Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Методи оптимізації та планування експерименту»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІВ-82

Стрілецький В.Р.

ПЕРЕВІРИВ:

ас. Регіда П.Г.

Київ – 2019

**Мета:**

Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

**Варіант 223**

x1\_min: -5 x1\_max: 15

x2\_min: -25 x2\_max: 10

x3\_min: -5 x3\_max: 20

**Текст програми:**

import random  
import numpy as np  
import sklearn.linear\_model as lm  
from scipy.stats import f, t  
from functools import partial  
  
  
def regression(x, b):  
 y = sum([x[i]\*b[i] for i in range(len(x))])  
 return y  
  
x\_range = ((-5, 15), (-25, 10), (-5, 20))  
  
x\_aver\_max = sum([x[1] for x in x\_range]) / 3  
x\_aver\_min = sum([x[0] for x in x\_range]) / 3  
  
y\_max = 200 + int(x\_aver\_max)  
y\_min = 200 + int(x\_aver\_min)  
  
  
# квадратна дисперсія  
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(round(s, 3))  
 return res  
  
def plan\_matrix(n, m):  
 print(f'\nГереруємо матрицю планування для n = {n}, m = {m}')  
 y = np.zeros(shape=(n,m), dtype=np.int64)  
 for i in range(n):  
 for j in range(m):  
 y[i][j] = random.randint(y\_min, y\_max)  
  
 x\_norm = [[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]]  
  
 for x in x\_norm:  
 x.append(x[1]\*x[2])  
 x.append(x[1]\*x[3])  
 x.append(x[2]\*x[3])  
 x.append(x[1]\*x[2]\*x[3])  
 x\_norm = np.array(x\_norm[:len(y)])  
  
 x = np.ones(shape=(len(x\_norm), len(x\_norm[0])), dtype=np.int64)  
 for i in range(len(x\_norm)):  
 for j in range(1, 4):  
 if x\_norm[i][j] == -1:  
 x[i][j] = x\_range[j-1][0]  
 else:  
 x[i][j] = x\_range[j-1][1]  
  
 for i in range(len(x)):  
 x[i][4] = x[i][1] \* x[i][2]  
 x[i][5] = x[i][1] \* x[i][3]  
 x[i][6] = x[i][2] \* x[i][3]  
 x[i][7] = x[i][1] \* x[i][3] \* x[i][2]  
  
 print('\nX:\n', x)  
 print('\nX нормоване:\n', x\_norm)  
 print('\nY:\n', y)  
  
 return x, y, x\_norm  
  
def find\_coef(X, Y, norm=False):  
 skm = lm.LinearRegression(fit\_intercept=False)  
 skm.fit(X, Y)  
 B = skm.coef\_  
  
 if norm == 1:  
 print('\nКоефіцієнти рівняння регресії з нормованими X:')  
 else:  
 print('\nКоефіцієнти рівняння регресії:')  
 B = [round(i, 3) for i in B]  
 print(B)  
 return B  
  
def kriteriy\_cochrana(y, y\_aver, n, m):  
 f1 = m - 1  
 f2 = n  
 q = 0.05  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 Gp = max(S\_kv) / sum(S\_kv)  
 print('\nПеревірка за критерієм Кохрена')  
 return Gp  
  
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)  
  
# оцінки коефіцієнтів  
def bs(x, y, y\_aver, n):  
 res = [sum(1 \* y for y in y\_aver) / n]  
 for i in range(7): # 8 - ксть факторів  
 b = sum(j[0] \* j[1] for j in zip(x[:,i], y\_aver)) / n  
 res.append(b)  
 return res  
  
def kriteriy\_studenta(x, y, y\_aver, n, m):  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 s\_kv\_aver = sum(S\_kv) / n  
  
 # статиcтична оцінка дисперсії  
 s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5  
 Bs = bs(x, y, y\_aver, n)  
 ts = [round(abs(B) / s\_Bs, 3) for B in Bs]  
  
 return ts  
  
def kriteriy\_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  
  
def check(X, Y, B, n, m, norm=False):  
 if norm == False:  
 print('\n\tПеревірка рівняння з натуральними значеннями факторів:')  
 else:  
 print('\n\tПеревірка рівняння з нормованими значеннями факторів:')  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 q = 0.05  
  
 ###  
 ### табличні значення  
 student = partial(t.ppf, q=1-0.025)  
 t\_student = student(df=f3)  
  
 G\_kr = cohren(f1, f2)  
 ###  
  
 y\_aver = [round(sum(i) / len(i), 3) for i in Y]  
 print('\nСереднє значення y:', y\_aver)  
  
 disp = s\_kv(Y, y\_aver, n, m)  
 print('Дисперсія y:', disp)  
  
 Gp = kriteriy\_cochrana(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 кількість дослідів")  
 m += 1  
 main(n, m)  
  
 ts = kriteriy\_studenta(X[:,1:], Y, y\_aver, n, m)  
 print('\nКритерій Стьюдента:\n',ts)  
 res = [t for t in ts if t > t\_student]  
 final\_k = [B[i] for i in range(len(ts)) if ts[i] in res]  
 print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'.format([round(i, 3) for i in B if i not in final\_k]))  
  
 y\_new = []  
 for j in range(n):  
 y\_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res], final\_k))  
  
 print(f'\nЗначення "y" з коефіцієнтами {final\_k}')  
 print(y\_new)  
  
  
 d = len(res)  
 if d >= n:  
 print('\nF4 <= 0')  
 print('')  
 return  
 f4 = n - d  
  
 F\_p = kriteriy\_fishera(Y, y\_aver, y\_new, n, m, d)  
  
 fisher = partial(f.ppf, q=0.95)  
 f\_t = fisher(dfn=f4, dfd=f3) # табличне знач  
 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')  
  
  
def main(n, m):  
  
 X, Y, X\_norm = plan\_matrix(n,m)  
  
 y\_aver = [round(sum(i) / len(i), 3) for i in Y]  
  
 B\_norm = find\_coef(X\_norm, y\_aver, norm=True)  
  
 check(X\_norm, Y, B\_norm, n, m, norm=True)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main(8, 3)