**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

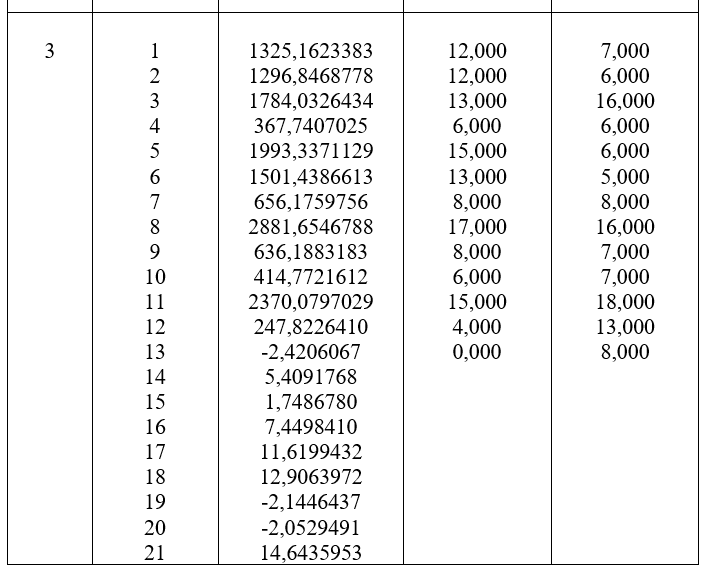
Лабораторная работа №5  
дисциплина: «Системный анализ и обработка информации»  
тема: «Регрессионный анализ»

Выполнил: ст. группы ВТ-32  
 Воскобойников И. С.  
Проверил: Полунин А.И.

Белгород 2021 г.

**Цель работы:** получить математическую модель процесса, заданного его значениями при разных значениях его двух аргументов. Числовые значения процесса измеряются со случайной ошибкой, распределенной по нормальному закону, математическое ожидание ошибки равно нулю, дисперсия неизвестна. Процесс описывается алгебраической зависимостью.

**Вариант 3**



На первом шаге исследований выбирают вид математической модели процесса исходя из его физической сущности. Представим его в виде

где ̶ оценки неизвестных коэффициентов; ̶ регрессоры, зависящие от двух аргументов х1, х2; ̶ число регрессоров. На основании таблицы экспериментальных данных для координат х1, х2 вычисляем матрицу



В данном случае:

F= [[ 12. 7. 84.]

[ 12. 6. 72.]

[ 13. 16. 208.]

[ 6. 6. 36.]

[ 15. 6. 90.]

[ 13. 5. 65.]

[ 8. 8. 64.]

[ 17. 16. 272.]

[ 8. 7. 56.]

[ 6. 7. 42.]

[ 15. 18. 270.]

[ 4. 13. 52.]]

Вектор оцениваем по методу наименьших квадратов

где Y – вектор оценки значений исследуемого процесса, полученный в экспериментах.

Для этого вычислим среднее арифметическое результатов эксперимента

и величины , где — значение модели, полученное при наборе регрессоров, вычисленных для i-го эксперимента.

= [[2.95952526]

[1.94973939]

[1.0191427 ]]

**Программная реализация**

from itertools import \*  
import numpy as np  
import matplotlib.pyplot as plt  
import math  
import scipy  
import scipy.optimize as opt  
import scipy.integrate as integrate  
  
def r1(x1,x2):  
 return 1  
def rx1(x1,x2):  
 return x1  
def rx2(x1,x2):  
 return x2  
def rx1\_2(x1,x2):  
 return x1\*x1  
def rx2\_2(x1,x2):  
 return x2\*x2  
def rx2x1(x1,x2):  
 return x2\*x1  
  
arm=[r1,rx1,rx1\_2,rx2x1,rx2\_2]  
  
def Get\_F(i:int,RRR):*#возвр.матр. F* count\_r=len(RRR[i])  
 F=np.zeros((n1,count\_r ))*#M: 12 x кол-во регрессоров* for it in range(n1):  
 for j in range(count\_r):  
 F[it][j]=RRR[i][j](X1[it][0],X2[it][0])  
 return F  
  
