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 З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи ІВ-91

Гришин О.С.

Номер у списку - 07

ПЕРЕВІРИВ:

асистент

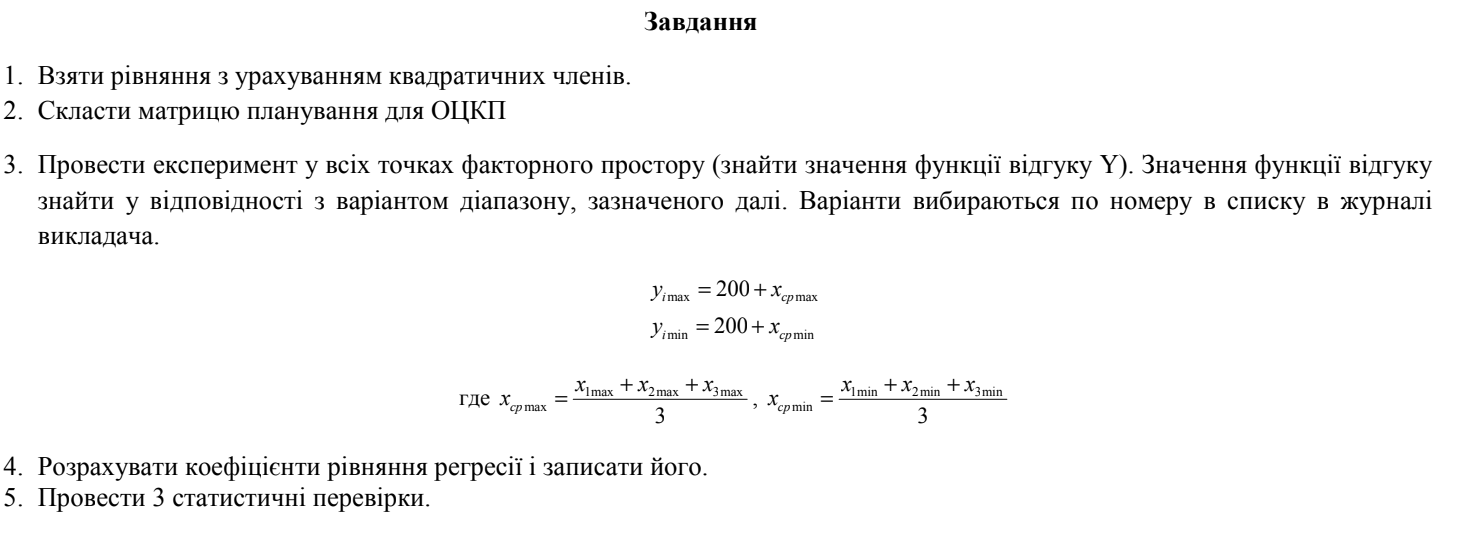
Регіда П.Г.

Київ 2021 р.

**Мета:** Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)

**Завдання:**

****

****

**Програмний код**

**Main.pt**

*from* lab05.src.Experiment *import* Experiment  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 experiment = Experiment((-9, 7),(-4, 7),(-10, 5))  
  
 experiment.run\_experiment(15, 3)

**Experiment.py**

*import* random  
*import* sklearn.linear\_model *as* lm  
*from* scipy.stats *import* f, t  
*from* functools *import* partial  
*from* pyDOE2 *import* \*  
*from* numpy *import* average  
*from* lab05.src.Criterium *import* Criteria  
*import* pandas *as* pd  
*from* tabulate *import* tabulate  
  
  
*class* Experiment:  
  
 *def \_\_init\_\_*(*self*, x1, x2, x3):  
  
 *self*.n, *self*.m = 15, 6  
  
 *self*.x, *self*.y, *self*.x\_norm = *None*, *None*, *None  
 self*.y\_average = *None  
  
 self*.b = *None  
  
 self*.x\_range = (x1, x2, x3)  
  
 *self*.x\_aver\_max = average([x[1] *for* x *in self*.x\_range])  
 *self*.x\_aver\_min = average([x[0] *for* x *in self*.x\_range])  
  
 *self*.y\_max = 200 + *int*(*self*.x\_aver\_max)  
 *self*.y\_min = 200 + *int*(*self*.x\_aver\_min)  
  
 @staticmethod  
 *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  
  
 *def* count\_y\_average(*self*):  
 *self*.y\_average = [*round*(*sum*(i) / *len*(i), 3) *for* i *in self*.y]  
  
 *def* regression(*self*, x, b):  
 *return sum*([x[i] \* b[i] *for* i *in range*(*len*(x))])  
  
 *def* count\_b\_coeficient(*self*):  
 *# X5, y5\_aver* skm = lm.LinearRegression(fit\_intercept=*False*)  
 skm.fit(*self*.x, *self*.y\_average)  
 *self*.b = skm.coef\_  
  
 *print*('\nКоефіцієнти рівняння регресії:')  
 *self*.b = [*round*(i, 3) *for* i *in self*.b]  
 *print*("y = {} +{}\*x1 +{}\*x2 +{}\*x3 + {}\*x1\*x2 + {}\*x1\*x3 + {}\*x2\*x3 + b{}\*x1\*x2\*x3 + {}x1^2 + {}x2^2 + {}x3^2\n".format(\**self*.b))  
 *print*('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(*self*.x, *self*.b))  
  
 *def* run\_check(*self*):  
 criteria = Criteria(*self*.x, *self*.y, *self*.n, *self*.m)  
  
 *print*('\n\tПеревірка рівняння:')  
 f1 = *self*.m - 1  
 f2 = *self*.n  
 f3 = f1 \* f2  
 q = 0.05  
  
 student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
  
 G\_kr = criteria.cohren()  
  
 y\_aver = [*round*(*sum*(i) / *len*(i), 3) *for* i *in self*.y]  
 *print*('\nСереднє значення y:', y\_aver)  
  
 disp = criteria.s\_kv(y\_aver)  
 *print*('Дисперсія y:', disp)  
  
 Gp = criteria.criteria\_cochrana(y\_aver)  
 *print*(f'Gp = {Gp}')  
 *if* Gp < G\_kr:  
 *print*(f'З ймовірністю {1 - q} дисперсії однорідні.')  
 *else*:  
 *print*("Необхідно збільшити кількість дослідів")  
 *self*.m += 1  
 new\_exp = Experiment(\**self*.x\_range)  
 new\_exp.run\_experiment(*self*.n, *self*.m)  
  
 ts = criteria.criteria\_studenta(*self*.x\_norm[:, 1:], y\_aver)  
 *print*('\nКритерій Стьюдента:\n', ts)  
 res = [t *for* t *in* ts *if* t > t\_student]  
 final\_k = [*self*.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 self*.b *if* i *not in* final\_k]))  
  
 y\_new = []  
 *for* j *in range*(*self*.n):  
 y\_new.append(*self*.regression([*self*.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 >= *self*.n:  
 *print*('\nF4 <= 0')  
  
 f4 = *self*.n - d  
  
 F\_p = criteria.criteria\_fishera(y\_aver, y\_new, 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* fill\_y(*self*):  
 *self*.y = np.zeros(shape=(*self*.n, *self*.m))  
 *for* i *in range*(*self*.n):  
 *for* j *in range*(*self*.m):  
 *self*.y[i][j] = random.randint(*self*.y\_min, *self*.y\_max)  
  
 *def* make\_matrix(*self*):  
 *self*.fill\_y()  
 *if self*.n > 14:  
 no = *self*.n - 14  
 *else*:  
 no = 1  
  
 *self*.x\_norm = ccdesign(3, center=(0, no))  
 *self*.x\_norm = np.insert(*self*.x\_norm, 0, 1, axis=1)  
  
 *for* i *in range*(4, 11):  
 *self*.x\_norm = np.insert(*self*.x\_norm, i, 0, axis=1)  
  
 l = 1.215  
  
 *for* i *in range*(*len*(*self*.x\_norm)):  
 *for* j *in range*(*len*(*self*.x\_norm[i])):  
 *if self*.x\_norm[i][j] < -1 *or self*.x\_norm[i][j] > 1:  
 *if self*.x\_norm[i][j] < 0:  
 *self*.x\_norm[i][j] = -l  
 *else*:  
 *self*.x\_norm[i][j] = l  
  
 *self*.x\_norm = *self*.add\_sq\_nums(*self*.x\_norm)  
  
 *self*.x = np.ones(shape=(*len*(*self*.x\_norm), *len*(*self*.x\_norm[0])), dtype=np.int64)  
 *for* i *in range*(8):  
 *for* j *in range*(1, 4):  
 *if self*.x\_norm[i][j] == -1:  
 *self*.x[i][j] = *self*.x\_range[j - 1][0]  
 *else*:  
 *self*.x[i][j] = *self*.x\_range[j - 1][1]  
  
 *for* i *in range*(8, *len*(*self*.x)):  
 *for* j *in range*(1, 3):  
 *self*.x[i][j] = (*self*.x\_range[j - 1][0] + *self*.x\_range[j - 1][1]) / 2  
  
 dx = [*self*.x\_range[i][1] - (*self*.x\_range[i][0] + *self*.x\_range[i][1]) / 2 *for* i *in range*(3)]  
  
 *self*.x[8][1] = l \* dx[0] + *self*.x[9][1]  
 *self*.x[9][1] = -l \* dx[0] + *self*.x[9][1]  
 *self*.x[10][2] = l \* dx[1] + *self*.x[9][2]  
 *self*.x[11][2] = -l \* dx[1] + *self*.x[9][2]  
 *self*.x[12][3] = l \* dx[2] + *self*.x[9][3]  
 *self*.x[13][3] = -l \* dx[2] + *self*.x[9][3]  
 *self*.x = *self*.add\_sq\_nums(*self*.x)  
  
 show\_arr = pd.DataFrame(*self*.x)  
 *print*('\nX:\n', tabulate(show\_arr, headers='keys', tablefmt='psql'))  
  
 show\_arr = pd.DataFrame(*self*.x\_norm)  
 *print*('\nX нормоване:\n', tabulate(show\_arr.round(0), headers='keys', tablefmt='psql'))  
  
 show\_arr = pd.DataFrame(*self*.y)  
 *print*('\nY:\n', tabulate(show\_arr, headers='keys', tablefmt='psql'))  
  
 *def* run\_experiment(*self*, n=*None*, m=*None*):  
 *if* n *is None and* m *is None*:  
 *pass  
 else*:  
 *self*.n = n  
 *self*.m = m  
  
 *self*.make\_matrix()  
 *self*.count\_y\_average()  
  
 *self*.count\_b\_coeficient()  
 *self*.run\_check()

**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**

