Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

З дисципліни «Методи наукових досліджень»

За темою:

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів»

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи ІВ-93

Дромашко Артем

Номер у списку – 6

Варіант - 306

ПЕРЕВІРИВ:

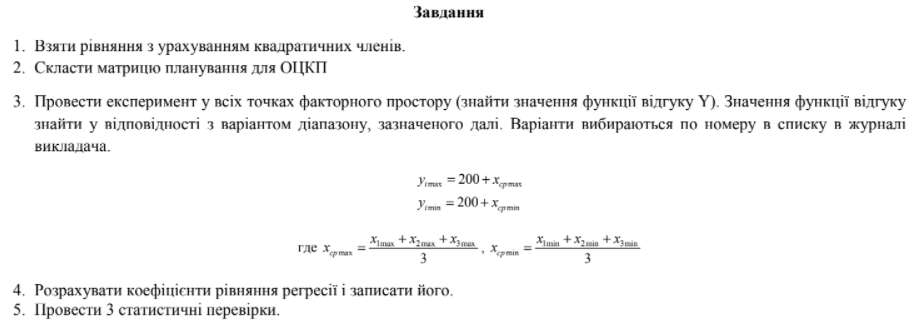
асистент

Регіда П.Г.

Київ 2021 р.

**Мета:** Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

**Завдання:**







**Програмний код**

import random  
import numpy as np  
import sklearn.linear\_model as lm  
from pyDOE2 import \*  
from scipy.stats import f, t  
from functools import partial  
  
  
# variant 306  
x\_range = ((-1, 2), (-9, 5), (-10, 7))  
avg\_x\_min = sum([x[0] for x in x\_range]) / 3  
avg\_x\_max = sum([x[1] for x in x\_range]) / 3  
y\_min = 200 + int(avg\_x\_min)  
y\_max = 200 + int(avg\_x\_max)  
  
  
def s\_kv(y, y\_average, n, m):  
 res = []  
 for i in range(n):  
 s = sum([(y\_average[i] - y[i][j]) \*\* 2 for j in range(m)]) / m  
 res.append(round(s, 3))  
 return res  
  
  
def plan\_matrix5(n, m):  
 print(f'\nГереруємо матрицю планування для n = {n}, m = {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)  
  
 if n > 14:  
 no = n - 14  
 else:  
 no = 1  
 x\_norm = ccdesign(3, center=(0, no))  
 x\_norm = np.insert(x\_norm, 0, 1, axis=1)  
  
 for i in range(4, 11):  
 x\_norm = np.insert(x\_norm, i, 0, axis=1)  
  
 l = 1.215  
  
 for i in range(len(x\_norm)):  
 for j in range(len(x\_norm[i])):  
 if x\_norm[i][j] < -1 or x\_norm[i][j] > 1:  
 if x\_norm[i][j] < 0:  
 x\_norm[i][j] = -l  
 else:  
 x\_norm[i][j] = l  
  
 def add\_sq\_nums(x):  
 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]  
 x[i][8] = x[i][1] \*\* 2  
 x[i][9] = x[i][2] \*\* 2  
 x[i][10] = x[i][3] \*\* 2  
 return x  
  
 x\_norm = add\_sq\_nums(x\_norm)  
  
 x = np.ones(shape=(len(x\_norm), len(x\_norm[0])), dtype=np.int64)  
 for i in range(8):  
 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(8, len(x)):  
 for j in range(1, 3):  
 x[i][j] = (x\_range[j - 1][0] + x\_range[j - 1][1]) / 2  
  
 dx = [x\_range[i][1] - (x\_range[i][0] + x\_range[i][1]) / 2 for i in range(3)]  
 x[8][1] = l \* dx[0] + x[9][1]  
 x[9][1] = -l \* dx[0] + x[9][1]  
 x[10][2] = l \* dx[1] + x[9][2]  
 x[11][2] = -l \* dx[1] + x[9][2]  
 x[12][3] = l \* dx[2] + x[9][3]  
 x[13][3] = -l \* dx[2] + x[9][3]  
  
 x = add\_sq\_nums(x)  
  
 print('\nX:\n', x)  
 print('\nX нормоване:\n')  
 for i in x\_norm:  
 print([round(x, 2) for x in i])  
 print('\nY:\n', y)  
  
 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\_coefficients(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)  
 print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X, B))  
 return B  
  
  
def cochran\_test(y, y\_average, n, m):  
 f1 = m - 1  
 f2 = n  
 q = 0.05  
 S\_kv = s\_kv(y, y\_average, n, m)  
 Gp = max(S\_kv) / sum(S\_kv)  
 print('\nПеревірка за критерієм Кохрена')  
 return Gp  
  
  
def cohran(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\_average, n):  
 res = [sum(1 \* y for y in y\_average) / n]  
  
 for i in range(len(x[0])):  
 b = sum(j[0] \* j[1] for j in zip(x[:, i], y\_average)) / n  
 res.append(b)  
 return res  
  
  
def student\_test(x, y, y\_average, n, m):  
 S\_kv = s\_kv(y, y\_average, n, m)  
 s\_kv\_aver = sum(S\_kv) / n  
  
 s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5  
 Bs = bs(x, y\_average, n)  
 ts = [round(abs(B) / s\_Bs, 3) for B in Bs]  
  
 return ts  
  
  
def fisher\_test(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):  
 print('\nПеревірка рівняння:')  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 q = 0.05  
  
 student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
  
 G\_kr = cohran(f1, f2)  
  
 y\_average = [round(sum(i) / len(i), 3) for i in Y]  
 print('\nСереднє значення y:', y\_average)  
  
 dispersion = s\_kv(Y, y\_average, n, m)  
 print('Дисперсія y:', dispersion)  
  
 Gp = cochran\_test(Y, y\_average, 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 = student\_test(X[:, 1:], Y, y\_average, 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 = fisher\_test(Y, y\_average, y\_new, n, m, d)  
 fisher = partial(f.ppf, q=0.95)  
 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')  
  
  
def main(n, m):  
 x5, y5, x5\_norm = plan\_matrix5(n, m)  
 y5\_average = [round(sum(i) / len(i), 3) for i in y5]  
 b5 = find\_coefficients(x5, y5\_average)  
 check(x5\_norm, y5, b5, n, m)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main(15, 8)

**Результати роботи програми**

