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 №6
З дисципліни «Методи оптимізації та планування»
Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІО-93
Яблоновський А.О - 9330

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

327	10	60	-35	10	-30	45	6,7+2,0*x1+2,4*x2+1,6*x3+5,7*x1*x1+0,7*x2*x2+2,6*x3*x3+8,7*x1*x2+0,8*x1*x3+3,1*x2*x3+1,3*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 = [6.7, 2.0, 2.4, 1.6, 5.7, 0.7, 2.6, 8.7, 0.8, 3.1, 1.3]
    return regression_equation(x1, x2, x3, coeffs)


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 = [[10, -35, -30],
                  [10, -35, 45],
                  [10, 10, -30],
                  [10, 10, 45],
                  [60, -35, -30],
                  [60, -35, 45],
                  [60, 10, -30],
                  [60, 10, 45],
                  [-61.9, 10, 22.5],
                  [-41.9, 10, 22.5],
                  [-10, -41.9, 22.5],
                  [-10, 61.9, 22.5],
                  [-10, 10, -29.4],
                  [-10, 10, 74.4],
                  [-10, 10, 22.5]]


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 result data
    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 роботи програми:

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.1453488372093023; 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
+10     -35     -30     -350    -300    +1050   +10500  +100    +1225   +900    +96822.2    +96815.2    +96816.2
+10     -35     +45     -350    +450    -1575   -15750  +100    +1225   +2025   -136280.3   -136270.3   -136275.3
+10     +10     -30     +100    -300    -300    -3000   +100    +100    +900    -24962.3    -24960.3    -24953.3
+10     +10     +45     +100    +450    +450    +4500   +100    +100    +2025   +44353.2    +44349.2    +44349.2
+60     -35     -30     -2100   -1800   +1050   +63000  +3600   +1225   +900    +545438.2   +545440.2   +545447.2
+60     -35     +45     -2100   +2700   -1575   -94500  +3600   +1225   +2025   -826899.3   -826895.3   -826904.3
+60     +10     -30     +600    -1800   -300    -18000  +3600   +100    +900    -150757.3   -150756.3   -150756.3
+60     +10     +45     +600    +2700   +450    +27000  +3600   +100    +2025   +247424.2   +247430.2   +247427.2
-61.9   +10     +22.5   -619.0  -1392.75 +225.0  -13927.5 +3831.61 +100    +506.25 -121109.162 -121116.162 -121111.162
-41.9   +10     +22.5   -419.0  -942.75 +225.0   -9427.5 +1755.61 +100    +506.25 -82124.962  -82124.962  -82130.962
-10     -41.9   +22.5   +419.0  -225.0  -942.75  +9427.5 +100    +1755.61 +506.25 +87904.556  +87905.556  +87905.556
-10     +61.9   +22.5   -619.0  -225.0  +1392.75 -13927.5 +100    +3831.61 +506.25 -108446.524 -108449.524 -108443.524
-10     +10     -29.4   -100    +294.0  -294.0   +2940.0 +100    +100    +864.36 +25928.728  +25929.728  +25926.728
-10     +10     +74.4   -100    -744.0  +744.0   -7440.0 +100    +100    +5535.36 -56169.692  -56169.692  -56169.692
-10     +10     +22.5   -100    -225.0  +225.0   -2250.0 +100    +100    +506.25 -18617.675  -18622.675  -18625.675

Рівняння регресії: y = +7.43 +2.00x1 +2.38x2 +1.60x3 +5.70x12 +0.70x13 +2.60x23 +8.70x123 +0.80x1^2 +3.10x2^2 +1.30x3^2
Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 7.429, 2.0, 2.378, 1.603, 5.701, 0.7, 2.6, 8.7, 0.8, 3.1, 1.3
Коефіцієнти ts: 18.02, 4.85, 5.77, 3.89, 13.83, 1.70, 6.31, 21.11, 1.94, 7.52, 3.15
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 важливий; β2 важливий; β3 важливий; β12 важливий; β13 неважливий; β23 важливий; β123 важливий; β11 неважливий; β22 важливий; β33 важливий
Рівняння регресії: y = +7.43 +2.00x1 +2.38x2 +1.60x3 +5.70x12 +2.60x23 +8.70x123 +3.10x2^2 +1.30x3^2
Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -35        x2 = -30        x3 = -350       : y = 96818.39721924675
x1 = -35        x2 = 45         x3 = -350       : y = -136274.06204398038
x1 = 10         x2 = -30        x3 = 100        : y = -24957.146331934444
x1 = 10         x2 = 45         x3 = 100        : y = 44350.571966534844
x1 = -35        x2 = -30        x3 = -2100      : y = 545441.7150802148
x1 = -35        x2 = 45         x3 = -2100      : y = -826899.9870350539
x1 = 10         x2 = -30        x3 = 600        : y = -150756.9467352003
x1 = 10         x2 = 45         x3 = 600        : y = 247427.44902518147
x1 = 10         x2 = 22.5       x3 = -619.0     : y = -121111.54148093885
x1 = 10         x2 = 22.5       x3 = -419.0     : y = -82127.15631396136
x1 = -41.9      x2 = 22.5       x3 = 419.0      : y = 87904.20019632598
x1 = 61.9       x2 = 22.5       x3 = -619.0     : y = -108446.45119251372
x1 = 10         x2 = -29.4      x3 = -100       : y = 25926.95897401994
x1 = 10         x2 = 74.4       x3 = -100       : y = -56170.06523288431
x1 = 10         x2 = 22.5       x3 = -100       : y = -18622.40042838579
Fp = 0.5339045604740378, Ft = 2.4205
Fp < Ft => модель адекватна
```

Висновок:

В даній лабораторній роботі я провів трьохфакторний експеримент і отримав адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.