

数据挖掘大作业二：关联规则挖掘

学院：计算机学院 学号：2120151036 姓名：王丹

一、任务说明

本次统计任务是在标准数据集上进行关联规则的抽取与分析，从而发现有价值的规则。在对标准数据集预处理部分和抽取规则与评价的部分使用 Java 编写，采用了 Apriori 算法。可视化部分采用 R 编写。

二、数据集介绍

a) 数据来源

使用了泰坦尼克的机器学习数据集，其中训练数据有近 900 条，测试数据有 400 余条。数据来源于网站：

<https://www.kaggle.com/c/titanic/data>

b) 数据样例：

[PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Cabin, Embarked, Ticket, Fare]

[1, 0, 3, "Braund, Mr. Owen Harris", male, 22, 1, 0, S, A/5 21171, 7.25]

c) 字段说明：

表一 数据字段说明

字段	说明
survived	Survival (0 = No; 1 = Yes)
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
name	Name
Sex	Sex
Age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
cabin	Cabin
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
ticket	Ticket Number
fare	Passenger Fare

三、预处理

A) 删除信息含量低的字段

在关联规则挖掘中，我们认为，乘客 ID，票号，费用以及姓名信息、房产对规则没有任何帮助，故将其去除，只对其他属性进行关联规则分析。使用 LiberOffice Calc 删除。

B) 替换部分数值内容

在 Survive 字段中，将 0 替换为 Dead，1 替换为 Survive。

在 Pclass 字段中，分别将数字替换为坐席等级：1 = 1class; 2 = 2class; 3 = 3class。

对年龄，进行了划分。0-12 岁为 child，12-20 为 teenager，20-40 为 youth，40-60 为 middle-aged，60 以上为 elderly。

亲属情况，按照是否为 0，简单的分为了有和无两种情况。

对于缺失项，因其对关联规则抽取意义不大，所以直接去除。

预处理后的数据样例：

[Survived,Pclass,Sex,Age,SibSp,Parch,Embarked]

```
BufferedReader reader = new BufferedReader(new FileReader("./train.csv"));
reader.readLine();//第一行信息，为标题信息，不用,如果需要，注释掉
File csv = new File("./predata.txt");// txt 数据文件
BufferedWriter bw = new BufferedWriter(new FileWriter(csv, true)); // 附加
String line = null;
double age = 0;
while ((line = reader.readLine()) != null) {
    if (line.contains(",,")) continue;//去除不完全项
    String item[] = line.split(",");//CSV 格式文件为逗号分隔符文件，这里根据逗号切分
    if(item.length!=7) continue;
    if (item[0].equals("0")) { item[0] = "Dead"; }
    else { item[0] = "Survived"; }
    item[1] += "class";
    age = Double.parseDouble(item[3]);
    if (age <= 12) {
        item[3] = "child";
    } else if (age <= 20) {
        item[3] = "teenager";
    } else if (age <= 40) {
        item[3] = "youth";
    } else if (age <= 60) {
        item[3] = "middle-aged";
    } else {
        item[3] = "elderly";
    }
    if (item[4].equals("0")) {
        item[4] = "noSibsp";
    } else {
        item[4] = "yesSibsp";
    }
    if (item[5].equals("0")) {
        item[5] = "noParch";
    } else {
        item[5] = "yesParch";
    }
    bw.write(item[0] + " " + item[1] + " " + item[2] + " " + item[3] + " " + item[4] + " " +
item[5]+" "+item[6]);
    bw.newLine(); // 添加新的数据行
}
```

四、找出频繁项集

首先，读取文件。

```
String encoding = "GBK"; // 字符编码(可解决中文乱码问题 )
File file = new File("./predata.txt");
if (file.isFile() && file.exists()) {
    InputStreamReader read = new InputStreamReader(
        new FileInputStream(file), encoding); // 读入文件
    BufferedReader bufferedReader = new BufferedReader(read);
    String lineTXT = null;
    while ((lineTXT = bufferedReader.readLine()) != null) { // 读一行文件
        String[] lineString = lineTXT.split(" "); // 以空格分隔数据
        List<String> lineList = new ArrayList<String>();
        for (int i = 0; i < lineString.length; i++) {
            lineList.add(lineString[i]);
        }
        record.add(lineList); // 将数据加入 ListArray 中
    }
    read.close();
}
```

