Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №3
ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

Виконала

студентка групи ІТ-91: Перевірила:

Луцай Катерина Сокульський О. Є.

Київ 2022

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

Завдання на лабораторну роботу

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

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

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

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

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Варіант 15

```
import numpy as np
from sklearn.preprocessing import minmax_scale

x1_min, x1_max = 10, 50
x2_min, x2_max = -20, 60
x3_min, x3_max = -20, 20

y_min = 200 + (x1_min + x2_min + x3_min)/3
y_max = 200 + (x1_max + x2_max + x3_max)/3

n = 4  # кількість точок
k = 4  # кількість коефіцієнтів

norm_factors = np.array([[1, 1, 1],
                         [1, 1, -1],
                         [1, -1, -1],
                         [-1, -1, -1]])

# отримання натуральних факторів з нормованих
def natural(norm_factors):
    natural = np.zeros_like(norm_factors)
    natural[:, 0][norm_factors[:, 0] == 1] = x1_max
    natural[:, 1][norm_factors[:, 1] == 1] = x2_max
    natural[:, 2][norm_factors[:, 2] == 1] = x3_max
```

```
    natural[:, 0][norm_factors[:, 0] == -1] = x1_min
    natural[:, 1][norm_factors[:, 1] == -1] = x2_min
    natural[:, 2][norm_factors[:, 2] == -1] = x3_min
    return natural

# регресія за факторами та коефцієнтами
def regression(matrix, coefs, n, k):
    def func(factors, coefs, k):
        y = coefs[0]
        for i in range(k-1):
            y += coefs[i+1] * factors[i]
        return y
    values = np.zeros(n)
    for f in range(n):
        values[f] = func(matrix[f], coefs, k)
    return values

# вирішення СЛР для натуральних значень і факторів
def solve_coef(factors, values):
    A = np.zeros((k, k))
    x_mean = np.mean(factors, axis=0)
    for i in range(1, k):
        for j in range(1, k):
            A[j, i] = np.mean(factors[:, i-1] * factors[:, j-1])
    A[0, 0] = n
    A[0, 1:] = x_mean
    A[1:, 0] = x_mean
    B = values * np.concatenate((np.ones(1), x_mean))
    coef = np.linalg.solve(A, B)
    return coef

# вирішення СЛР для нормованих значень і факторів
def solve_norm_coef(factors, values):
    coefs = np.zeros(k)
    coefs[0] = values.mean()
    for i in range(k-1):
        coefs[i+1] = np.mean(values * factors[:, 0])
    return coefs

print(f"X1 min: {x1_min}\tX1 max: {x1_max}")
print(f"X2 min: {x2_min}\tX2 max: {x2_max}")
print(f"X3 min: {x3_min}\tX3 max: {x3_max}")
print(f"Y min: {y_min}\tY max: {y_max}")

# отримання оптимальної кількості випробувань для однієї комбінації факторів
Gt = [9.065, 7.679, 6.841]
for i in range(2, 10):
    rand_y = np.random.randint(y_min, y_max, (n, i))
    std_y = np.std(rand_y, axis=1)**2
    print(f"Max Y dispersion: {std_y.max()}")
    Gp = std_y.max()/std_y.mean()
    print(f"m = {i} Gp: {Gp}\tGt: {Gt[i-2]}")
    if Gp < Gt[i-2]:
        m = i  # кількість випробувань
        break

print(f"Mean Y dispersion: {std_y.mean()}")

factors = natural(norm_factors)
```

```
# генерація Y значень для експериментів
rand_y = np.random.randint(y_min, y_max, (n, m))
norm_y = minmax_scale(rand_y.reshape(n*m), feature_range=(-1,1)).reshape(n ,m)

# перевірка за критеріїм стюдента
coefs_value = np.zeros(4)
coefs_value[0] = np.mean(rand_y.mean(axis=1))
for i in range(3):
    coefs_value[i+1] = np.mean(rand_y.mean(axis=1) * factors[:, i])
stud_crit = np.abs(coefs_value) / np.sqrt(std_y.mean()/(n*m))
ts = [2.776, 2.306, 2.179]
sig_coefs = len(stud_crit[stud_crit > ts[m-2]])
print(f"All coefs are significant: {sig_coefs == k}\t{sig_coefs}")

norm_plan = np.concatenate((norm_factors, norm_y), axis=1)
print(f"Normalized plan:\n{norm_plan}")
plan = np.concatenate((factors, rand_y), axis=1)
print(f"Natural plan:\n{plan}")

norm_coefs = solve_coef(norm_factors, norm_y.mean(axis=1))
print("Normalized coefs", norm_coefs)
coefs = solve_coef(factors, rand_y.mean(axis=1))
print("Natural coefs", coefs)

reg_val = regression(norm_factors, norm_coefs, n, k)

# перевірка критерію Фішера
f3 = n * (m - 1) # 4/8/12
f4 = 1 if n - sig_coefs == 0 else n - sig_coefs
disp = m/f4 * np.sum((reg_val - norm_y.mean(axis=1))**2)
Fp = disp / std_y.mean()
Ft = [[7.7, 5.3, 4.8],
      [6.9, 4.5, 3.9]]
print(f"Fisher crit: {Fp < Ft[f4-1][m-2]}")
```

Результат виконання:

```
X1 min: 10   X1 max: 50

X2 min: -20  X2 max: 60

X3 min: -20  X3 max: 20

Y min: 190.0 Y max: 243.33333333333334

Max Y dispersion: 225.0

m = 2 Gp: 3.4917555771096023     Gt: 9.065

Mean Y dispersion: 64.4375

All coefs are significant: True  4

Normalized plan:

[[ 1.         1.         1.         0.19047619  0.33333333]
```

```
 [ 1.          1.         -1.         -1.          0.57142857]
 [ 1.         -1.         -1.         -1.         -0.57142857]
 [-1.         -1.         -1.         -0.61904762  1.        ]]
Natural plan:
[[ 50  60  20 221 224]
 [ 50  60 -20 196 229]
 [ 50 -20 -20 196 205]
 [ 10 -20 -20 204 238]]
Normalized coefs [ 0.07653061 -0.24659864  0.20238095 -0.15816327]
Natural coefs [10.87264151  4.25813679 -0.47647406 -1.82134434]
Fisher crit: True
```

Висновки: було проведено дробовий трьохфакторний експеримент. Скласти матрицю планування, знайдено коефіцієнти рівняння регресії, проведено 3 статистичні перевірки.