**Міністерство освіти і науки України**

**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**»

Тема: «Проведення двофакторного експерименту з використанням лінійного рівняння регресії»

**Виконав:**

Студент 2 курсу кафедри ОТ ФІОТ,

Навчальної групи ІО-91

Щурик Є.В.

Номер залікової книжки - 9129

**Перевірив:**

Ас. Регіда П.Г.

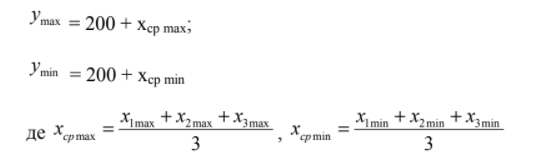
**Київ 2021**

**Мета:**

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

**Завдання:**

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 відповідності з варіантом діапазону, зазанченого далі(випадковим чином).



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

3) Провести 3 статистичні перевірки.

4) Написати коп’ютерну програму, яка усе це виконує.

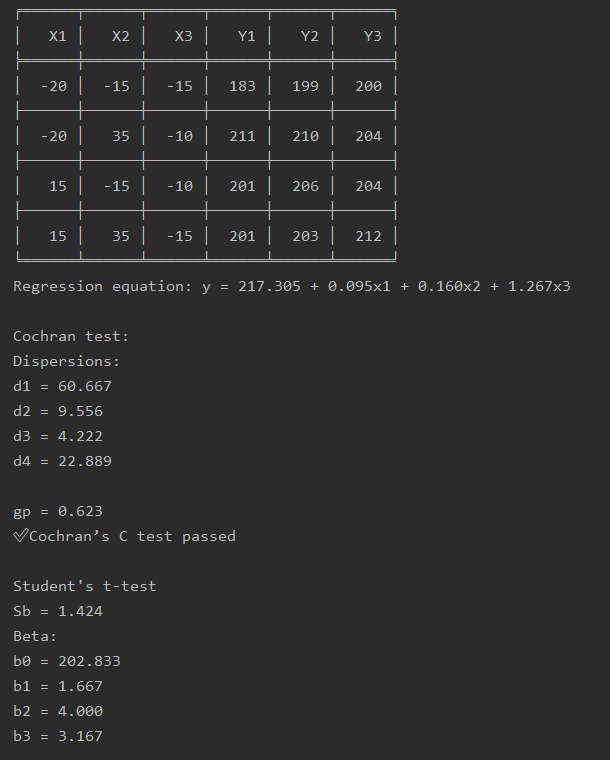
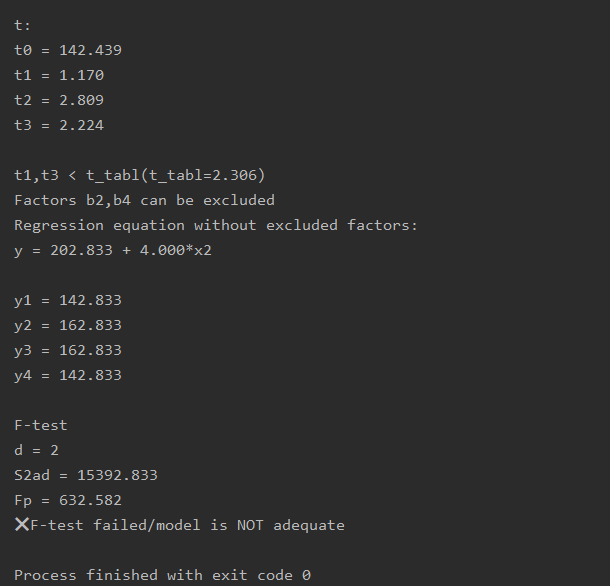
**Варіант:**



**Роздруківка коду:**

import math  
import random  
import numpy as np  
from tabulate import tabulate  
  
  
m = 3  
Y = []  
  
x1min = -20  
x1max = 15  
  
x2min = -15  
x2max = 35  
  
x3min = -15  
x3max = -10  
  
  
x0 = [1, 1, 1, 1]  
x1 = [-1, -1, 1, 1]  
x2 = [-1, 1, -1, 1]  
x3 = [-1, 1, 1, -1]  
  
x = [x0, x1, x2, x3]  
X = [  
 [x1min, x1min, x1max, x1max],  
 [x2min, x2max, x2min, x2max],  
 [x3min, x3max, x3max, x3min],  
]  
y\_min = 200 + (x1min + x2min + x3min) / 3  
y\_max = 200 + (x1max + x2max + x3max) / 3  
  
y1 = []  
y2 = []  
y3 = []  
  
for i in range(4):  
 y1.append(random.randint(int(y\_min), int(y\_max)))  
 y2.append(random.randint(int(y\_min), int(y\_max)))  
 y3.append(random.randint(int(y\_min), int(y\_max)))  
  