编写函数获得频繁 1-项集，设定频繁项集的最小支持度为 0.2。

```
/**
 * 首先用 List<List<String>> 类型的 record 将矩阵形式的数据读入内存；
 */
record = getRecord(); // 获取原始数据记录
List<List<String>> cItemset = findFirstCandidate(); // 获取第一次的备选集
List<List<String>> lItemset = getSupportedItemset(cItemset); // 获取备选集 cItemset
满足支持的集合
for (int i = 0; i < lItemset.size(); i++) {
    System.out.println(lItemset.get(i));
}
```

抽取完后，使用 println 函数打印抽取的频繁 1-项集情况：

[Dead]	0.5955056179775281
[3class]	0.49859550561797755
[male]	0.6362359550561798
[youth]	0.5393258426966292
[yesSibsp]	0.34129213483146065
[noParch]	0.7289325842696629
[S]	0.7780898876404494
[Survived]	0.4044943820224719
[1class]	0.25842696629213485
[female]	0.3637640449438202
[noSibsp]	0.6587078651685393
[yesParch]	0.2710674157303371
[2class]	0.24297752808988765

图 1 频繁 1-项集的分布情况

频繁 k-项集计算如下：

```

/**
 * 程序先求出 k-1 备选集，由备选集和数据库记录 record 求得满足支持度的 k-1
级集合，在满足支持度集合中求出满足自信度的集合，
 * 若满足置信度的集合为空，程序停止； 否则输出满足自信度的集合，以及对
应的支持度和自信度，并由满足支持度的 k-1 级集合求出 k 级备选集，进入下一轮
循环；
 * 直至程序结束，输出全部频繁集
 */
while (endTag != true) { // 只要能继续挖掘
    List<List<String>> ckItemset = getNextCandidate(lItemset); // 获取第下一次的
    备选集
    List<List<String>> lkItemset = getSupportedItemset(ckItemset); // 获取备选
    集 cItemset 满足支持的集合
    getConfidencedItemset(lkItemset, lItemset, dkCountMap, dCountMap); // 获
    取备选集 cItemset 满足置信度的集合
    if (conflItemset.size() != 0) // 满足置信度的集合不为空
    {
        printConflItemset(conflItemset); // 打印满足置信度的集合
    }
    conflItemset.clear(); // 清空置信度的集合
    cItemset = ckItemset; // 保存数据，为下次循环迭代准备
    lItemset = lkItemset;
    dCountMap.clear();
    dCountMap.putAll(dkCountMap);
}

```

频繁 2-项集如下：

[Dead, 3class]	0.3792134831460674
[Dead, male]	0.5056179775280899
[Dead, youth]	0.3258426966292135
[Dead, noParch]	0.4705056179775281
[Dead, S]	0.4957865168539326
[Dead, noSibsp]	0.4157303370786517
[3class, male]	0.3553370786516854
[3class, youth]	0.2696629213483146
[3class, noParch]	0.3707865168539326
[3class, S]	0.40730337078651685
[3class, noSibsp]	0.3497191011235955
[male, youth]	0.3497191011235955
[male, noParch]	0.5168539325842697
[male, S]	0.5168539325842697
[male, noSibsp]	0.4592696629213483
[youth, noParch]	0.4452247191011236
[youth, S]	0.42696629213483145
[youth, Survived]	0.21348314606741572
[youth, noSibsp]	0.38202247191011235
[yesSibsp, S]	0.2598314606741573
[noParch, S]	0.5730337078651685
[noParch, Survived]	0.25842696629213485
[noParch, female]	0.21207865168539325
[noParch, noSibsp]	0.5646067415730337
[S, Survived]	0.28230337078651685
[S, female]	0.2612359550561798
[S, noSibsp]	0.5182584269662921
[S, yesParch]	0.2050561797752809
[S, 2class]	0.21910112359550563
[Survived, female]	0.273876404494382
[Survived, noSibsp]	0.24297752808988765

频繁 3-项集:

[Dead, 3class, male]	0.30196629213483145
[Dead, 3class, youth]	0.20926966292134833
[Dead, 3class, noParch]	0.29213483146067415
[Dead, 3class, S]	0.32162921348314605
[Dead, 3class, noSibsp]	0.2654494382022472
[Dead, male, youth]	0.2851123595505618
[Dead, male, noParch]	0.4241573033707865
[Dead, male, S]	0.42134831460674155
[Dead, male, noSibsp]	0.375
[Dead, youth, noParch]	0.2851123595505618
[Dead, youth, S]	0.27247191011235955
[Dead, youth, noSibsp]	0.24157303370786518
[Dead, noParch, S]	0.4002808988764045
[Dead, noParch, noSibsp]	0.3848314606741573
[Dead, S, noSibsp]	0.34691011235955055
[3class, male, youth]	0.2050561797752809
[3class, male, noParch]	0.2949438202247191
[3class, male, S]	0.300561797752809
[3class, male, noSibsp]	0.2710674157303371
[3class, youth, noParch]	0.23735955056179775
[3class, youth, S]	0.22893258426966293
[3class, youth, noSibsp]	0.21207865168539325
[3class, noParch, S]	0.3089887640449438
[3class, noParch, noSibsp]	0.3160112359550562
[3class, S, noSibsp]	0.2851123595505618
[male, youth, noParch]	0.31741573033707865
[male, youth, S]	0.28230337078651685
[male, youth, noSibsp]	0.27247191011235955
[male, noParch, S]	0.4297752808988764
[male, noParch, noSibsp]	0.42696629213483145
[male, S, noSibsp]	0.37780898876404495
[youth, noParch, S]	0.35814606741573035
[youth, noParch, noSibsp]	0.34831460674157305
[youth, S, noSibsp]	0.30196629213483145
[noParch, S, noSibsp]	0.45646067415730335

频繁 4-项集:

[Dead, 3class, male, noParch]	0.2542134831460674
[Dead, 3class, male, S]	0.25842696629213485
[Dead, 3class, male, noSibsp]	0.2303370786516854
[Dead, 3class, noParch, S]	0.25140449438202245
[Dead, 3class, noParch, noSibsp]	0.25
[Dead, 3class, S, noSibsp]	0.2247191011235955
[Dead, male, youth, noParch]	0.25842696629213485
[Dead, male, youth, S]	0.23735955056179775
[Dead, male, youth, noSibsp]	0.2205056179775281
[Dead, male, noParch, S]	0.36235955056179775
[Dead, male, noParch, noSibsp]	0.3553370786516854
[Dead, male, S, noSibsp]	0.31741573033707865
[Dead, youth, noParch, S]	0.24297752808988765
[Dead, youth, noParch, noSibsp]	0.22893258426966293
[Dead, youth, S, noSibsp]	0.20224719101123595
[Dead, noParch, S, noSibsp]	0.32865168539325845
[3class, male, noParch, S]	0.25280898876404495
[3class, male, noParch, noSibsp]	0.26264044943820225
[3class, male, S, noSibsp]	0.23174157303370788
[3class, youth, noParch, S]	0.20365168539325842
[3class, youth, noParch, noSibsp]	0.20224719101123595
[3class, noParch, S, noSibsp]	0.26264044943820225
[male, youth, noParch, S]	0.2612359550561798
[male, youth, noParch, noSibsp]	0.2598314606741573
[male, youth, S, noSibsp]	0.2205056179775281
[male, noParch, S, noSibsp]	0.3609550561797753
[youth, noParch, S, noSibsp]	0.2808988764044944

频繁 5-项集:

[Dead, 3class, male, noParch, S]	0.2205056179775281
[Dead, 3class, male, noParch, noSibsp]	0.22612359550561797
[Dead, 3class, noParch, S, noSibsp]	0.21348314606741572
[Dead, male, youth, noParch, S]	0.21910112359550563
[Dead, male, youth, noParch, noSibsp]	0.21067415730337077
[Dead, male, noParch, S, noSibsp]	0.3061797752808989
[3class, male, noParch, S, noSibsp]	0.2247191011235955
[male, youth, noParch, S, noSibsp]	0.2148876404494382

