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 №6

З дисципліни «Методи наукових досліджень»

За темою:

«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи ІВ-91

Гришин О.С.

Номер у списку - 07

ПЕРЕВІРИВ:

асистент

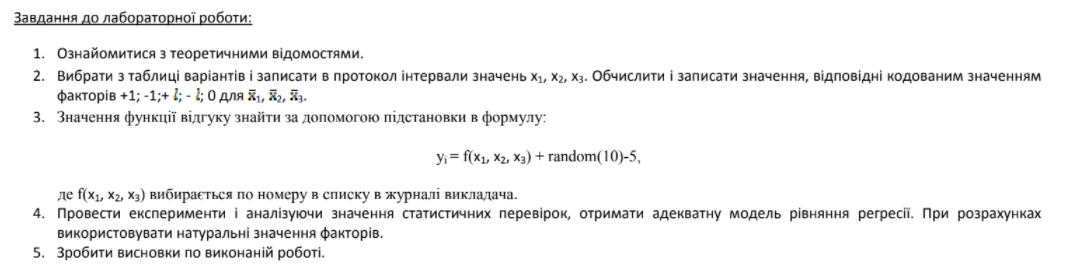
Регіда П.Г.

Київ 2021 р.

**Мета:** Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний

композиційний план.

**Завдання:**

****

**Програмний код**

**Main.py**

*import* sys  
*from* functools *import* partial  
*from* random *import* randrange  
*from* pyDOE2 *import* ccdesign  
*from* Criterium *import* Criteria  
*import* numpy *as* np  
*from* numpy.linalg *import* solve  
*from* prettytable *import* PrettyTable  
*from* scipy.stats *import* f, t  
  
*if len*(sys.argv) == 1:  
 m = 3  
*else*:  
 m = sys.argv[1]  
n = 15  
  
*# VAR 107*x1min = -5  
x1max = 15  
x2min = -15  
x2max = 35  
x3min = 15  
x3max = 30  
  
x01 = (x1max + x1min) / 2  
x02 = (x2max + x2min) / 2  
x03 = (x3max + x3min) / 2  
deltax1 = x1max - x01  
deltax2 = x2max - x02  
deltax3 = x3max - x03  
  
  
*def* function(X1, X2, X3):  
 y = 3.9 + 5.6 \* X1 + 7.9 \* X2 + 7.3 \* X3 + 2.0 \* X1 \* X1 + 0.5 \* X2 \* X2 + 4.2 \* X3 \* X3 + 1.5 \* X1 \* X2 + \  
 0.1 \* X1 \* X3 + 9.9 \* X2 \* X3 + 5.3 \* X1 \* X2 \* X3 + randrange(0, 10) - 5  
 *return* y  
  
  
*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  
  
  
*if* n > 14:  
 no = n - 14  
*else*:  
 no = 1  
xn = ccdesign(3, center=(0, no))  
xn = np.insert(xn, 0, 1, axis=1)  
  
*for* i *in range*(4, 11):  
 xn = np.insert(xn, i, 0, axis=1)  
  
l = 1.215  
  
*for* i *in range*(*len*(xn)):  
 *for* j *in range*(*len*(xn[i])):  
 *if* xn[i][j] < -1 *or* xn[i][j] > 1:  
 *if* xn[i][j] < 0:  
 xn[i][j] = -l  
 *else*:  
 xn[i][j] = l  
  
x\_norm = add\_sq\_nums(xn)  
  
x1 = [x1min, x1min, x1min, x1min, x1max, x1max, x1max, x1max, -1.73 \* deltax1 + x01, 1.73 \* deltax1 + x01, x01, x01,  
 x01, x01, x01]  
x2 = [x2min, x2min, x2max, x2max, x2min, x2min, x2max, x2max, x02, x02, -1.73 \* deltax2 + x02, 1.73 \* deltax2 + x02,  
 x02, x02, x02]  
x3 = [x3min, x3max, x3min, x3max, x3min, x3max, x3min, x3max, x03, x03, x03, x03, -1.73 \* deltax3 + x03,  
 1.73 \* deltax3 + x03, x03]  
x1x2 = [0] \* 15  
x1x3 = [0] \* 15  
x2x3 = [0] \* 15  
x1x2x3 = [0] \* 15  
x1kv = [0] \* 15  
x2kv = [0] \* 15  
x3kv = [0] \* 15  
*for* i *in range*(15):  
 x1x2[i] = x1[i] \* x2[i]  
 x1x3[i] = x1[i] \* x3[i]  
 x2x3[i] = x2[i] \* x3[i]  
 x1x2x3[i] = x1[i] \* x2[i] \* x3[i]  
 x1kv[i] = x1[i] \*\* 2  
 x2kv[i] = x2[i] \*\* 2  
 x3kv[i] = x3[i] \*\* 2  
  
list\_for\_a = *list*(*zip*(x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3, x1kv, x2kv, x3kv))  
  
