关于决策树ID3算法的实现代码：

**package** artificialIntelligence;

**import** java.util.Scanner;

**public** **class** ID3 {

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

Scanner read=**new** Scanner(System.***in***);

**int** caseNumber;

**int** xNumber;

System.***out***.println("请输入样本数：");

caseNumber=read.nextInt();

System.***out***.println("请输入属性数：");

xNumber=read.nextInt();

ID3 id=**new** ID3();

**int**[][] table=id.creatTable(caseNumber,xNumber);

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

System.***out***.println("请输入第"+(i+1)+"行数据");

**for**(**int** j=0;j<xNumber+1;j++) {

table[i][j]=read.nextInt();

}

}

id.id3tree(table,caseNumber,xNumber);

}

**public** **int**[][] creatTable(**int** caseNumber,**int** xNumber)

{

**int**[][] a;

a=**new** **int**[caseNumber][xNumber+1];

**return** a;

}

**public** **int**[][] evalution(**int**[][] table,**int** caseNumber,**int** xNumber) {

Scanner read=**new** Scanner(System.***in***);

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

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

table[i][j]=read.nextInt();

}

}

**return** table;

}

**public** **int**[][] creatClassification(**int** caseNumber)

{

**int**[][] a;

a=**new** **int**[2][caseNumber+1];

**return** a;

}

**public** **int**[][] classify(**int**[][] table,**int** clomun,**int** caseNumber)

{

**int** count=1;

**int**[][] classification=creatClassification(caseNumber);

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

**for**(**int** j=1;j<caseNumber+1;j++)

{

**if**(classification[0][j]!=table[i][clomun]&&j==count)

{

classification[0][count]=table[i][clomun];

classification[1][count]=1;

count++;

**break**;

}

**else** {

**if**(classification[0][j]==table[i][clomun])

{

classification[1][j]++;

**break**;

}

}

}

}

classification[0][0]=count;

classification[1][0]=clomun;

**return** classification;

}

**public** **double** getES(**int**[][] waitCalculate) {

**double** total=0;

**double** ES=0;

**for**(**int** i=1;i<waitCalculate[0][0];i++) {

total=total+(**double**)waitCalculate[1][i];

}

**for**(**int** i=1;i<waitCalculate[0][0];i++) {

ES=ES-cutFive((**double**)waitCalculate[1][i], total)\*(Math.*log*(cutFive((**double**)waitCalculate[1][i], total))/Math.*log*(2.0));

}

**return** ES;

}

**public** **double** cutFive(**double** a,**double** b) {

**double** c=a/b\*100000;

**int** d=(**int**)c;

**double** e=(**double**)(d%100000)/100000.0;

**double** f=d/100000;

**double** g=f+e;

**return** g;

}

**public** **int**[][] creatX1(**int** row){ //建立各属性取不同值时的表

**int**[][] a;

a=**new** **int**[row][2];

**return** a;

}

**public** **double** getESX1(**int**[][] table,**int**[][] classifyX,**int** caseNumber,**int** xNumber){//计算加权信息熵

**double**[] ESX1=**new** **double**[classifyX[0][0]-1];

**double** total=0;

**for**(**int** i=1;i<classifyX[0][0];i++) {

total=total+classifyX[1][i];

}

**for**(**int** i=1;i<classifyX[0][0];i++)

{

**int** count=0;

**int**[][] a=creatX1(classifyX[1][i]);

**for**(**int** j=0;j<caseNumber;j++)

{

**if**(table[j][classifyX[1][0]]==classifyX[0][i])

{

a[count][1]=table[j][xNumber];

count++;

}

}

a[0][0]=classifyX[0][i];

**int**[][] array=classify2(a,1,classifyX[1][i],caseNumber);

ESX1[i-1]=getES(array);

}

**double** ESX=0;

**for**(**int** i=1;i<classifyX[0][0];i++) {

ESX=ESX+cutFive((**double**)classifyX[1][i], total)\*ESX1[i-1];