可以看出，总共抽出了 114 项频繁集，其中 3 个变元的数目最多，有 35 条。

五、 导出关联规则，计算支持度和置信度

使用 Apriori 算法，抽取关联规则。这里，我们设置的筛选条件为支持度大于 20%，置信度大于 50%。

```
/**
 * @param lItemset
 * @param lItemset
 * @param dkCountMap2
 * @param dCountMap2 根据 lItemset, lItemset, dkCountMap2, dCountMap2
 求出满足自信度的集合
 */
private static List<List<String>> getConfidencedlItemset(
    List<List<String>> lItemset, List<List<String>> lItemset,
    Map<Integer, Integer> dkCountMap2, Map<Integer, Integer>
dCountMap2) {
    for (int i = 0; i < lItemset.size(); i++) {
        getConfItem(lItemset.get(i), lItemset, dkCountMap2.get(i),
            dCountMap2);
    }
    return null;
}
```

```
male youth noParch S -->Dead相对支持度: 0.2191780821917808置信度: 0.8387096774193549
Dead youth noParch S -->male相对支持度: 0.2191780821917808置信度: 0.9017341040462428
Dead male noParch S -->youth相对支持度: 0.2191780821917808置信度: 0.6046511627906976
Dead male youth S -->noParch相对支持度: 0.2191780821917808置信度: 0.9230769230769231
Dead male youth noParch -->S相对支持度: 0.2191780821917808置信度: 0.8478260869565217
male youth noParch noSibsp -->Dead相对支持度: 0.2107481559536354置信度: 0.8108108108108109
Dead youth noParch noSibsp -->male相对支持度: 0.2107481559536354置信度: 0.9202453987730062
Dead male noParch noSibsp -->youth相对支持度: 0.2107481559536354置信度: 0.5928853754940712
Dead male youth noSibsp -->noParch相对支持度: 0.2107481559536354置信度: 0.9554140127388535
Dead male youth noParch -->noSibsp相对支持度: 0.2107481559536354置信度: 0.8152173913043478
male noParch S noSibsp -->Dead相对支持度: 0.3062873199859501置信度: 0.8482490272373541
Dead noParch S noSibsp -->male相对支持度: 0.3062873199859501置信度: 0.9316239316239316
Dead male S noSibsp -->noParch相对支持度: 0.3062873199859501置信度: 0.9646017699115044
Dead male noParch noSibsp -->S相对支持度: 0.3062873199859501置信度: 0.8616600790513834
Dead male noParch S -->noSibsp相对支持度: 0.3062873199859501置信度: 0.8449612403100775
male noParch S noSibsp -->3class相对支持度: 0.2247980330172111置信度: 0.622568093385214
3class noParch S noSibsp -->male相对支持度: 0.2247980330172111置信度: 0.8556149732620321
3class male S noSibsp -->noParch相对支持度: 0.2247980330172111置信度: 0.9696969696969697
3class male noParch noSibsp -->S相对支持度: 0.2247980330172111置信度: 0.8556149732620321
3class male noParch S -->noSibsp相对支持度: 0.2247980330172111置信度: 0.8888888888888888
youth noParch S noSibsp -->male相对支持度: 0.21496311907270813置信度: 0.765
male noParch S noSibsp -->youth相对支持度: 0.21496311907270813置信度: 0.5953307392996109
male youth S noSibsp -->noParch相对支持度: 0.21496311907270813置信度: 0.9745222929936306
male youth noParch noSibsp -->S相对支持度: 0.21496311907270813置信度: 0.827027027027027
male youth noParch S -->noSibsp相对支持度: 0.21496311907270813置信度: 0.8225806451612904
规则数目: 305
成功构建 (总时间: 0 秒)
```

图 2 查看抽取的关联规则

这样，我们抽出了 305 条关联规则。可以发现许多关联规则并不是我们想要的，我们更希望专注于右边的预测变元（如是否存活），因此，我们需要对规则集进行筛选与评价，获得我们想要的规则。

六、 去除冗余的规则

我们抽出了 305 条关联规则。可以发现许多关联规则并不是我们想要的，我们更希望专注于右边的预测变元（如是否存活）。

对于关联规则，首先按照变元进行选取。我们想知道哪种人最可能存活，因此将右变元为存活(Survived)的规则子集筛选出来：

```
//筛选右变元为幸存的规则子集
temp=conflitemset2.get(i).get(conflitemset2.get(i).size() - 2);
if(!temp.contains("Survived")) continue;
```

我们可以得到图 5 所示结果，只有一条有价值的规则，就是，女性的存活率比较高，有 27%的数据支持这个结论，其置信度达到了 75%。

```
*****频繁模式挖掘结果*****
规则数目: 52
female -->Survived相对支持度: 0.273972602739726置信度: 0.752895752895753
*****频繁模式挖掘结果*****
规则数目: 105
*****频繁模式挖掘结果*****
规则数目: 108
*****频繁模式挖掘结果*****
规则数目: 40
```

