Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування

Лабораторна робота №6

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

Виконав:

студент групи ІО-82

Фещенко Кирил

Залікова книжка № 8224

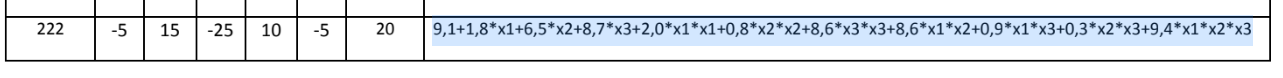
Номер у списку групи 22

Перевірив ас. Регіда П. Г.

Київ 2020 р

**Завдання**





**Код**

import random , numpy  
from scipy.stats import t , f  
  
  
def kohren(mat\_y , m , n):  
 s = [ ]  
 for i in range ( n ):  
 ks = 0  
 for j in range ( m ):  
 ks += (mat\_y[ i ][ -1 ] - mat\_y[ i ][ j ]) \*\* 2  
 s.append ( ks / m )  
 gp = max ( s ) / sum ( s )  
 fisher = table\_fisher ( 0.95 , n , m , 1 )  
 gt = fisher / (fisher + (m - 1) - 2)  
 return gp < gt  
  
  
def geny(n , m):  
 def f(x1 , x2 , x3):  
 f = 9.1 + 1.8 \* x1 + 6.5 \* x2 + 8.7 \* x3  
 f += 2.0 \* x1 \* x1 + 0.8 \* x2 \* x2 + 8.6 \* x3 \* x3  
 f += 8.6 \* x1 \* x2 + 0.9 \* x1 \* x3 + 0.3 \* x2 \* x3 + 9.4 \* x1 \* x2 \* x3  
 return f  
  
 mat\_y = [ [ round ( f ( \*xnat[ i ] ) + random.randint ( 0 , 10 ) - 5 , 2 ) for j in range ( m ) ] for i in  
 range ( n ) ]  
 for elem in mat\_y:  
 elem.append ( sum ( elem ) / len ( elem ) )  
 return mat\_y  
  
  
# give combinations of xnat elements or others  
def cmb(arr):  
 return [ 1 , \*arr ,  
 round ( arr[ 0 ] \* arr[ 1 ] , 2 ) ,  
 round ( arr[ 0 ] \* arr[ 2 ] , 2 ) ,  
 round ( arr[ 1 ] \* arr[ 2 ] , 2 ) ,  
 round ( arr[ 0 ] \* arr[ 1 ] \* arr[ 2 ] , 2 ) ,  
 round ( arr[ 0 ] \* arr[ 0 ] , 2 ) ,  
 round ( arr[ 1 ] \* arr[ 1 ] , 2 ) ,  
 round ( arr[ 2 ] \* arr[ 2 ] , 2 ) ]  
  
  
# calculate b koefficients  
def get\_b(lmaty):  
 a00 = [ [ ] ,  
 [ xnatmod[ 0 ] ] , [ xnatmod[ 1 ] ] , [ xnatmod[ 2 ] ] ,  
 [ xnatmod[ 0 ] , xnatmod[ 1 ] ] ,  
 [ xnatmod[ 0 ] , xnatmod[ 2 ] ] ,  
 [ xnatmod[ 1 ] , xnatmod[ 2 ] ] ,  
 [ xnatmod[ 0 ] , xnatmod[ 1 ] , xnatmod[ 2 ] ] ,  
 [ xnatmod[ 0 ] , xnatmod[ 0 ] ] ,  
 [ xnatmod[ 1 ] , xnatmod[ 1 ] ] ,  
 [ xnatmod[ 2 ] , xnatmod[ 2 ] ] ]  
  
 def calcxi(n , listx):  
 sumxi = 0  
 for i in range ( n ):  
 lsumxi = 1  
 for j in range ( len ( listx ) ):  
 lsumxi \*= listx[ j ][ i ]  
 sumxi += lsumxi  
 return sumxi  
  