def Get\_res\_s(it:int,A):  
 xx0=[]  
 xx1=[]  
 xx2=[]  
 yy0=[]  
 yynew=[]  
 for i in range(n1):  
 xx0.append(i)  
 xx1.append(X1[i][0])  
 xx2.append(X2[i][0])  
 yy0.append(Y0[i][0])  
 tmp=0.0  
 for j in range(len(A)):  
 *# print("j = ", A[j][0], " RRR = ", RRR[it][j](X1[i][0],X2[i][0]))* tmp+=A[j][0]\*RRR[it][j](X1[i][0],X2[i][0])  
 yynew.append(tmp)  
 return xx0,xx1,xx2,yy0,yynew  
  
def Get\_Grafic(xx0,yy0,yynew):  
 fig, ax = plt.subplots()  
 ax.scatter(xx0, yy0)  
 ax.plot(xx0, yy0, **'r'**, lw=2, label=**"Theoretical"**)  
 ax.plot(xx0, yynew, **'b'**, lw=2, label=**"Fit"**)  
 ax.legend()  
 ax.set\_xlim(0, 13)  
 ax.set\_xlabel(**r"$x$"**, fontsize=18)  
 ax.set\_ylabel(**r"$y$"**, fontsize=18)  
 plt.show()  
 return  
  
n1=13  
n2=10  
*##вектор x1,x2,y*X1=np.array([  
 [12.0],  
 [12.0],  
 [13.0],  
 [6.0],  
 [15.0],  
 [13.0],  
 [8.0],  
 [17.0],  
 [8.0],  
 [6.0],  
 [15.0],  
 [4.0],  
 [0.0]  
])  
  
X2=np.array([  
 [7.0],  
 [6.0],  
 [16.0],  
 [6.0],  
 [6.0],  
 [5.0],  
 [8.0],  
 [16.0],  
 [7.0],  
 [7.0],  
 [18.0],  
 [13.0],  
 [8.0]  
])  
  
Y0=np.array([  
 [1325.1623383],  
 [1296.8468778],  
 [1784.0326434],  
 [367.7407025],  
 [1993.3371129],  
 [1501.4386613],  
 [656.1759756],  
 [2881.6546788],  
 [636.1883183],  
 [414.7721612],  
 [2370.0797029],  
 [247.8226410],  
 [-2.4206067],  
])  
  
Y=np.array([  
  
 5.4091768,  
 1.7486780,  
 7.4498410,  
 11.6199432,  
 12.9063972,  
 -2.1446437,  
 -2.0529491,  
 14.6485953,  
  
])  
  
  
def Get\_Qo(yy0,yynew):  
 Qo=0.0 *#Q остаток* for i in range(n1):  
 Qo+=(yy0[i]-yynew[i])\*\*2  
 return Qo  
  
def Get\_RRR():  
 tf=[]  
 *# for i in range(2, len(arm)):* j = combinations(arm, 3) *# ВООООООТТТУУУТТ РЕГРЕССОРЫ -----------* tf+=list(j)  
 return tf  
global r\_min  
r\_min=0  
global r\_num  
r\_num=0  
RRR=Get\_RRR()  
for iterat in range(len(RRR)):  
 F=Get\_F(iterat,RRR)  
 A=(np.linalg.inv(((F.transpose()).dot(F))).dot(F.transpose())).dot(Y0)  
 print(**'A='**,A)  
 xx0,xx1,xx2,yy0,yynew=Get\_res\_s(iterat,A)  
 *#print(yy0)  
 #print(yynew)* s = 0  
 for i in range(len(A)):  
 s+=A[i]  
 my\_t=np.mean(yy0)*#MY теор* print(**'мат.ожидание теор='**,my\_t)  
 Q=0.0 *#Q теор* Qr=0.0 *#Q реал* Qo=0.0 *#Q остаток* for i in range(n1):  
 Q+=(yy0[i]-my\_t)\*\*2  
 Qr+=(yynew[i]-my\_t)\*\*2  
 Qo+=(yy0[i]-yynew[i])\*\*2  
 R\_2=Qr/Q  
 R\_22=1.0-Qo/Q  
 if(abs(1-r\_min) > abs(1-R\_2)):  
 r\_min=R\_2  
 r\_num=iterat  
 print(**'F=**\n**'**,F)  
 print(**'S = '**, s, **" R = "**,R\_2, **"R22 = "**, R\_22)  
 y\_13\_22=np.average(Y)  
 q\_e\_2=0.0  
 l=len(Y)  
 for i in range(l):  
 q\_e\_2+=((Y[i]-y\_13\_22)\*\*2)/(l-1)  
 q\_e\_2=np.sqrt(q\_e\_2)  
 q\_a=[]  
 C=np.abs( np.linalg.inv( np.dot(F.transpose(),F) ))  
 q\_a=np.zeros(len(C))  
 for i in range(len(q\_a)):  
 q\_a[i]=np.sqrt(C[i][i])\*q\_e\_2  
  
