural_plan, y_arr, " для натуралізованих факторів:")
coefficients = find_coefficients(natural_plan, y_arr)
print_equation(coefficients)
importance = student_criteria(m, N, y_arr, coefficients)
```

```
d = len(list(filter(None, importance)))
fisher_criteria(m, N, d, natural_plan, y_arr, coefficients, importance)
```

Результат

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.13738738738738737; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні - все правильно

Матриця планування для натуралізованих факторів:
x1      x2      x3      x12     x13     x23     x123    x1^2    x2^2    x3^2    y1      y2      y3
-25     +5      +15     -125    -375    +75     -1875   +625    +25     +225    -2468.0 -2462.0 -2467.0
-25     +5      +25     -125    -625    +125    -3125   +625    +25     +625    -2835.0 -2830.0 -2836.0
-25     +40     +15     -1000   -375    +600    -15000  +625    +1600   +225    -17915.0 -17917.0 -17919.0
-25     +40     +25     -1000   -625    +1000   -25000  +625    +1600   +625    -33792.0 -33791.0 -33789.0
+75     +5      +15     +375    +1125   +75     +5625   +5625   +25     +225    +14221.0 +14222.0 +14226.0
+75     +5      +25     +375    +1875   +125    +9375   +5625   +25     +625    +23259.0 +23256.0 +23252.0
+75     +40     +15     +3000   +1125   +600    +45000  +5625   +1600   +225    +99927.0 +99929.0 +99923.0
+75     +40     +25     +3000   +1875   +1000   +75000  +5625   +1600   +625    +156448.0 +156449.0 +156455.0
-61.5   +22.5   +20.0   -1383.75 -1230.0 +450.0  -27675.0 +3782.25 +506.25 +400.0  -47231.775 -47230.775 -47227.775
+111.5  +22.5   +20.0   +2508.75 +2230.0 +450.0  +50175.0 +12432.25 +506.25 +400.0  +104523.175 +104517.175 +104524.175
+25.0   -7.775  +20.0   -194.375 +500.0  -155.5  -3887.5  +625.0  +60.451 +400.0  -4649.124 -4657.124 -4655.124
+25.0   +52.775 +20.0   +1319.375 +500.0 +1055.5 +26387.5 +625.0  +2785.201 +400.0 +69946.531 +69951.531 +69951.531
+25.0   +22.5   +11.35  +562.5  +283.75 +255.375 +6384.375 +625.0  +506.25 +128.822 +16848.38 +16843.38 +16839.38
+25.0   +22.5   +28.65  +562.5  +716.25 +644.625 +16115.625 +625.0 +506.25 +820.822 +38173.415 +38171.415 +38175.415
+25.0   +22.5   +20.0   +562.5  +500.0  +450.0  +11250.0 +625.0  +506.25 +400.0  +27146.25 +27152.25 +27148.25

Рівняння регресії: y = +5.36 +6.37x1 +0.67x2 +2.64x3 +1.90x12 +0.40x13 +0.70x23 +1.80x123 +0.20x1^2 +6.00x2^2 +4.80x3^2


Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 5.362, 6.373, 0.667, 2.643, 1.902, 0.401, 0.696, 1.8, 0.2, 5.999, 4.803
Коефіцієнти ts: 14.03, 16.67, 1.74, 6.91, 4.98, 1.05, 1.82, 4.71, 0.52, 15.69, 12.56
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 важливий; β2 неважливий; β3 важливий; β12 важливий; β13 неважливий; β23 неважливий; β123 важливий; β11 неважливий; β22 важливий; β33 важливий
Рівняння регресії: y = +5.36 +6.37x1 +2.64x3 +1.90x12 +1.80x123 +6.00x2^2 +4.80x3^2


Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = 5          x2 = 15         x3 = -125       : y = -2466.2028973563
x1 = 5          x2 = 25         x3 = -125       : y = -2834.000034450879
x1 = 40         x2 = 15         x3 = -1000      : y = -17917.108239440655
x1 = 40         x2 = 25         x3 = -1000      : y = -33790.57204320306
x1 = 5          x2 = 15         x3 = 375        : y = 14223.332946143226
x1 = 5          x2 = 25         x3 = 375        : y = 23256.20247571332
x1 = 40         x2 = 15         x3 = 3000       : y = 99927.09427072477
x1 = 40         x2 = 25         x3 = 3000       : y = 156451.63046696168
x1 = 22.5       x2 = 20.0       x3 = -1383.75   : y = -47229.38922724063
x1 = 22.5       x2 = 20.0       x3 = 2508.75    : y = 104520.21778201175
x1 = -7.775     x2 = 20.0       x3 = -194.375   : y = -4653.583851338202
x1 = 52.775     x2 = 20.0       x3 = 1319.375   : y = 69949.08157352144
x1 = 22.5       x2 = 11.35      x3 = 562.5      : y = 16843.664533813517
x1 = 22.5       x2 = 28.65      x3 = 562.5      : y = 38172.8971533044
x1 = 22.5       x2 = 20.0       x3 = 562.5      : y = 27148.920727515386
Fp = 0.309788107190237, Ft = 2.2662
Fp < Ft => модель адекватна
```

Висновок

В даній лабораторній роботі було проведено трьохфакторний експеримент і отримано адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.