 a0 = [ 15 ]  
 for i in range ( 10 ):  
 a0.append ( calcxi ( n , a00[ i + 1 ] ) )  
  
 a1 = [ calcxi ( n , a00[ i ] + a00[ 1 ] ) for i in range ( len ( a00 ) ) ]  
 a2 = [ calcxi ( n , a00[ i ] + a00[ 2 ] ) for i in range ( len ( a00 ) ) ]  
 a3 = [ calcxi ( n , a00[ i ] + a00[ 3 ] ) for i in range ( len ( a00 ) ) ]  
 a4 = [ calcxi ( n , a00[ i ] + a00[ 4 ] ) for i in range ( len ( a00 ) ) ]  
 a5 = [ calcxi ( n , a00[ i ] + a00[ 5 ] ) for i in range ( len ( a00 ) ) ]  
 a6 = [ calcxi ( n , a00[ i ] + a00[ 6 ] ) for i in range ( len ( a00 ) ) ]  
 a7 = [ calcxi ( n , a00[ i ] + a00[ 7 ] ) for i in range ( len ( a00 ) ) ]  
 a8 = [ calcxi ( n , a00[ i ] + a00[ 8 ] ) for i in range ( len ( a00 ) ) ]  
 a9 = [ calcxi ( n , a00[ i ] + a00[ 9 ] ) for i in range ( len ( a00 ) ) ]  
 a10 = [ calcxi ( n , a00[ i ] + a00[ 10 ] ) for i in range ( len ( a00 ) ) ]  
  
 a = numpy.array (  
 [ [ \*a0 ] , [ \*a1 ] , [ \*a2 ] , [ \*a3 ] , [ \*a4 ] , [ \*a5 ] , [ \*a6 ] , [ \*a7 ] , [ \*a8 ] , [ \*a9 ] ,  
 [ \*a10 ] ] )  
 c0 = [ calcxi ( n , [ lmaty ] ) ]  
 for i in range ( len ( a00 ) - 1 ):  
 c0.append ( calcxi ( n , a00[ i + 1 ] + [ lmaty ] ) )  
 c = numpy.array ( c0 )  
 b = numpy.linalg.solve ( a , c )  
  
 return b  
  
  
def table\_student(prob , n , m):  
 x\_vec = [ i \* 0.0001 for i in range ( int ( 5 / 0.0001 ) ) ]  
 par = 0.5 + prob / 0.1 \* 0.05  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs ( t.cdf ( i , f3 ) - par ) < 0.000005:  
 return i  
  
  
def table\_fisher(prob , n , m , d):  
 x\_vec = [ i \* 0.001 for i in range ( int ( 10 / 0.001 ) ) ]  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs ( f.cdf ( i , n - d , f3 ) - prob ) < 0.0001:  
 return i  
  
  
def student(n , m , mat\_y):  
 disp = [ ]  
 for i in mat\_y:  
 s = 0  
 for k in range ( m ):  
 s += (i[ -1 ] - i[ k ]) \*\* 2  
 disp.append ( s / m )  
  
 sbt = (sum ( disp ) / n / n / m) \*\* (0.5)  
  
 bs = [ ]  
 for i in range ( 11 ):  
 ar = [ ]  
 for j in range ( len ( mat\_y ) ):  
 ar.append ( mat\_y[ j ][ -1 ] \* cmb ( xnorm[ j ] )[ i ] / n )  
 bs.append ( sum ( ar ) )  
  
 t = [ (bs[ i ] / sbt) for i in range ( 11 ) ]  
 tt = table\_student ( 0.95 , n , m )  
 st = [ i > tt for i in t ]  
 return st  
  
  
def fisher(b\_0 , x\_mod , n , m , d , mat\_y):  
 if d == n:  
 return True  
 disp = [ ]  
 for i in mat\_y:  
 s = 0  
 for k in range ( m ):  
 s += (i[ -1 ] - i[ k ]) \*\* 2  
 disp.append ( s / m )  
  