图 3 右变元为存活的规则

接下来，我们想分析，查看哪些人更可能在这场灾难中丧命。因此，抽取右变元为死亡(Dead)的子集。

```
3class male noParch noSibsp -->Dead相对支持度: 0.22620302072356868置信度: 0.8609625668449198
3class noParch S noSibsp -->Dead相对支持度: 0.21355813136635055置信度: 0.8128342245989305
male youth noParch S -->Dead相对支持度: 0.2191780821917808置信度: 0.8387096774193549
male youth noParch noSibsp -->Dead相对支持度: 0.2107481559536354置信度: 0.8108108108108109
male noParch S noSibsp -->Dead相对支持度: 0.3062873199859501置信度: 0.8482490272373541
规则数目: 43
成功构造 (总时间: 0 秒)
```

图 4 右变元为死亡的规则数目

可以看到，这次我们抽取了 43 条规则，数目较多。因此，需要对这些规则进行评价筛选。

七、 关联规则评价，使用 Lift 指标

在这里，我们分别使用支持度，置信度与提升值对规则进行评价。

```
//计算 lift 值
private static List<List<String>> liftDeadItemset(){
    List<List<String>> liftList = new ArrayList<List<String>>();
    List<String> liftString = new ArrayList<String>();
    double lift=0.0;
    int size=0;
    for(int i=0;i<deadItemset.size();i++){
        liftString = deadItemset.get(i);
        size = liftString.size();
        lift = Double.parseDouble(liftString.get(size-1))/Dead_Support;
        liftString.add(""+lift);
        liftList.add(liftString);
    }
    return liftList;
}
```

*****关联规则评价结果*****

死亡规则数目: 43

```
3class -->Dead相对支持度: 0.3793466807165437置信度: 0.7605633802816901lift: 1.2771724687749135
male -->Dead相对支持度: 0.505795574288725置信度: 0.7947019867549668lift: 1.3344995626640008
youth -->Dead相对支持度: 0.3259571478749561置信度: 0.6041666666666666lift: 1.0145440251572326
noParch -->Dead相对支持度: 0.47067088162978576置信度: 0.6454720616570327lift: 1.0939059148580361
S -->Dead相对支持度: 0.495960660344222置信度: 0.6371841155234657lift: 1.069988420407329
noSibsp -->Dead相对支持度: 0.4158763610818405置信度: 0.6311300639658849lift: 1.0598221828861085
3class male -->Dead相对支持度: 0.3020723568668774置信度: 0.8498023715415021lift: 1.4270266239093146
3class youth -->Dead相对支持度: 0.20934316824727783置信度: 0.7760416666666666lift: 1.3031643081761006
3class noParch -->Dead相对支持度: 0.2922374429223744置信度: 0.7878787878787878lift: 1.3230417381360777
3class S -->Dead相对支持度: 0.3217421847558834置信度: 0.7896551724137931lift: 1.326024723487313
3class noSibsp -->Dead相对支持度: 0.2655426765015806置信度: 0.7590361445783133lift: 1.2746078654239599
male youth -->Dead相对支持度: 0.2852125043905866置信度: 0.8152610441767069lift: 1.3690232628627719
male noParch -->Dead相对支持度: 0.42430628731998593置信度: 0.8206521739130435lift: 1.378076292042658
male S -->Dead相对支持度: 0.4214963119072708置信度: 0.8152173913043478lift: 1.3689499589827727
male noSibsp -->Dead相对支持度: 0.37513171759747105置信度: 0.8165137614678899lift: 1.3711268824649472
youth noParch -->Dead相对支持度: 0.2852125043905866置信度: 0.6403785488958991lift: 1.075352657579906
youth S -->Dead相对支持度: 0.27256761503336846置信度: 0.6381578947368421lift: 1.0716236345580934
youth noSibsp -->Dead相对支持度: 0.24165788549350192置信度: 0.6323529411764706lift: 1.0618756936736957
noParch S -->Dead相对支持度: 0.40042149631190727置信度: 0.6985294117647058lift: 1.1730022197558267
noParch noSibsp -->Dead相对支持度: 0.384966631541974置信度: 0.681592039800995lift: 1.1445602177790293
S noSibsp -->Dead相对支持度: 0.3470319634703196置信度: 0.6693766937669376lift: 1.1240476555708954
3class male noParch -->Dead相对支持度: 0.25430277485072006置信度: 0.8619047619047621lift: 1.4473495058400718
3class male S -->Dead相对支持度: 0.2585177379697927置信度: 0.8598130841121495lift: 1.4438370657732322
3class male noSibsp -->Dead相对支持度: 0.23041798384264137置信度: 0.8497409326424871lift: 1.42692345292795
3class noParch S -->Dead相对支持度: 0.2514927994380049置信度: 0.8136363636363636lift: 1.3662950257289879
3class noParch noSibsp -->Dead相对支持度: 0.25008781173164735置信度: 0.7911111111111111lift: 1.328469601677149
3class S noSibsp -->Dead相对支持度: 0.2247980330172111置信度: 0.7881773399014779lift: 1.3235430802119157
male youth noParch -->Dead相对支持度: 0.2585177379697927置信度: 0.8141592920353983lift: 1.3671731507764235
male youth S -->Dead相对支持度: 0.2374429223744292置信度: 0.8407960199004975lift: 1.4119027503989485
male youth noSibsp -->Dead相对支持度: 0.2205830698981384置信度: 0.8092783505154639lift: 1.358976852752383
male noParch S -->Dead相对支持度: 0.3624968282402529置信度: 0.8431372549019608lift: 1.4158342582315946
male noParch noSibsp -->Dead相对支持度: 0.35546188970846504置信度: 0.8322368421052632lift: 1.3975297914597815
```

图 5 lift 值

A)按照支持度排序

```
@Override
public int compareTo(Item o) {
    //按照支持度排序
    // return -(this.getSupport().compareTo(o.getSupport()));
    //按照置信度排序
    return -(this.getConfidence().compareTo(o.getConfidence()));
    //按照 lift 值排序
    // return -(this.getLift().compareTo(o.getLift()));
}
```

*****关联规则排序评价结果*****

```
male-->Dead相对支持度: 0.505795574288725置信度: 0.7947019867549668lift: 1.3344995626640008
S-->Dead相对支持度: 0.495960660344222置信度: 0.6371841155234657lift: 1.069988420407329
noParch-->Dead相对支持度: 0.47067088162978576置信度: 0.6454720616570327lift: 1.0839059148580361
male noParch-->Dead相对支持度: 0.42430628731998593置信度: 0.8206521739130435lift: 1.378076292042658
male S-->Dead相对支持度: 0.4214963119072708置信度: 0.8152173913043478lift: 1.3689499589827727
noSibsp-->Dead相对支持度: 0.4158763610818405置信度: 0.6311300639658849lift: 1.0598221828861085
noParch S-->Dead相对支持度: 0.40042149631190727置信度: 0.6985294117647058lift: 1.1730022197558267
noParch noSibsp-->Dead相对支持度: 0.384966631541974置信度: 0.6815920398009951lift: 1.1445602177790293
3class-->Dead相对支持度: 0.3793468807165437置信度: 0.7605633802816901lift: 1.2771724687749135
male noSibsp-->Dead相对支持度: 0.37513171759747105置信度: 0.8165137614678899lift: 1.3711268824649472
成功构造 (总时间: 0 秒)
```

可以看到，其中男性出现的频率较高，并且有较大的数据支持量。

B)按照置信度排序

接下来，按照置信度对这些规则进行排序，获取到推测更为准确的规则。

*****关联规则排序评价结果*****

```
3class male noParch S-->Dead相对支持度: 0.2205830698981384置信度: 0.8722222222222222lift: 1.4646750524109013
3class male noParch-->Dead相对支持度: 0.25430277485072006置信度: 0.8619047619047621lift: 1.4473495058400718
3class male noParch noSibsp-->Dead相对支持度: 0.22620302072356868置信度: 0.8609625668449198lift: 1.4457673292301483
3class male S-->Dead相对支持度: 0.25851773796979277置信度: 0.8598130841121495lift: 1.4438370657732322
3class male-->Dead相对支持度: 0.302072356868774置信度: 0.8498023715415021lift: 1.4270266239093146
3class male noSibsp-->Dead相对支持度: 0.23041798384264137置信度: 0.8497409326424871lift: 1.42692345292795
male noParch S noSibsp-->Dead相对支持度: 0.3062873199859501置信度: 0.8482490272373541lift: 1.42441817781367
male noParch S-->Dead相对支持度: 0.3624868282402529置信度: 0.8431372549019608lift: 1.4158342582315946
male youth S-->Dead相对支持度: 0.2374429223744292置信度: 0.8407960199004975lift: 1.4119027503989485
male S noSibsp-->Dead相对支持度: 0.3175272216368107置信度: 0.8401486988847584lift: 1.4108157396366698
成功构造 (总时间: 0 秒)
```

图 6 选取置信度最高的前十条“死亡”规则

C)按照提升值排序

按照提升值对这些规则进行排序，可以得到质量较高的规则。


```

*****关联规则排序评价结果*****
3class male noParch S-->Dead相对支持度: 0.2205830698981384置信度: 0.8722222222222222lift: 1.4646750524109013
3class male noParch-->Dead相对支持度: 0.25430277485072006置信度: 0.8619047619047621lift: 1.4473495058400718
3class male noParch noSibsp-->Dead相对支持度: 0.22620302072356868置信度: 0.86096256684491981lift: 1.4457673292301483
3class male S-->Dead相对支持度: 0.25851773796979277置信度: 0.85981308411214951lift: 1.4438370657732322
3class male-->Dead相对支持度: 0.3020723568668774置信度: 0.8498023715415021lift: 1.4270266239093146
3class male noSibsp-->Dead相对支持度: 0.23041798384264137置信度: 0.8497409326424871lift: 1.42692345292795
male noParch S noSibsp-->Dead相对支持度: 0.3062873199859501置信度: 0.8492490272373541lift: 1.42441817781367
male noParch S-->Dead相对支持度: 0.3624868282402529置信度: 0.84313725490196081lift: 1.4158342582315946
male youth S-->Dead相对支持度: 0.2374429223744292置信度: 0.84079601990049751lift: 1.4119027503989485
male S noSibsp-->Dead相对支持度: 0.3175272216368107置信度: 0.84014869888475841lift: 1.4108157396366698
成功构造 (总时间: 0 秒)

```

图 7 选取置信度最高的前十条“死亡”规则

我们继续分析，可以看出，几个关键字眼出现了规则中 noParch（没有父母或孩子），noSibSp（没有兄弟姐妹），男性，以及 Southampton 登船口。

八、 可视化

由于 matlab 和 java 中没有可视化的包，所以采用 R 实现，使用扩展包 arulesViz。

```

#加载算法库
library("Matrix")
library("arules")
library("arulesViz")
#*****
#筛选右变元为死亡的规则子集
x=subset(rules,subset=rhs%in%"Dead")
x
inspect(x)
#画散点图
plot(x)
plot(x, measure = c("support", "lift"), shading = "confidence")
#画泡泡图
plot(rules, method = "grouped")
#画平行坐标图
plot(x, method="paracord", control=list(reorder=TRUE))

```

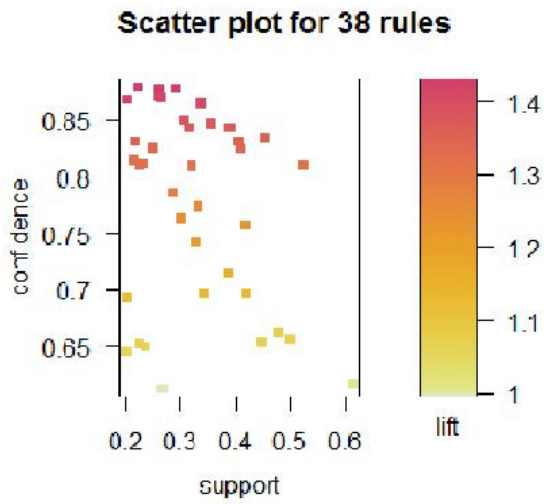


图 8 散点图

从图中可以看出，高 lift 对应低 support。

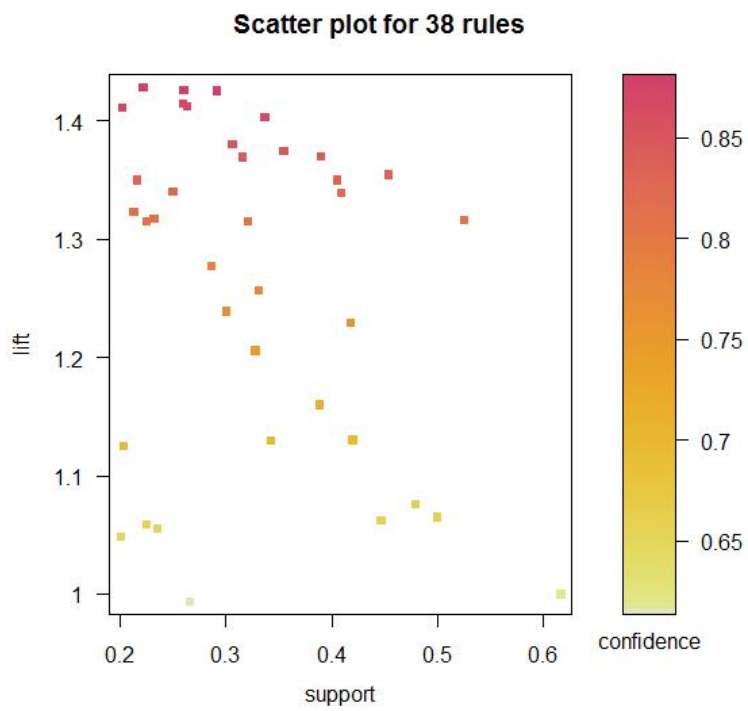


图 9 散点图

从图中看出，很多规则都有高 lift。

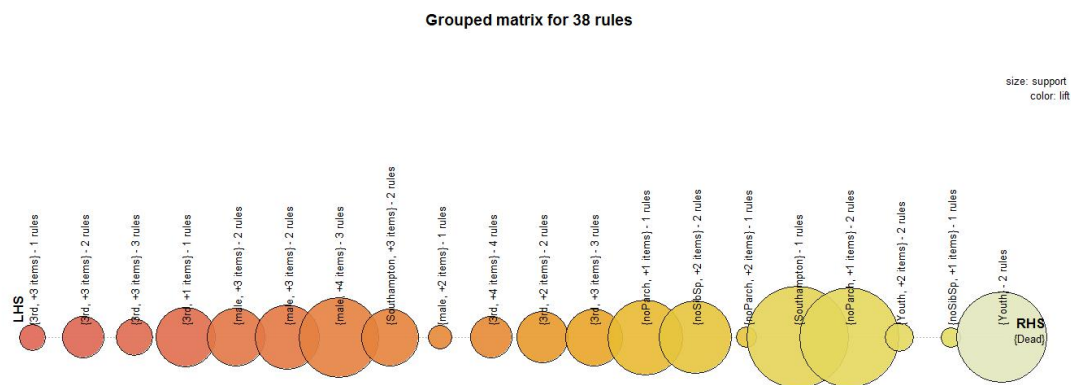


图 10 泡泡图

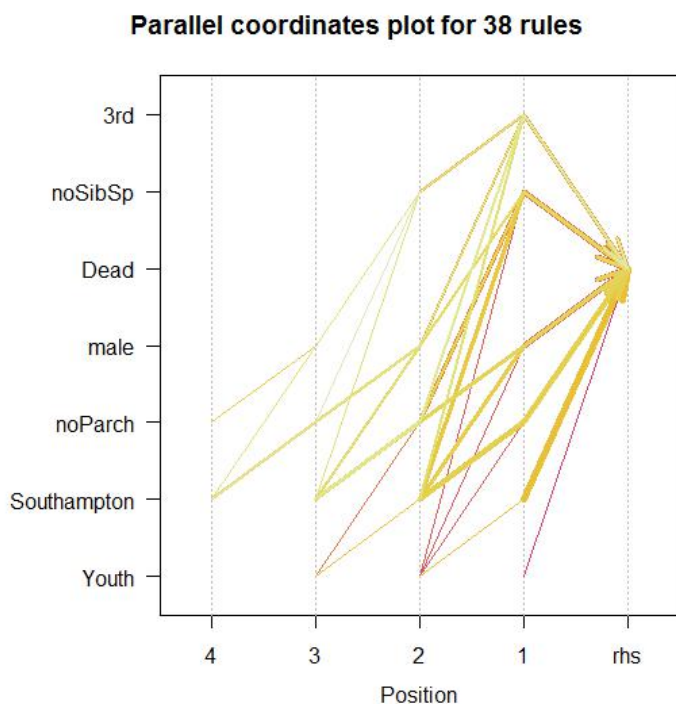


图 11 平行坐标图

九、 结论

经过以上关联规则抽取，我们得到初步结论，在泰坦尼克号遭遇不测时，船上优先考虑了女士的逃生路线。对于船内设计，三等座因其价格较低，在船舱的位置更不利于逃生。同时，推测最先的进水口应在 Southampton 口附近，导致转移时间较短，附近的人更容易死亡。在灾害发生时，无亲无故的人，求生欲望更弱，死亡率也会更高一些。