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 №4

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:

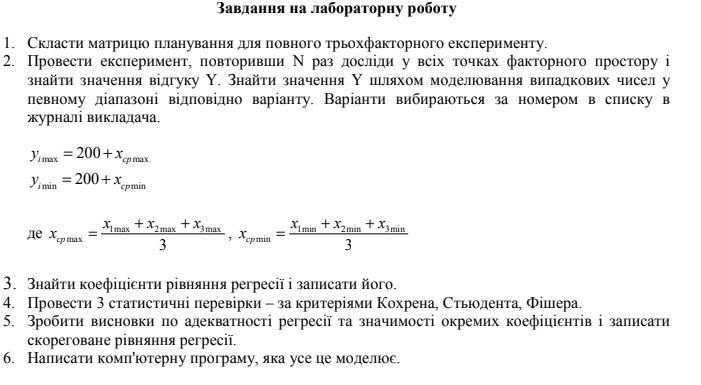
асистент

Регіда П.Г.

Київ 2021 р.

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

**Завдання:**

****

****

**Програмний код**

**Main.pt**

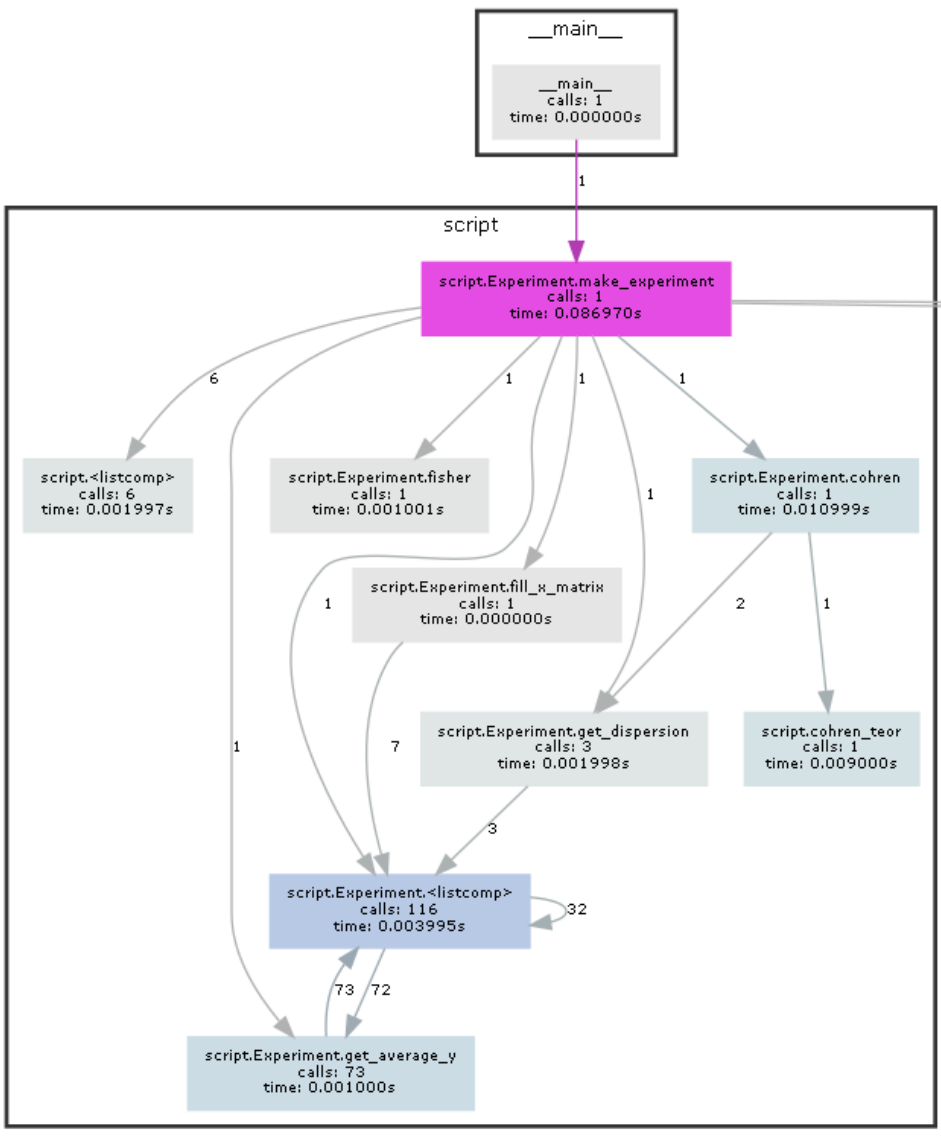
*import* logging  
*import* os  
*import* sys  
*from* pycallgraph *import* PyCallGraph  
*from* pycallgraph.output *import* GraphvizOutput  
  
logging.basicConfig(filename='app.log', filemode='w', format='%(levelname)s - %(message)s')  
  
*try*:  
 *from* script *import* \*  
*except ImportError*:  
 logging.exception('Import error')  
 sys.exit()  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 e = Experiment((10, 40), (25, 45), (40, 45))  
  
 *try*:  
 *with* PyCallGraph(output=GraphvizOutput()):  
 e.make\_experiment()  
 *except Exception as* e:  
 logging.exception(f'{e} in module '  
 f'- {os.path.split(sys.exc\_info()[-1].tb\_frame.f\_code.co\_filename)[1]} '  
 f'- at line {sys.exc\_info()[-1].tb\_lineno}', exc\_info=*False*)  
 sys.exit()

**Script.py**

*import* math  
*import* numpy *as* np  
*from* numpy *import* average, transpose  
*from* numpy.linalg *import* solve  
*from* prettytable *import* PrettyTable  
*from* scipy.stats *import* f  
*from* scipy.stats *import* t *as* t\_criterium  
*from* functools *import* partial  
*from* random *import* randint  
  
  
*class* Experiment:  
 m, N, d = 3, 8, 8  
  
 x0\_factor = [1, 1, 1, 1, 1, 1, 1, 1]  
 x1\_factor = [-1, -1, 1, 1, -1, -1, 1, 1]  
 x2\_factor = [-1, 1, -1, 1, -1, 1, -1, 1]  
 x3\_factor = [-1, 1, 1, -1, 1, -1, -1, 1]  
 x1x2\_factor = [a \* b *for* a, b *in zip*(x1\_factor, x2\_factor)]  
 x1x3\_factor = [a \* b *for* a, b *in zip*(x1\_factor, x3\_factor)]  
 x2x3\_factor = [a \* b *for* a, b *in zip*(x2\_factor, x3\_factor)]  
 x1x2x3\_factor = [a \* b \* c *for* a, b, c *in zip*(x1\_factor, x2\_factor, x3\_factor)]  
  
 x1\_list = []  
 x2\_list = []  
 x3\_list = []  
 x1x2\_list = []  
 x1x3\_list = []  
 x2x3\_list = []  
 x1x2x3\_list = []  
 x\_main\_list = [x0\_factor, x1\_list, x2\_list, x3\_list, x1x2\_list, x1x3\_list, x2x3\_list, x1x2x3\_list]  
 x\_factor\_list = [x0\_factor, x1\_factor, x2\_factor, x3\_factor, x1x2\_factor, x1x3\_factor, x2x3\_factor, x1x2x3\_factor]  
  
 list\_bi = []  
  
 *def \_\_init\_\_*(*self*, x1, x2, x3):  
  
 *self*.F1 = *self*.m - 1  
 *self*.F2 = *self*.N  
 *self*.F3 = *self*.F1 \* *self*.F2  
 *self*.F4 = *self*.N - *self*.d  
  
 *self*.x1 = x1  
 *self*.x2 = x2  
 *self*.x3 = x3  
 *self*.x\_tuple = (*self*.x1, *self*.x2, *self*.x3)  
 *# print(self.x\_tuple)  
 self*.x\_max\_average = average([i[1] *for* i *in self*.x\_tuple])  
 *# print(self.x\_max\_average)  
 self*.x\_min\_average = average([i[0] *for* i *in self*.x\_tuple])  
 *# print(self.x\_min\_average)  
 self*.y\_max = *int*(200 + *self*.x\_max\_average)  
 *self*.y\_min = *int*(200 + *self*.x\_min\_average)  
 *self*.y\_min\_max = [*self*.y\_min, *self*.y\_max]  
 *# print(self.y\_max, self.y\_min)  
 self*.mat\_Y = [[randint(*self*.y\_min\_max[0], *self*.y\_min\_max[1]) *for* \_ *in range*(*self*.m)] *for* \_ *in range*(*self*.N)]  
 *# pprint(self.mat\_Y)  
  
 def* get\_average\_y(*self*):  
 *"""середні значення функцій"""  
 return* [*round*(*sum*(*self*.mat\_Y[k1]) / *self*.m, 3) *for* k1 *in range*(*self*.N)]  
  
 *def* get\_dispersion(*self*):  
 *return* [*round*(*sum*([((k1 - *self*.get\_average\_y()[j]) \*\* 2) *for* k1 *in self*.mat\_Y[j]]) / *self*.m, 3) *for* j *in  
 range*(*self*.N)]  
  
 *def* fill\_x\_matrix(*self*):  
 *"""Заповнює матрицю експерименту"""* [*self*.x1\_list.append(*self*.x1[0 *if* i == -1 *else* 1]) *for* i *in self*.x1\_factor]  
 [*self*.x2\_list.append(*self*.x2[0 *if* i == -1 *else* 1]) *for* i *in self*.x2\_factor]  
 [*self*.x3\_list.append(*self*.x3[0 *if* i == -1 *else* 1]) *for* i *in self*.x3\_factor]  
 [*self*.x1x2\_list.append(a \* b) *for* a, b *in zip*(*self*.x1\_list, *self*.x2\_list)]  
 [*self*.x1x3\_list.append(a \* b) *for* a, b *in zip*(*self*.x1\_list, *self*.x3\_list)]  
 [*self*.x2x3\_list.append(a \* b) *for* a, b *in zip*(*self*.x2\_list, *self*.x3\_list)]  
 [*self*.x1x2x3\_list.append(a \* b \* c) *for* a, b, c *in zip*(*self*.x1\_list, *self*.x2\_list, *self*.x3\_list)]  
  
 *def* cohren(*self*):  
  
 *def* cohren\_teor(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)  
  
 Gp = *max*(*self*.get\_dispersion()) / *sum*(*self*.get\_dispersion())  
 Gt = cohren\_teor(*self*.F1, *self*.F2)  
 *print*("\nGp = ", Gp, " Gt = ", Gt)  
 *return* Gp < Gt  
  
 *def* fisher(*self*):  
 fisher\_teor = partial(f.ppf, q=1 - 0.05)  
 Ft = fisher\_teor(dfn=*self*.F4, dfd=*self*.F3)  
 *return* Ft  
  
 *def* make\_experiment(*self*):  
 *self*.fill\_x\_matrix()  
  
 dispersion = *self*.get\_dispersion()  
 sum\_dispersion = *sum*(dispersion)  
 y\_average = *self*.get\_average\_y()  
  
 column\_names1 = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1", "Y2", "Y3", "Y", "S^2"]  
 trans\_y\_mat = transpose(*self*.mat\_Y).tolist()  
  
 """При розрахунку ко-ів використовуєемо як натуральні так і нормовані значення"""  
 list\_for\_solve\_a = *list*(*zip*(\**self*.x\_main\_list))  
 list\_for\_solve\_b = *self*.x\_factor\_list  
  
 *for* k *in range*(*self*.N):  
 S = 0  
 *for* i *in range*(*self*.N):  
 S += (list\_for\_solve\_b[k][i] \* y\_average[i]) / *self*.N  
 *self*.list\_bi.append(*round*(S, 5))  
  
 pt = PrettyTable()  
 cols = *self*.x\_factor\_list  
 [cols.extend(ls) *for* ls *in* [trans\_y\_mat, [y\_average], [dispersion]]]  
 [pt.add\_column(column\_names1[coll\_id], cols[coll\_id]) *for* coll\_id *in range*(13)]  
 *print*(pt, "\n")  
 *print*('Рівняння регресії з коефіцієнтами від нормованих значень факторів')  
 *print*("y = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3 \n".format(\**self*.list\_bi))  
  
 pt = PrettyTable()  
 cols = *self*.x\_main\_list  
 [cols.extend(ls) *for* ls *in* [trans\_y\_mat, [y\_average], [dispersion]]]  
 [pt.add\_column(column\_names1[coll\_id], cols[coll\_id]) *for* coll\_id *in range*(13)]  
 *print*(pt, "\n")  
  
 """solve з бібліотеки numpy вирішує систему рівнянь відносно невідомих к-тів  
 рівняння регресії при використанні натуральних значень"""  
 list\_ai = [*round*(i, 5) *for* i *in* solve(list\_for\_solve\_a, y\_average)]  
 *print*('Рівняння регресії з коефіцієнтами від натуральних значень факторів')  
 *print*("y = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3".format(\*list\_ai))  
  
 *if self*.cohren():  
 *print*("Дисперсія однорідна!\n")  
 Dispersion\_B = sum\_dispersion / *self*.N  
 Dispersion\_beta = Dispersion\_B / (*self*.m \* *self*.N)  
 S\_beta = math.sqrt(*abs*(Dispersion\_beta))  
 beta\_list = np.zeros(8).tolist()  
 *for* i *in range*(*self*.N):  
 beta\_list[0] += (y\_average[i] \* *self*.x0\_factor[i]) / *self*.N  
 beta\_list[1] += (y\_average[i] \* *self*.x1\_factor[i]) / *self*.N  
 beta\_list[2] += (y\_average[i] \* *self*.x2\_factor[i]) / *self*.N  
 beta\_list[3] += (y\_average[i] \* *self*.x3\_factor[i]) / *self*.N  
 beta\_list[4] += (y\_average[i] \* *self*.x1x2\_factor[i]) / *self*.N  
 beta\_list[5] += (y\_average[i] \* *self*.x1x3\_factor[i]) / *self*.N  
 beta\_list[6] += (y\_average[i] \* *self*.x2x3\_factor[i]) / *self*.N  
 beta\_list[7] += (y\_average[i] \* *self*.x1x2x3\_factor[i]) / *self*.N  
 t\_list = [*abs*(beta\_list[i]) / S\_beta *for* i *in range*(0, *self*.N)]  
  
 *for* i, j *in enumerate*(t\_list):  
 *if* j < t\_criterium.ppf(q=0.975, df=*self*.F3):  
 *print*(f'незначний {beta\_list[i]}')  
 beta\_list[i] = 0  
 *self*.d -= 1  
 *print*()  
 *print*("y = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3".format(\*beta\_list))  
  
 Y\_counted = [*sum*([beta\_list[0], \*[beta\_list[i] \* *self*.x\_main\_list[1:][j][i] *for* i *in range*(*self*.N)]])  
 *for* j *in range*(*self*.N)]  
 Dispersion\_ad = 0  
 *for* i *in range*(*len*(Y\_counted)):  
 Dispersion\_ad += ((Y\_counted[i] - y\_average[i]) \*\* 2) \* *self*.m / (*self*.N - *self*.d)  
 Fp = Dispersion\_ad / Dispersion\_beta  
 Ft = *self*.fisher()  
  
 *if* Ft > Fp:  
 *print*("Рівняння регресії адекватне!")  
 *else*:  
 *print*("Рівняння регресії неадекватне.")

**Результати роботи програми**

Метрики (повна версія із замірами залучення інших бібліотек можна знайти в репозиторії)



Результати розрахунків