 sad = sum ( [ (sum ( [ cmb ( xnat[ i ] )[ j ] \* b\_0[ j ] for j in range ( 11 ) ] ) - mat\_y[ i ][ -1 ]) \*\* 2 for i in  
 range ( n ) ] )  
 sad = sad \* m / (n - d)  
 fp = sad / sum ( disp ) / n  
 ft = table\_fisher ( 0.95 , n , m , d )  
 return fp < ft  
  
  
def all\_print():  
 titles\_x = [ "№" , "X1" , "X2" , "X3" , "X1\*X2" , "X1\*X3" , "X2\*X3" , "X1\*X2\*X3" , "X1^2" , "X2^2" , "X3^2" ]  
 # cycles for table with normal  
 # title, combinations of Xnorm  
 for j in range ( 11 ):  
 s = ""  
 if j == 0:  
 s = " {:^2s} "  
 if j >= 1 and j < 4:  
 s = "{:^8s} "  
 if j >= 4 and j < 7:  
 s = "{:^10s} "  
 if j == 7:  
 s = "{:^11s} "  
 if j > 7 and j < 11:  
 s = "{:^10s} "  
 print ( s.format ( titles\_x[ j ] ) , end="" )  
  
 print ()  
 # aggregate for table, combinationns of Xnorm  
 for i in range ( n ):  
 print ( " {:2d} ".format ( i ) , end="" )  
 for j in range ( 1 , 11 ):  
 x = cmb ( xnorm[ i ] )[ j ]  
 s = ""  
 if j >= 1 and j < 4:  
 s = "{:^ 8} "  
 if j >= 4 and j < 7:  
 s = "{:^ 10} "  
 if j == 7:  
 s = "{:^ 11} "  
 if j > 7 and j < 11:  
 s = "{:^ 10} "  
 # using construction similar to ternar operator for printing 0, instead of 0.0  
 print ( s.format ( x ) , end="" )  
 print ()  
 print ( "\n" )  
  
 # cycle for pretty printing title of table with normal parameters  
 for j in range ( 11 ):  
 s = ""  
 if j == 0:  
 s = "{:^4s}" # for №  
 if j >= 1 and j < 4:  
 s = "{:^7s}" # for X0  
 if j >= 4 and j < 7:  
 s = "{:^8s}" # for X + num  
 if j == 7:  
 s = "{:^11s}" # for X\*X\*X  
 if j > 7 and j < 11:  
 s = "{:^9s}" # for X\*X, with different combinations  
 print ( s.format ( titles\_x[ j ] ) , end="" ) # taking all titles from list  
  
 # this cycle is used for printing Yi in title of table  
 for i in range ( m ):  
 print ( "{:^11s}".format ( "Yi" + str ( i + 1 ) ) , end="" )  
 # printing Y middle, Y experimental and dispersion  
 print ( "{:^11s}{:^11s}".format ( "Ys" , "Ye" ) , end="" )  
  
 print ()  
 # fill table with data  
 for i in range ( n ):  
 print ( "{:^3d}".format ( i ) , end="" )  
 for j in range ( 1 , 11 ):  
 s = ""  
 if j >= 1 and j < 4:  
 s = "{:^ 7}"  
 if j >= 4 and j < 7:  
 s = "{:^ 8}"  
 if j == 7:  
 s = "{:^ 12}"  
 if j > 7 and j < 11:  
 s = "{:^ 9}"  
 print ( s.format ( cmb ( xnat[ i ] )[ j ] ) , end="" )  
  
 for j in maty[ i ][ :-1 ]:  
 print ( "{:^ 11}".format ( j ) , end="" )  
 print ( "{:^ 11}{:^ 11}"  
 .format ( maty[ i ][ -1 ] ,  
 round ( sum ( [ cmb ( xnat[ i ] )[ j ] \* b0[ j ] \* dmas[ j ] for j in range ( 11 ) ] ) ) ,  
 2 ) , end="" )  
  
