Національний Технічний Університет України

«Київський Політехнічний Інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

|  |  |
| --- | --- |
| Прийняв  Доц. Марковський О.П.  «\_\_»\_\_\_\_\_\_\_\_ 2015 р. | Виконав  Студент 2-ого курсу ФІОТ  групи ІО-41  Смішний Д.М. |

**Варіант завдання**:

|  |  |
| --- | --- |
| Кількість вибірок | Тип регресії |
| 2 | Степенева(4) |

**Код програми**

**public** **class** Regresion {

MatrixCalculation mc = **new** MatrixCalculation();

**public** **double**[] arrayX2Create(**double** xArr1[] ){

**double** xArr2[]= **new** **double**[xArr1.length];

**for** (**int** k=0; k<xArr1.length; k++){

xArr2[k]=xArr1[k];

}

**for** (**int** i=0; i<xArr1.length; i++){

xArr2[i]=Math.*pow*(xArr2[i],2);

}

**return** xArr2;

}

**public** **double**[] arrayX3Create(**double** xArr1[]){

**double** xArr3[]=**new** **double**[xArr1.length];

**for** (**int** k=0; k<xArr1.length; k++){

xArr3[k]=xArr1[k];

}

**for** (**int** i=0; i<xArr1.length; i++){

xArr3[i]=Math.*pow*(xArr3[i],3);

}

**return** xArr3;

}

**public** **double**[] arrayX4Create(**double** xArr1[]){

**double** xArr4[]=**new** **double**[xArr1.length];

**for** (**int** k=0; k<xArr1.length; k++){

xArr4[k]=xArr1[k];

}

**for** (**int** i=0; i<xArr1.length; i++){

xArr4[i]=Math.*pow*(xArr4[i],4);

}

**return** xArr4;

}

**public** **double** mathExpect(**double** arr[]){

**double** result=0;

**for** (**int** i=0; i<arr.length; i++){

result+=arr[i];

}

**return** result/(**double**)arr.length;

}

**public** **double** covariation(**double** arr1[], **double**[] arr2){

**double** m1 = mathExpect(arr1);

**double** m2 = mathExpect(arr2);

**double** cov=0.;

**for** (**int** i=0; i<arr1.length; i++){

cov+=(arr1[i]-m1)\*(arr2[i]-m2);

}

**return** cov/(**double**)arr1.length;

}

**public** **double** covariationMatrix(**double**[] arr1, **double** arr2[], **double** arr3[],**double** arr4[], **double** arrY[], **int** index){

**double** cov11=covariation(arr1,arr1);

**double** cov12=covariation(arr1,arr2);

**double** cov13=covariation(arr1,arr3);

**double** cov14=covariation(arr1,arr4);

**double** cov21 = covariation(arr2,arr1);

**double** cov22= covariation(arr2,arr2);

**double** cov23 = covariation(arr2,arr3);

**double** cov24 = covariation(arr2,arr4);

**double** cov31 = covariation(arr3, arr1);

**double** cov32 = covariation(arr3,arr2);

**double** cov33 = covariation(arr3,arr3);

**double** cov34 = covariation(arr3,arr4);

**double** cov41 = covariation(arr4,arr1);

**double** cov42 = covariation(arr4, arr2);

**double** cov43 = covariation(arr4, arr3);

**double** cov44 = covariation(arr4,arr4);

**double** cov1y = covariation(arr1, arrY);

**double** cov2y = covariation(arr2, arrY);

**double** cov3y = covariation(arr3, arrY);

**double** cov4y = covariation(arr4, arrY);

**double**[] covYArray = {cov1y, cov2y, cov3y, cov4y};

**double** covMatrix[][] = {{cov11, cov12, cov13, cov14},

{cov21, cov22, cov23, cov24},

{cov31, cov32, cov33, cov34},

{cov41, cov42, cov43, cov44}};

**double** L=0;

**if** (index!=-1){

**for** (**int** i=0; i<covMatrix.length; i++){

**for** (**int** j=0; j<covMatrix[i].length;j++){

covMatrix[i][index]=covYArray[i];

