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:
Студент 2 курсу ФІОТ
групи ІО-91
Лаппо М.О.

Перевірив:
Регіда П.Г.

Київ 2021

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

$$y_{max} = 200 + x_{cp\ max};$$

$$y_{min} = 200 + x_{cp\ min}$$

де $x_{cp\,max} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cp\,min} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

Варіант:

116	-10	50	-20	60	-20	5

Роздруківка програми:

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

x_r = [(-10, 50), (-20, 60), (-20, 5)]
x_aver_max = (50 + 60 + 5)/ 3
x_aver_min = (-10 - 20 - 20) / 3
y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)


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


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_n = 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_n = x_n[:len(y)]

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

```
            else:
                x[i][j] = x_r[j-1][1]

    print('Матриця планування')
    print(np.concatenate((x, y), axis=1))

    return x, y, x_n


def find_coeff(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('Рівняння 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 krit_cochrena(y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('Перевірка за критерієм Кохрена')
    return Gp

def bs(x, y, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(3):  # 4 - кєть факторів
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res

def krit_studenta(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, y_aver, n)
    ts = [abs(B) / s_Bs for B in Bs]
    return ts

def krit_fishera(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
```

```python
def cohren(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 = cohren(f1, f2)

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

    B = find_coeff(x, y_aver, n)

    Gp = krit_cochrena(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 = krit_studenta(x_norm[:, 1:], y, y_aver, n, m)
    print('Критерій 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 = []
    for j in range(n):
        y_new.append(reg([x[j][ts.index(i)] for i in ts if i in res], final_k))

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

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

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

    print('Перевірка 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__':
    main(4, 4)
```

Результати роботи програми:

```
C:\TeoriyaUmovirnostey\venv\Scripts\python.exe C:/TeoriyaUmovirnostey/тесе/main4.py
Матриця планування
[[  1. -10. -20. -20. 234. 185. 213. 189.]
 [  1. -10.  60.   5. 228. 184. 184. 187.]
 [  1.  50. -20.   5. 191. 224. 223. 199.]
 [  1.  50.  60. -20. 201. 200. 210. 208.]]
Рівняння регресії
202.58 + 0.11*x1 + -0.09*x2 + -0.1*x3
Перевірка за критерієм Кохрена
Gp = 0.4029821843532146
З ймовірністю 0.95 дисперсії однорідні.
Критерій Стьюдента:
 [52.38342029120088, 0.835563759246149, 0.8998378945727759, 0.32137067663313423]
Коефіцієнти [0.11, -0.09, -0.1] статистично незначущі, тому ми виключаємо їх з рівняння.
Значення "y" з коефіцієнтами [202.58]
[202.58, 202.58, 202.58, 202.58]
Перевірка адекватності за критерієм Фішера
Fp = 0.5672121869351916
F_t = 3.490294819497605
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```

Контрольні запитання

1. Що називається дробовим факторним експериментом?

Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту

2. Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. Для чого перевіряється критерій Стьюдента?

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння

4. Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваного об”єкта.