Y.append(y1)  
Y.append(y2)  
Y.append(y3)  
  
# Show X and Y values as a table  
table = list()  
table.append([x[0] for x in X] + [x[0] for x in Y])  
table.append([x[1] for x in X] + [x[1] for x in Y])  
table.append([x[2] for x in X] + [x[2] for x in Y])  
table.append([x[3] for x in X] + [x[3] for x in Y])  
print(  
 tabulate(  
 table,  
 headers=["X1", "X2", "X3", "Y1", "Y2", "Y3"],  
 floatfmt=".4f",  
 tablefmt="fancy\_grid",  
 )  
)  
  
# Calculate average for every row  
y1avg = sum([x[0] for x in Y]) / 3  
y2avg = sum([x[1] for x in Y]) / 3  
y3avg = sum([x[2] for x in Y]) / 3  
y4avg = sum([x[3] for x in Y]) / 3  
yavg = [y1avg, y2avg, y3avg, y4avg]  
  
  
def generate\_random\_y(ymin, ymax):  
  
 for i in range(5):  
 Y.append(  
 [  
 random.randint(ymin, ymax),  
 random.randint(ymin, ymax),  
 random.randint(ymin, ymax),  
 ]  
 )  
  
def generate\_test\_y():  
  
 # You can use an example Y values. Just set test\_y variable to True when creating Lab2 object  
 Y.append([9, 15, 20])  
 Y.append([10, 14, 18])  
 Y.append([11, 10, 12])  
 Y.append([15, 12, 10])  
 Y.append([9, 14, 16])  
  
  
# Normalize regression factors  
  
mx1 = sum(X[0]) / 4  
mx2 = sum(X[1]) / 4  
mx3 = sum(X[2]) / 4  
my = (y1avg + y2avg + y3avg + y4avg) / 4  
a1 = (  
 X[0][0] \* y1avg  
 + X[0][1] \* y2avg  
 + X[0][2] \* y3avg  
 + X[0][3] \* y4avg  
) / 4  
a2 = (  
 X[1][0] \* y1avg  
 + X[1][1] \* y2avg  
 + X[1][2] \* y3avg  
 + X[1][3] \* y4avg  
) / 4  
a3 = (  
 X[2][0] \* y1avg  
 + X[2][1] \* y2avg  
 + X[2][2] \* y3avg  
 + X[2][3] \* y4avg  
) / 4  
  
a11 = sum([x \*\* 2 for x in X[0]]) / 4  
a22 = sum([x \*\* 2 for x in X[1]]) / 4  
a33 = sum([x \*\* 2 for x in X[2]]) / 4  
  
a12 = sum([X[0][i] \* X[1][i] for i in range(4)]) / 4  
a13 = sum([X[0][i] \* X[2][i] for i in range(4)]) / 4  
a23 = sum([X[1][i] \* X[2][i] for i in range(4)]) / 4  
  
a21 = a12  
a31 = a13  
a32 = a23  
  
# Calculating determinants of matrixes  
b0 = np.linalg.det(  
 [  
 [my, mx1, mx2, mx3],  
 [a1, a11, a12, a13],  
 [a2, a12, a22, a32],  
 [a3, a13, a23, a33],  
 ]  
) / np.linalg.det(  
 [  
 [1, mx1, mx2, mx3],  
 [mx1, a11, a12, a13],  
 [mx2, a12, a22, a32],  
 [mx3, a13, a23, a33],  
 ]  
)  
  
b1 = np.linalg.det(  
 [  
 [1, my, mx2, mx3],  
 [mx1, a1, a12, a13],  
 [mx2, a2, a22, a32],  
 [mx3, a3, a23, a33],  
 ]  
) / np.linalg.det(  
 [  
 [1, mx1, mx2, mx3],  
 [mx1, a11, a12, a13],  
 [mx2, a12, a22, a32],  
 [mx3, a13, a23, a33],  
 ]  
)  
  
b2 = np.linalg.det(  
 [  
 [1, mx1, my, mx3],  
 [mx1, a11, a1, a13],  
 [mx2, a12, a2, a32],  
 [mx3, a13, a3, a33],  
 ]  
) / np.linalg.det(  
 [  
 [1, mx1, mx2, mx3],  
 [mx1, a11, a12, a13],  
 [mx2, a12, a22, a32],  
 [mx3, a13, a23, a33],  
 ]  
)  
  
b3 = np.linalg.det(  
 [  
 [1, mx1, mx2, my],  
 [mx1, a11, a12, a1],  
 [mx2, a12, a22, a2],  
 [mx3, a13, a23, a3],  
 ]  
) / np.linalg.det(  
 [  
 [1, mx1, mx2, mx3],  
 [mx1, a11, a12, a13],  
 [mx2, a12, a22, a32],  
 [mx3, a13, a23, a33],  
 ]  
)  
b = [b0, b1, b2, b3]  
print(  
 "Regression equation: y = {:.3f} + {:.3f}x1 + {:.3f}x2 + {:.3f}x3".format(  
 b0, b1, b2, b3  
 )  
)  
  
print("\nCochran test:")  
  
