Міністерство освіти і науки України

Національний технічний університет України «КПІ» імені Ігоря Сікорського

Кафедра обчислювальної техніки ФІОТ

ЗВІТ

з лабораторної роботи №6

з навчальної дисципліни «МЕТОДИ ОБЧИСЛЕННЯ ТА ПЛАНУВАННЯ ЕКСПЕРИМЕНТУ»

Тема:

« Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами »

Виконав:

студент 2-го курсу ФІОТ

групи ІО-91

Денисенко М. О.

Варіант - 7

Перевірив:

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;+ ; - ; 0 для 1, 2, 3.

3. Значення функції відгуку знайти за допомогою підстановки в формулу:

yi = f(x1, x2, x3) + 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.

107	-5	15	-15	35	15	30	3,9+5,6*x1+7,9*x2+7,3*x3+2,0*x1*x1+0,5*x2*x2+4,2*x3*x3+1,5*x1*x2+0,1*x1*x3+9,9*x2*x3+5,3*x1*x2*x3

Код 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

SYMBOLS = ['\u2219', '\u00B2', '\u2260', '\u2248']
# [0] - 'знак умножения', [1] - 'степень квадрат', [2] - 'не равно', [3] - 'примерно равно',

COLUMNS = ["X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", f"X1{SYMBOLS[1]}",
f"X2{SYMBOLS[1]}", f"X3{SYMBOLS[1]}"]

x1min = -5
x1max = 15
x2min = -15
x2max = 35
x3min = 15
x3max = 30

x01 = (x1min + x1max) / 2
xl1 = -1.73 * (x1max - x01) + x01
xL1 = 1.73 * (x1max - x01) + x01
x02 = (x2min + x2max) / 2
xl2 = -1.73 * (x2max - x02) + x02
xL2 = 1.73 * (x2max - x02) + x02
x03 = (x3min + x3max) / 2
xl3 = -1.73 * (x3max - x03) + x03
xL3 = 1.73 * (x3max - x03) + x03
```

```
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 = [[x1min, x2min, x3min],
                  [x1min, x2max, x3max],
                  [x1max, x2min, x3max],
                  [x1max, x2max, x3min],
                  [x1min, x2min, x3max],
                  [x1min, x2max, x3min],
                  [x1max, x2min, x3min],
                  [x1max, x2max, x3max],
                  [xl1, x02, x03],
                  [xL1, x02, x03],
                  [x01, xl2, x03],
                  [x01, xL2, x03],
                  [x01, x02, xl3],
                  [x01, x02, xL3],
                  [x01, x02, x03]]


def regression_equation(x1, x2, x3, coefficients, importance=None):
    if importance is None:
        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(coefficients, factors_array), importance)])


def func(x1, x2, x3):
    # 3.9 + 5.6*x1 + 7.9*x2 + 7.3*x3 + 2.0*x1*x1 + 0.5*x2*x2 + 4.2*x3*x3 + 1.5*x1*x2 + 0.1*x1*x3 + 9.9*x2*x3
+ 5.3*x1*x2*x3;
    coefficients = [3.9, 5.6, 7.9, 7.3, 2, 0.5, 4.2, 1.5, 0.1, 9.9, 5.3]
    return regression_equation(x1, x2, x3, coefficients)


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):
    labels_table = list(map(lambda x: x.ljust(10), COLUMNS + [f"Y{i + 1}" for i in range(m)]))
    rows_table = [list(factors[i]) + list(y_vals[i]) for i in range(N)]
    print("\nМатриця планування для натуралізованих факторів:")
    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(beta_coefficients):
    ind = [[1, 2], [1, 3], [2, 3]]
    equation = str(round(beta_coefficients[0], 3))
    for i in range(1, 11):
        sign = " + " if beta_coefficients[i] > 0 else " - "
        if i < 4:
            equation += f"{sign}{abs(round(beta_coefficients[i], 3))}{SYMBOLS[0]}x{i}"
        elif i < 7:
            equation += "{0}{1:.3f}{2}x{3}{2}x{4}".format(sign, abs(beta_coefficients[i]), SYMBOLS[0], ind[i - 4][0],
                                                          ind[i - 4][1])
        elif i == 7:
            equation += "{0}{1:.3f}{2}x1{2}x2{2}x3".format(sign, abs(beta_coefficients[i]), SYMBOLS[0])
        else:
            equation += f"{sign}{abs(round(beta_coefficients[i], 3))}{SYMBOLS[0]}x{i - 7}{SYMBOLS[1]}"
    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)
    coefficients = [[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(coefficients, free_values)
    return list(beta_coefficients)


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__()


def cochran_criteria(m, N, y_table):
    print(f"\nПеревірка рівномірності дисперсій за критерієм Кохрена: m = {m}, N = {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
    q = 0.05
    gt = get_cochran_value(f1, f2, q)
    print(f"Gp = {gp}; Gt = {gt}; f1 = {f1}; f2 = {f2}; q = {q}")
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні")
        return True
```

```
    else:
        print("Gp > Gt => дисперсії нерівномірні")
        return False


def get_student_value(f3, q):
    return Decimal(abs(t.ppf(q / 2, f3))).quantize(Decimal('.0001')).__float__()


def student_criteria(m, N, y_table, beta_coefficients):
    print(f"\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {m}, N = {N} ")
    average_variation = numpy.average(list(map(numpy.var, y_table)))
    set_factors_table(natural_plan)
    variation_beta_s = average_variation / N / m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = numpy.array([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 = ["важливий" if el > t_our else "неважливий" for el in list(t_i)]
    # print result data

    b_coef = "Оцінки коефіцієнтів βs:"
    for beta in beta_coefficients:
        b_coef += f" {round(beta, 3)},"
    print(b_coef.rstrip(','))

    t_values = "Коефіцієнти ts:"
    for t in t_i:
        t_values += f" {round(t, 3)},"
    print(t_values.rstrip(','))

    print(f"f3 = {f3}; q = {q}; tтабл = {t_our}")
    beta_i = ["βs0", "βs1", "βs2", "βs3", "βs12", "βs13", "βs23", "βs123", "βs1.1", "βs2.2", "βs3.3"]
    for i in range(len(beta_i)):
        print(f"{beta_i[i]} - {importance[i]}")
    return importance


def get_fisher_value(f3, f4, q):
    return Decimal(abs(f.isf(q, f4, f3))).quantize(Decimal('.0001')).__float__()


def fisher_criteria(m, N, d, x_table, y_table, b_coefficients):
    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])
    # print(theoretical_y)
    average_y = numpy.array(list(map(lambda el: numpy.average(el), y_table)))
    s_ad = m / (N - d) * sum((theoretical_y - average_y) ** 2)
    # print(s_ad)
    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([f"{el[0]}: y = {el[1]}" 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)
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)
```

Результат роботи:

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.24462365591397842; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні

Матриця планування для натуралізованих факторів:
X1      X2      X3      X1X2    X1X3    X2X3    X1X2X3  X1²     X2²     X3²     Y1        Y2        Y3
-5      -15      15      75     -75     -225     1125    25      225     225     4247.4    4239.4    4242.4
-5       35      30     -175    -150     1050    -5250   25      1225    900     13478.4   13477.4   13477.4
 15     -15      30     -225     450     -450    -6750   225     225     900     -5035.6   -5036.6   -5034.6
 15      35      15      525     225      525     7875   225     1225    225     28996.4   28996.4   28993.4
-5      -15      30      75     -150     -450     2250   25      225     900     8641.4    8632.4    8631.4
-5       35      15     -175    -75       525    -2625   25      1225    225     11566.4   11568.4   11562.4
 15     -15      15     -225     225     -225    -3375   225     225     225     -2825.6   -2818.6   -2823.6
 15      35      30      525     450      1050    15750   225     1225    900     46812.4   46810.4   46815.4
-12.3    10.0    22.5   -123.0  -276.75  225.0   -2767.5 151.29  100.0   506.25  279.899   278.899   276.899
 22.3    10.0    22.5    223.0   501.75  225.0    5017.5  497.29  100.0   506.25  13258.009 13259.009 13265.009
 5.0    -33.25   22.5   -166.25  112.5   -748.125 -3740.625 25.0  1105.562 506.25  4538.856  4538.856  4534.856
 5.0     53.25   22.5    266.25  112.5   1198.125 5990.625 25.0   2835.562 506.25  45980.331 45979.331 45980.331
 5.0     10.0    9.525   50.0    47.625  95.25    476.25  25.0    100.0   90.726  2897.016  2893.016  2891.016
 5.0     10.0    35.475  50.0    177.375 354.75   1773.75 25.0    100.0   1258.476 12370.551 12369.551 12373.551
 5.0     10.0    22.5    50.0    112.5   225.0    1125.0  25.0    100.0   506.25  6737.525  6742.525  6742.525

Рівняння регресії: y = 2.62 + 5.75·x1 + 7.973·x2 + 7.647·x3 + 1.985·x1·x2 + 0.494·x1·x3 + 4.197·x2·x3 + 1.501·x1·x2·x3 + 0.093·x1² + 9.899·x2² + 5.29·x3²

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 2.62, 5.75, 7.973, 7.647, 1.985, 0.494, 4.197, 1.501, 0.093, 9.899, 5.29
Коефіцієнти ts: 7.487, 16.431, 22.782, 21.852, 5.672, 1.411, 11.992, 4.288, 0.265, 28.287, 15.117
f3 = 30; q = 0.05; tтабл = 2.0423
βs0 - важливий
βs1 - важливий
βs2 - важливий
βs3 - важливий
βs12 - важливий
βs13 - неважливий
βs23 - важливий
βs123 - важливий
βs1.1 - неважливий
βs2.2 - важливий
βs3.3 - важливий

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -15        x2 =  15        x3 =  75        : y = 4244.746891011498
x1 =  35        x2 =  30        x3 = -175       : y = 13478.78517746424
x1 = -15        x2 =  30        x3 = -225       : y = -5034.652885904795
x1 =  35        x2 =  15        x3 =  525       : y = 28994.63968465491
x1 = -15        x2 =  30        x3 =  75        : y = 8637.286415625076
x1 =  35        x2 =  15        x3 = -175       : y = 11566.245652851112
x1 = -15        x2 =  15        x3 = -225       : y = -2822.1924105178705
x1 =  35        x2 =  30        x3 =  525       : y = 46812.512542601195
x1 =  10.0      x2 =  22.5      x3 = -123.0     : y = 276.11914862328695
x1 =  10.0      x2 =  22.5      x3 =  223.0     : y = 13261.17165697704
x1 = -33.25     x2 =  22.5      x3 = -166.25    : y = 4535.202173107104
x1 =  53.25     x2 =  22.5      x3 =  266.25    : y = 45980.37753108925
x1 =  10.0      x2 =  9.525     x3 =  50.0      : y = 2893.329149040409
x1 =  10.0      x2 =  35.475    x3 =  50.0      : y = 12369.616693287886
x1 =  10.0      x2 =  22.5      x3 =  50.0      : y = 6740.872178258112
Fp = 3.43732799427176, Ft = 2.6896
Fp > Ft => модель неадекватна

Process finished with exit code 0
```

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії. При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння

регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості

$q = 0.05$.