*for* i *in range*(*len*(list\_for\_a)):  
 list\_for\_a[i] = *list*(list\_for\_a[i])  
 *for* j *in range*(*len*(list\_for\_a[i])):  
 list\_for\_a[i][j] = *round*(list\_for\_a[i][j], 3)  
  
planning\_matrix\_x = PrettyTable()  
planning\_matrix\_x.field\_names = ['X1', 'X2', 'X3', 'X1X2', 'X1X3', 'X2X3', 'X1X2X3', 'X1X1', 'X2X2', 'X3X3']  
*print*("Матриця планування з натуралізованими коефіцієнтами X:")  
planning\_matrix\_x.add\_rows(list\_for\_a)  
*print*(planning\_matrix\_x)  
  
Y = [[function(list\_for\_a[j][0], list\_for\_a[j][1], list\_for\_a[j][2]) *for* i *in range*(m)] *for* j *in range*(15)]  
  
planing\_matrix\_y = PrettyTable()  
planing\_matrix\_y.field\_names = ['Y1', 'Y2', 'Y3']  
*print*("Матриця планування Y:")  
planing\_matrix\_y.add\_rows(Y)  
*print*(planing\_matrix\_y)  
  
Y\_average = []  
*for* i *in range*(*len*(Y)):  
 Y\_average.append(np.mean(Y[i], axis=0))  
*print*("Середні значення відгуку за рядками:")  
*print*(Y\_average)  
  
dispersions = []  
*for* i *in range*(*len*(Y)):  
 a = 0  
 *for* k *in* Y[i]:  
 a += (k - np.mean(Y[i], axis=0)) \*\* 2  
 dispersions.append(a / *len*(Y[i]))  
  
  
*def* find\_known(num):  
 a = 0  
 *for* j *in range*(15):  
 a += Y\_average[j] \* list\_for\_a[j][num - 1] / 15  
 *return* a  
  
  
*def* get\_exp\_val(k):  
 *return* beta[0] + beta[1] \* list\_for\_a[k][0] + beta[2] \* list\_for\_a[k][1] + beta[3] \* list\_for\_a[k][2] + \  
 beta[4] \* list\_for\_a[k][3] + beta[5] \* list\_for\_a[k][4] + beta[6] \* list\_for\_a[k][5] + beta[7] \* \  
 list\_for\_a[k][6] + beta[8] \* list\_for\_a[k][7] + beta[9] \* list\_for\_a[k][8] + beta[10] \* list\_for\_a[k][9]  
  
  
*def* a(first, second):  
 a = 0  
 *for* j *in range*(15):  
 a += list\_for\_a[j][first - 1] \* list\_for\_a[j][second - 1] / 15  
 *return* a  
  
  
my = *sum*(Y\_average) / 15  
mx = []  
*for* i *in range*(10):  
 number\_lst = []  
 *for* j *in range*(15):  
 number\_lst.append(list\_for\_a[j][i])  
 mx.append(*sum*(number\_lst) / *len*(number\_lst))  
  
det1 = [  
 [1, \*mx],  
 \*[[mx[row - 1], \*[a(row, col) *for* col *in range*(1, 11)]] *for* row *in range*(1, 11)]  
]  
  
det2 = [my, \*[find\_known(num) *for* num *in range*(1, 11)]]  
  
beta = solve(det1, det2)  
  
*print*("\nОтримане рівняння регресії:")  
*print*("{} + {} \* X1 + {} \* X2 + {} \* X3 + {} \* Х1X2 + {} \* Х1X3 + {} \* Х2X3+ {} \* Х1Х2X3 + {} \* X11^2 + {} \* X22^2 + "  
 "{} \* X33^2 = ŷ "  
 .format(\*beta))  
  
*print*("Експериментальні значення:")  
y\_i = [get\_exp\_val(k) *for* k *in range*(15)]  
*print*(y\_i)  
  
criteria = Criteria(list\_for\_a, Y, n, m)  
  
  
*print*("\n\n------------------------------- Перевірка за критерієм Кохрена -------------------------------")  
G\_kr = criteria.cohren()  
Gp = criteria.criteria\_cochrana(Y\_average)  
*print*(f'Gp = {Gp}')  
*if* Gp < G\_kr:  
 *print*(f'З ймовірністю {1 - criteria.q} дисперсії однорідні.')  
*else*:  
 *print*("Необхідно збільшити кількість дослідів")  
 m += 1  
 new\_com = [sys.argv[0] + f" {m}"]  
 *print*([sys.executable] + new\_com)  
  