}

}

L = mc.CalculateMatrix(covMatrix);

} **else**{

L = mc.CalculateMatrix(covMatrix);

};

**return** L;

}

**public** **double** beta(**double** deltaL, **double** L){

**return** deltaL/L;

}

**public** **double** alpha(**double**[] beta, **double**[]mathExpectXArr, **double** mathExpectY){

**double** sum=0;

**for** (**int** i=0; i<beta.length; i++){

sum+=beta[i]\*mathExpectXArr[i];

}

**return** mathExpectY - sum;

}

}

**public** **class** MatrixCalculation {

**public** **double** CalculateMatrix(**double**[][] matrix){

**double** calcResult=0.0;

**if** (matrix.length==2){

calcResult=matrix[0][0]\*matrix[1][1]-matrix[1][0]\*matrix[0][1];

}

**else**{

**int** koeff=1;

**for**(**int** i=0; i<matrix.length; i++){

**if**(i%2==1){

koeff=-1;

}

**else**{

koeff=1;

};

calcResult += koeff\*matrix[0][i]\***this**.CalculateMatrix(**this**.GetMinor(matrix,0,i));

}

}

**return** calcResult;

}

**private** **double**[][] GetMinor(**double**[][] matrix, **int** row, **int** column){

**int** minorLength = matrix.length-1;

**double**[][] minor = **new** **double**[minorLength][minorLength];

**int** dI=0;

**int** dJ=0;

**for**(**int** i=0; i<=minorLength; i++){

dJ=0;

**for**(**int** j=0; j<=minorLength; j++){

**if**(i==row){

dI=1;

}

**else**{

**if**(j==column){

dJ=1;

}

**else**{

minor[i-dI][j-dJ] = matrix[i][j];

}

}

}

}

**return** minor;

}

}

**public** **class** WorkClass {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**double**[] X1 = {5, 3, 13, 6, 4, 2, 8, 1, 10, 9};

**double**[] Y = {23, 42, 53, 2, 4, 2, 13, 2, 12, 5};

Regresion r = **new** Regresion();

**double**[] X2 = r.arrayX2Create(X1);

**double**[] X3 = r.arrayX3Create(X1);

**double**[] X4 = r.arrayX4Create(X1);

**double** mathExpectY = r.mathExpect(Y);

**double** mathExpextX1 = r.mathExpect(X1);

**double** mathExpextX2 = r.mathExpect(X2);

**double** mathExpextX3 = r.mathExpect(X3);

**double** mathExpextX4 = r.mathExpect(X4);

**double**[] mathExpectArr = {mathExpextX1,mathExpextX2,mathExpextX3,mathExpextX4};

**double** L = r.covariationMatrix(X1, X2, X3, X4, Y, -1);

**double** L1 = r.covariationMatrix(X1, X2, X3, X4, Y, 0);

**double** L2 = r.covariationMatrix(X1, X2, X3, X4, Y, 1);

**double** L3 = r.covariationMatrix(X1, X2, X3, X4, Y, 2);

**double** L4 = r.covariationMatrix(X1, X2, X3, X4, Y, 3);

**double** beta1 = r.beta(L1, L);

**double** beta2 = r.beta(L2, L);

**double** beta3 = r.beta(L3, L);

**double** beta4 = r.beta(L4, L);

**double**[] betaArr = {beta1, beta2, beta3, beta4};

**double** alpha =r.alpha(betaArr, mathExpectArr, mathExpectY);

System.***out***.println("Бети: ");

**for** (**int** i=0; i<betaArr.length; i++){

System.***out***.print(betaArr[i]+" ");

}

System.***out***.println();

System.***out***.println("Альфа: ");

System.***out***.println(alpha);

}

}

**Результати:**

Бети:

37.75727746065271 -9.228463514842687 0.8044502688645221 -0.021565313894310982

Альфа:

-29.92700035345062