# Cochran  
  
# Calculate dispersions for every row  
d1 = sum([((x[0] - y1avg) \*\* 2) for x in Y]) / 3  
d2 = sum([((x[1] - y2avg) \*\* 2) for x in Y]) / 3  
d3 = sum([((x[2] - y3avg) \*\* 2) for x in Y]) / 3  
d4 = sum([((x[3] - y4avg) \*\* 2) for x in Y]) / 3  
d = [d1, d2, d3, d4]  
print("Dispersions:")  
for i in range(4):  
 print("d{} = {:.3f}".format(i + 1, d[i]))  
gp = max(d) / sum(d)  
print("\ngp = {:.3f}".format(gp))  
if gp < 0.7679:  
 print("✅Cochran’s C test passed")  
else:  
 print("❌Cochran’s C test failed")  
  
print("\nStudent's t-test")  
  
# Student  
  
N = len(d)  
N = N  
sbsq = sum(d) / N  
sbsq = sbsq  
sbssq = sbsq / (m \* N)  
sbs = math.sqrt(sbssq)  
  
print("Sb = {:.3f}".format(sbs))  
  
b0 = sum([yavg[i] \* x[0][i] for i in range(0, N)]) / N  
b1 = sum([yavg[i] \* x[1][i] for i in range(0, N)]) / N  
b2 = sum([yavg[i] \* x[2][i] for i in range(0, N)]) / N  
b3 = sum([yavg[i] \* x[3][i] for i in range(0, N)]) / N  
  
b = [b0, b1, b2, b3]  
t = [abs(x) / sbs for x in b]  
  
print("Beta:")  
for i in range(4):  
 print("b{} = {:.3f}".format(i, b[i]))  
  
print("\nt:")  
for i in range(4):  
 print("t{} = {:.3f}".format(i, t[i]))  
  
print()  
  
t\_tabl = 2.306  
regr\_eq = ""  
nm = []  
xs = ["x1", "x2", "x3"]  
d = 0  
  
yd = [0, 0, 0, 0]  
for i in range(4):  
 if t[i] < t\_tabl:  
 nm.append(i)  
 d += 1  
 else:  
 for j in range(4):  
 if i == 0:  
 yd[j] += b[i]  
 else:  
 yd[j] += b[i] \* X[2][j]  
 if i == 0:  
 regr\_eq += "{:.3f} + ".format(b[0])  
 else:  
 regr\_eq += "{:.3f}\*{} + ".format(b[i], xs[i - 1])  
if len(regr\_eq) != 0:  
 regr\_eq = regr\_eq[0:-2]  
  
nmt = ",".join(["t" + str(x) for x in nm])  
nmb = ",".join(["b" + str(x + 1) for x in nm])  
print("{} < t\_tabl(t\_tabl=2.306)".format(nmt))  
print("Factors {} can be excluded".format(nmb))  
print("Regression equation without excluded factors:")  
print("y = {}\n".format(regr\_eq))  
  
for i in range(4):  
 print("y{} = {:.3f}".format(i + 1, yd[i]))  
  
d = d  
yd = yd  
  
print("\nF-test")  
  
# Fisher crit  
print("d = {}".format(d))  
sadsq = (m / (N - d)) \* sum(  
 [(yd[i] - yavg[i]) \*\* 2 for i in range(N)]  
)  
Fp = sadsq / sbsq  
print("S2ad = {:.3f}\nFp = {:.3f}".format(sadsq, Fp))  
table = [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.3, 3.1, 2.9]  
tablec = table[N - d - 1]  
if Fp < tablec:  
 print("✅F-test passed/model is adequate")  
else:  
 print("❌F-test failed/model is NOT adequate")

**Результат виконання програми:**

**** ****

**Висновок:**

В ході лабораторної роботи проведено дробовий трьохфакторний експеримент, складено матрицю планування та знайдено коефіцієнти рівняння регресії, а також проведено 3 статистичні перевірки. В ході реалізації завдання було розроблено програму, завдяки чому отримані знання закріплено практично.