 print(**'q\_a='**,q\_a)  
  
 print(**"--------------- i = "**,iterat)  
F=Get\_F(r\_num,RRR)  
A=(np.linalg.inv(((F.transpose()).dot(F))).dot(F.transpose())).dot(Y0)  
print(**'A='**,A)  
xx0,xx1,xx2,yy0,yynew=Get\_res\_s(r\_num,A)  
  
print(**'i='**,r\_num,**'func='**,RRR[r\_num])*#STR\_RRR  
#Get\_Grafic(xx0,yy0,yynew)*my\_t=np.mean(yy0)*#MY теор*print(**'мат.ожидание теор='**,my\_t)  
Q=0.0 *#Q теор*Qr=0.0 *#Q реал*Qo=0.0 *#Q остаток*for i in range(n1):  
 Q+=(yy0[i]-my\_t)\*\*2  
 Qr+=(yynew[i]-my\_t)\*\*2  
 Qo+=(yy0[i]-yynew[i])\*\*2  
  
R\_2=Qr/Q  
R\_22=1.0-Qo/Q  
*##print('R^2=',R\_2,'=',R\_22)*print(**'Q остаток='**,Qo,**'Q реал='**,Qr,**'Q теор='**,Q)  
print(**'R^2='**,R\_2,**'='**,R\_22)  
K=len(A)*#кол-во регрессоров*Sr\_2=Qr/(K-1)  
*#Sr\_2=0#?????????????????????????????????????????????????????????*So\_2=Qo/(n1-K)  
FF=Sr\_2/So\_2  
print(**'F='**,FF)  
  
y\_13\_22=np.average(Y)  
q\_e\_2=0.0  
l=len(Y)  
for i in range(l):  
 q\_e\_2+=((Y[i]-y\_13\_22)\*\*2)/(l-1)  
q\_e\_2=np.sqrt(q\_e\_2)  
*#print(F)  
#print(q\_e\_2)*C=np.abs( np.linalg.inv( np.dot(F.transpose(),F) ))  
q\_a=np.zeros(len(C))  
for i in range(len(q\_a)):  
 q\_a[i]=np.sqrt(C[i][i])\*q\_e\_2  
q\_a=q\_a\*\*2  
print(**'q\_a='**,q\_a)  
  
T=np.zeros(len(q\_a))  
sk=np.abs(np.sqrt(A[0][0]\*A[0][0]+A[1][0]\*A[1][0]+A[2][0]\*A[2][0]))  
for i in range(len(q\_a)):  
 T[i]=sk/q\_a[i]  
  
print(**'T='**,T)  
  
q\_a=np.sqrt(q\_a)  
t\_r=2.2621  
for i in range(len(q\_a)):  
 print(**"Довер.интервалы i="**,i+1,**" ;"**,A[i][0]-t\_r\*q\_a[i],**" <= "**,A[i][0],**" <= "**,A[i][0]+t\_r\*q\_a[i])

1+x1+x1^2

1+x1^2+x2x1

x1+x2+x1x2

График:

