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

Лабораторна робота №3
з дисципліни «Методи оптимізації та планування експерименту»

Виконав:
Студентка ФІОТ
групи ІО-93
Макоткін В. М.

Перевірив:
Регіда П.Г.

Київ – 2021р

Варіант: 319

Мета: Провести дробовий трьох факторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Завдання за варіантом:

319	-20	30	20	60	-20	-5

Код програми *(main.py)*:

```
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial
import numpy as np
import random


x_vals = [(-20,  30),
          (20,   60),
          (-20, -5)]


x_avg_max = (max(x_vals[0]) + max(x_vals[1]) + max(x_vals[2])) / 3
x_avg_min = (min(x_vals[0]) + min(x_vals[1]) + min(x_vals[2])) / 3


y_max = 200 + int(x_avg_max)
y_min = 200 + int(x_avg_min)


def plan_matrix(n, m):
    y = np.zeros(shape=(n, m))

    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)
    x_norm = np.array([[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]])

    x_norm = x_norm[:len(y)]

    x = np.ones(shape=(len(x_norm), len(x_norm[0])))
    for i in range(len(x_norm)):
        for j in range(1, len(x_norm[i])):
            if x_norm[i][j] == -1: x[i][j] = x_vals[j - 1][0]
            else:                  x[i][j] = x_vals[j - 1][1]
```

```
    print('\nМатриця планування')
    print(np.concatenate((x, y), axis=1))

    return x, y, x_norm


def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


def find_coefficient(x, y_aver, n):
    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n
    my  = sum(y_aver) / n
    a12 = sum([x[i][1] * x[i][2] for i in range(len(x))]) / n
    a13 = sum([x[i][1] * x[i][3] for i in range(len(x))]) / n
    a23 = sum([x[i][2] * x[i][3] for i in range(len(x))]) / n
    a11 = sum([i ** 2 for i in x[:, 1]]) / n
    a22 = sum([i ** 2 for i in x[:, 2]]) / n
    a33 = sum([i ** 2 for i in x[:, 3]]) / n
    a1  = sum([y_aver[i] * x[i][1] for i in range(len(x))]) / n
    a2  = sum([y_aver[i] * x[i][2] for i in range(len(x))]) / n
    a3  = sum([y_aver[i] * x[i][3] for i in range(len(x))]) / n

    X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23],
[mx3, a13, a23, a33]]
    Y = [my, a1, a2, a3]
    B = [round(i, 2) for i in solve(X, Y)]
    print('\nРівняння регресії')
    print(f'{B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return B


def s_kv(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(s)
    return res


def cochrane_criteria(y, y_aver, n, m):
    S_kv =  s_kv(y, y_aver, n, m)
    Gp =    max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(3):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res


def studenta_criteria(x, y, y_aver, n, m):
    S_kv =      s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

    s_Bs =  (s_kv_aver / n / m) ** 0.5
```

```
    Bs =    bs(x, y_aver, n)
    ts =    [abs(B) / s_Bs for B in Bs]

    return ts


def fishera_criteria(y, y_aver, y_new, n, m, d):
    S_ad =      m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in
range(len(y))])
    S_kv =      s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def cochrena(f1, f2, q=0.05):
    q1 =           q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)


def main(n, m):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q =  0.05

    student = partial(t.ppf, q=1 - 0.025)
    t_student = student(df=f3)
    G_kr =      cochrena(f1, f2)

    x, y, x_norm = plan_matrix(n, m)
    y_aver =       [round(sum(i) / len(i), 2) for i in y]

    B =  find_coefficient(x, y_aver, n)
    Gp = cochrane_criteria(y, y_aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити ксть дослідів")
        m += 1
        main(n, m)

    ts = studenta_criteria(x_norm[:, 1:], y, y_aver, n, m)
    print('\nКритерій Стьюдента:\n', ts)
    res =     [t for t in ts if t > t_student]
    final_k = [B[ts.index(i)] for i in ts if i in res]
    print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
            [i for i in B if i not in final_k]))

    y_new = list()
    for j in range(n):
        y_new.append(regression([x[j][ts.index(i)] for i in ts if i in res],
final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    print(y_new)

    d =     len(res)
    f4 =    n - d
    F_p =   fishera_criteria(y, y_aver, y_new, n, m, d)
```

```
    fisher = partial(f.ppf, q=1 - 0.05)
    f_t =    fisher(dfn=f4, dfd=f3)

    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')


if __name__ == '__main__':
    M, N = 4, 4
    main(M, N)
```

Висновок:

Провів дробовий трьох факторний експеримент. Склав матрицю планування, знайшов коефіцієнти рівняння регресії, провів 3 статистичні перевірки.