*print*("\n------------------ Перевірка значущості коефіцієнтів за критерієм Стьюдента ------------------")  
student = partial(t.ppf, q=1 - criteria.q)  
t\_student = student(df=criteria.f3)  
ts = criteria.criteria\_studenta(xn[:, 1:], Y\_average)  
res = [t *for* t *in* ts *if* t > t\_student]  
final\_k = [beta[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* beta *if* i *not in* final\_k]))  
d = *len*(final\_k)  
y\_st = []  
*for* i *in range*(15):  
 y\_st.append(res[0] + res[1] \* x1[i] + res[2] \* x2[i] + res[3] \* x3[i] + res[4] \* x1x2[i] + res[5] \*  
 x1x3[i] + res[6] \* x2x3[i] + res[7] \* x1x2x3[i] + res[8] \* x1kv[i] + res[9] \*  
 x2kv[i] + res[10] \* x3kv[i])  
  
*print*("\n\n------------------------- Перевірка адекватності за критерієм Фішера -------------------------")  
F\_p = criteria.criteria\_fishera(Y\_average, y\_st, d)  
f4 = n - d  
fisher = partial(f.ppf, q=1-criteria.q)  
f\_t = fisher(dfn=f4, dfd=criteria.f3)  
  
*print*('\nПеревірка адекватності за критерієм Фішера')  
*print*('Fp =', F\_p)  
*print*('F\_t =', f\_t)  
*if len*(final\_k) == 2:  
 *print*('Математична модель не адекватна експериментальним даним')  
*else*:  
 *print*('Математична модель адекватна експериментальним даним')

**Criteria.py**

*from* scipy.stats *import* f  
  
  
*class* Criteria:  
 *def \_\_init\_\_*(*self*, x, y, n, m):  
 *self*.x = x  
 *self*.y = y  
  
 *self*.n = n  
 *self*.m = m  
 *self*.f1 = *self*.m - 1  
 *self*.f2 = *self*.n  
 *self*.q = 0.05  
 *self*.q1 = *self*.q / *self*.f1  
  
 *def* s\_kv(*self*, y\_aver):  
 res = []  
 *for* i *in range*(*self*.n):  
 s = *sum*([(y\_aver[i] - *self*.y[i][j]) \*\* 2 *for* j *in range*(*self*.m)]) / *self*.m  
 res.append(*round*(s, 3))  
 *return* res  
  
 *def* criteria\_cochrana(*self*, y\_aver):  
 S\_kv = *self*.s\_kv(y\_aver)  
 Gp = *max*(S\_kv) / *sum*(S\_kv)  
 *print*('\nПеревірка за критерієм Кохрена')  
 *return* Gp  
  
 *def* cohren(*self*):  
 fisher\_value = f.ppf(q=1 - *self*.q1, dfn=*self*.f2, dfd=(*self*.f1 - 1) \* *self*.f2)  
 *return* fisher\_value / (fisher\_value + *self*.f1 - 1)  
  
 *def* bs(*self*, x, y\_aver):  
 res = [*sum*(y\_aver) / *self*.n]  
  
 *for* i *in range*(*len*(x[0])):  
 b = *sum*(j[0] \* j[1] *for* j *in zip*(x[:, i], y\_aver)) / *self*.n  
 res.append(b)  
  
 *return* res  
  
 *def* criteria\_studenta(*self*, x, y\_aver):  
 S\_kv = *self*.s\_kv(y\_aver)  
 s\_kv\_aver = *sum*(S\_kv) / *self*.n  
  
 s\_Bs = (s\_kv\_aver / *self*.n / *self*.m) \*\* 0.5  
 Bs = *self*.bs(x, y\_aver)  
 ts = [*round*(*abs*(B) / s\_Bs, 3) *for* B *in* Bs]  
  
 *return* ts  
  
 *def* criteria\_fishera(*self*, y\_aver, y\_new, d):  
 S\_ad = *self*.m / (*self*.n - d) \* *sum*([(y\_new[i] - y\_aver[i]) \*\* 2 *for* i *in range*(*len*(*self*.y))])  
 S\_kv = *self*.s\_kv(y\_aver)  
 S\_kv\_aver = *sum*(S\_kv) / *self*.n  
  
 *return* S\_ad / S\_kv\_aver

**Результати роботи програми**