}

**return** ESX;

}

**public** **int**[][] classify2(**int**[][] table,**int** clomun,**int** row,**int** caseNumber)

{

**int** count=1;

**int**[][] classification=creatClassification(caseNumber);

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

**for**(**int** j=1;j<row+1;j++)

{

**if**(classification[0][j]!=table[i][clomun]&&j==count)

{

classification[0][count]=table[i][clomun];

classification[1][count]=1;

count++;

**break**;

}

**else** {

**if**(classification[0][j]==table[i][clomun])

{

classification[1][j]++;

**break**;

}

}

}

}

classification[0][0]=count;

classification[1][0]=clomun;

**return** classification;

}

**public** **double** getESX(**int**[][] table,**int** clomun,**int** caseNumber,**int** xNumber) {//得到clomun列属性信息增益

**int**[][] classifiX=creatClassification(caseNumber);

classifiX=classify(table, clomun,caseNumber);

**double** ESX=getESX1(table, classifiX,caseNumber,xNumber);

**return** ESX;

}

**public** **int** getMinESX(**int**[][] table,**int** caseNumber,**int** xNumber) {//得到最小信息增益所在列

**int** j=-1;

**double**[] ESX=**new** **double**[xNumber];

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

ESX[i]=getESX(table, i,caseNumber,xNumber);

}

**double** ESXMin=getMin(ESX,xNumber);

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

**if**(ESX[i]==ESXMin)

j=i+1;

}

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

**return** j;

}

**public** **double** getMin(**double**[] ESX,**int** xNumber)//得到最小信息增益

{

**double** min=0;

**for**(**int** i=0;i<xNumber;i++)

{

**for**(**int** j=0;j<xNumber;j++)

{

**if**(ESX[i]<=ESX[j]&&j==xNumber-1)

min=ESX[i];

**else**

{

**if**(ESX[i]>ESX[j])

**break**;

}

}

}

**return** min;

}

**public** **int** djustLeaf(**int**[][] table,**int** caseNumber,**int** xNumber)

{

**int** clomun=getMinESX(table,caseNumber,xNumber)-1;

**int**[][] a=classify(table, clomun,caseNumber);

**for**(**int** i=1;i<a[0][0];i++)

{

**int** count=0;

**int**[][] b=creatX1(a[1][i]);

**for**(**int** j=0;j<caseNumber;j++)

{

**if**(table[j][a[1][0]]==a[0][i])

{

b[count][1]=table[j][xNumber];

count++;

}

}

b[0][0]=a[0][i];

**int**[][] array=classify2(b,1,a[1][i],caseNumber);

**if**(array[0][0]==2)

{

System.***out***.println(a[0][i]+" leaf");

}

**else** {

System.***out***.println(a[0][i]+" noleaf");

}

}

**return** clomun;

}

**public** **int**[][] creatNewTable(**int**[][] table,**int** delete\_col,**int** value)//删去当前信息增益最大属性

{

**int** new\_line\_cur=0;

**int**[][] newtable=**new** **int**[table.length][table[0].length];

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

{

**if**(table[i][delete\_col]==value)

**continue**;

**else**

**for**(**int** j=0;j<table[0].length-1;j++)

{

**if**(j>=delete\_col)

{

newtable[new\_line\_cur][j]=table[i][j+1];

**if** (j == table[0].length-2) {

new\_line\_cur++;

}

}

**else**

{

newtable[new\_line\_cur][j]=table[i][j];

}

}

}

newtable[0][newtable[0].length-1]=new\_line\_cur;

**return** newtable;

}

**public** **void** id3tree(**int**[][] table,**int** caseNumber,**int** xNumber) {

**while**(xNumber>=1){

**int** delete=djustLeaf(table,caseNumber,xNumber);

**int**[][] array=classify(table, delete,caseNumber);

**for**(**int** i=1;i<array[0][0];i++) {

**int** x=xNumber;

**int** c=caseNumber;

**int**[][] table1=creatTable(c, x+1);

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

**for**(**int** k=0;k<x+1;k++) {

table1[j][k]=table[j][k];

}

}

**int**[][] newtable=creatNewTable(table1, delete,array[0][i]);

c=newtable[0][xNumber-1];

x--;

id3tree(newtable,c,x);

}

xNumber--;

}

}

}