 print ()  
  
 print ( "\nФункція відгуку зі значущими коефіцієнтами:\n\tY = " , end="" )  
 if dmas[ 0 ] != 0:  
 print ( "{:.3f}".format ( b0[ 0 ] ) , end="" )  
 for i in range ( 1 , 11 ):  
 if dmas[ i ] != 0:  
 print ( " + {:.3f}\*{}".format ( b0[ i ] , titles\_x[ i ] ) , end="" )  
 print ()  
  
  
l = 1.73  
  
x1min = 10  
x1max = 30  
x01 = (x1min + x1max) / 2  
xl1 = l \* (x1max - x01) + x01  
  
x2min = 10  
x2max = 35  
x02 = (x2min + x2max) / 2  
xl2 = l \* (x2max - x02) + x02  
  
x3min = 10  
x3max = 25  
x03 = (x3min + x3max) / 2  
xl3 = l \* (x3max - x03) + x03  
  
xnorm = [ [ -1 , -1 , -1 ] ,  
 [ -1 , 1 , 1 ] ,  
 [ 1 , -1 , 1 ] ,  
 [ 1 , 1 , -1 ] ,  
 [ -1 , -1 , 1 ] ,  
 [ -1 , 1 , -1 ] ,  
 [ 1 , -1 , -1 ] ,  
 [ 1 , 1 , 1 ] ,  
 [ -l , 0 , 0 ] ,  
 [ l , 0 , 0 ] ,  
 [ 0 , -l , 0 ] ,  
 [ 0 , l , 0 ] ,  
 [ 0 , 0 , -l ] ,  
 [ 0 , 0 , l ] ,  
 [ 0 , 0 , 0 ] ]  
  
xnat = [ [ x1min , x2min , x3min ] ,  
 [ x1min , x2min , x3max ] ,  
 [ x1min , x2max , x3min ] ,  
 [ x1min , x2max , x3max ] ,  
 [ x1max , x2min , x3min ] ,  
 [ x1max , x2min , x3max ] ,  
 [ x1max , x2max , x3min ] ,  
 [ x1max , x2max , x3max ] ,  
 [ -xl1 , x02 , x03 ] ,  
 [ xl1 , x02 , x03 ] ,  
 [ x01 , -xl2 , x03 ] ,  
 [ x01 , xl2 , x03 ] ,  
 [ x01 , x02 , -xl3 ] ,  
 [ x01 , x02 , xl3 ] ,  
 [ x01 , x02 , x03 ] ]  
  
n = 15  
m = 2  
  
while True:  
 while True:  
 print ( "\nПоточний m = {}\n".format ( m ) )  
 xnatmod = [ [ xnat[ i ][ j ] for i in range ( 15 ) ] for j in range ( 3 ) ]  
 maty = geny ( n , m )  
 matymod = [ maty[ i ][ -1 ] for i in range ( len ( maty ) ) ]  
  
 kohren\_flag = kohren ( maty , 3 , 15 )  
 print ( "Дисперсія {}однорідна, з ймовірністю = {:.2}"  
 .format ( "" if kohren\_flag else "не " , 0.95 ) )  
 if kohren\_flag:  
 break  
 else:  
 m += 1  
  
 b0 = get\_b ( matymod )  
  
 dmas = student ( n , m , maty )  
 d = sum ( dmas )  
  
 fishercheck = fisher ( b0 , xnatmod , n , m , d , maty )  
 print ( "Рівняння {}адекватне, з ймовірністю = {:.2f}\n"  
 .format ( "" if fishercheck else "не " , 0.95 ) )  
 all\_print ()  
 print ( "\nКількість значущих коефіцієнтів, d = {}".format ( d ) )  
 if fishercheck:  
 break

**Результати роботи програми**

