

《数据挖掘与机器学习》实验指导书

2020年9月1日

计算机学院

前言

随着数据库技术的发展，特别是数据仓库以及Web等新型数据源的日益普及，形成了数据丰富，知识缺乏的严重局面。针对如何有效地利用这些海量的数据信息的挑战，数据挖掘技术应运而生，并显示出强大的生命力。数据挖掘技术使数据处理技术进入了一个更高级的阶段，是对未来人类产生重大影响的十大新兴技术之一。因此加强数据挖掘领域的理论与实践学习也已成为专业学生的必修内容。

本实验指导书通过大量的实例，循序渐进地引导学生做好各章的实验。根据实验教学大纲，我们编排了7个实验，每个实验又分了五部分内容：实验目的、实验内容、实验步骤、实验报告要求、注意事项。在实验之前，由教师对实验作一定的讲解后，让学生明确实验目的，并对实验作好预习工作。在实验中，学生根据实验指导中的内容进行验证与总结，然后再去完成实验步骤中安排的任务。实验完成后，学生按要求完成实验报告。整个教学和实验中，强调学生切实培养动手实践能力，掌握数据挖掘与机器学习的基本方法。

目录

选作实验 利用 Weka 软件进行数据挖掘.....	4
实验一 数据预处理实验	25
实验二 Apriori 算法实现.....	27
实验三 K-Means 聚类算法实现.....	30
实验四 DBSACN 聚类算法实现.....	33
实验五 ID3 算法实现.....	34
实验六 贝叶斯算法实现.....	39

选作实验 利用 Weka 软件进行数据挖掘

一、实验目的

通过使用weka软件对数据进行预处理，体会理论课堂中讲解的数据预处理、关联规则、分类和聚类的经典算法的实际使用。并通过对样本数据的数据挖掘过程，加深对该数据挖掘算法的理解与应用过程。

实验类型：**验证**

计划课时：**4学时**

二、实验内容

- 1、weka安装和配置；
- 2、关联规则算法使用和比较；
- 3、聚类的使用和比较；
- 4、分类算法的使用和比较；

三、实验方法及步骤

3.1、Weka 简介

Weka 的全名是怀卡托智能分析环境（Waikato Environment for Knowledge



Analysis)，是一款免费的，非商业化（与之对应的是 SPSS 公司商业数据挖掘产品——Clementine）的，基于 JAVA 环境下开源的机器学习（machine learning）以及数据挖掘（data mining）软件。它和它的源代码可在其官方网站下载。有趣的是，该软件的缩写 WEKA 也是 New Zealand 独有的一种鸟名，而 Weka 的主要开发者同时恰好来自 New Zealand 的 the University of Waikato。

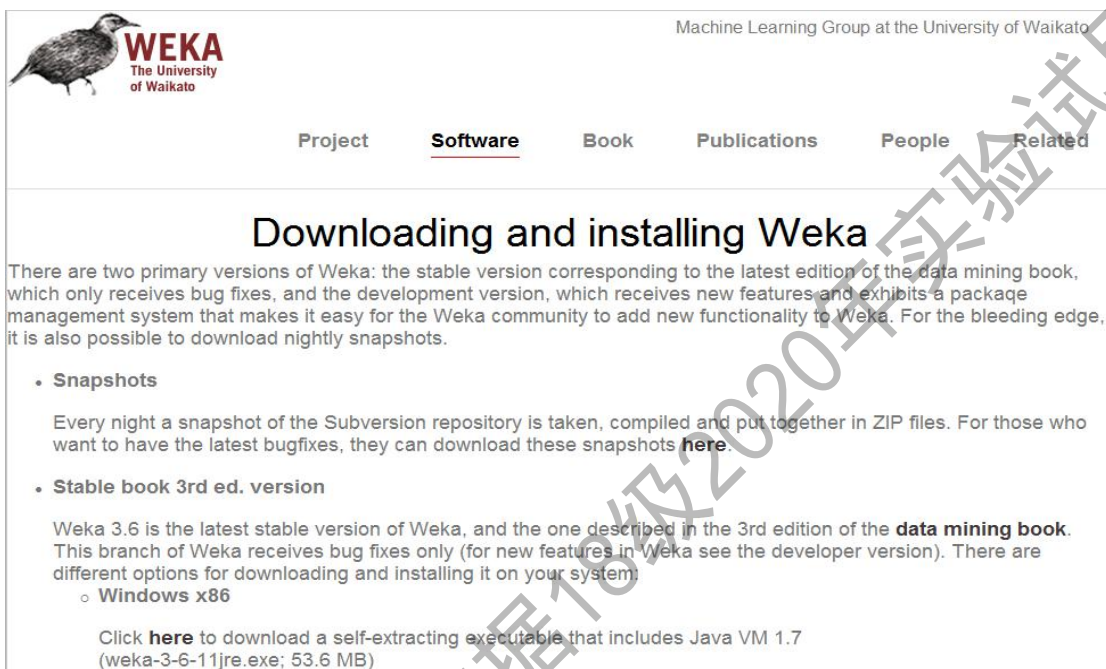
Weka 作为一个公开的数据挖掘工作平台，集合了大量能承担数据挖掘任务的机器学习算法，包括对数据进行预处理，分类，回归、聚类、关联规则以及在新的交互式界面上的可视化。

如果想自己实现数据挖掘算法的话，可以参考 Weka 的接口文档。在 Weka 中集

成自己的算法甚至借鉴它的方法自己实现可视化工具并不是件很困难的事情。

3.2 Weka 的下载安装

<http://www.cs.waikato.ac.nz/ml/weka/downloading.html> 为 Weka 的官方网站，其中有对应的相关信息。点击 Software，可以选择自己计算机对应的软件安装包。



Machine Learning Group at the University of Waikato

WEKA
The University of Waikato

Project **Software** Book Publications People Related

Downloading and installing Weka

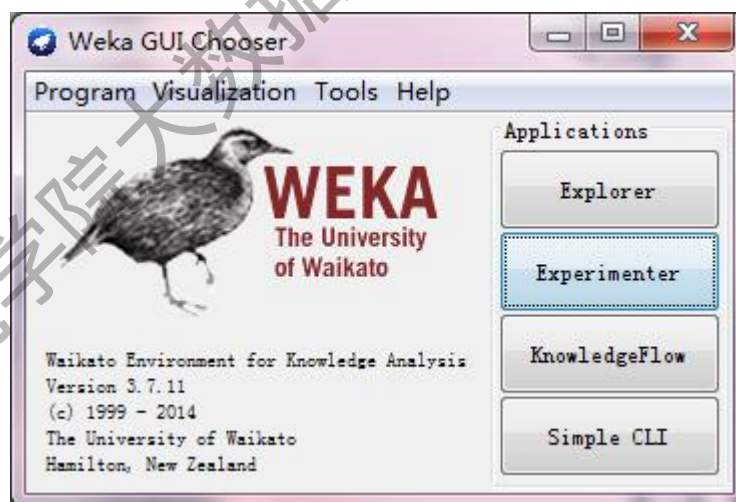
There are two primary versions of Weka: the stable version corresponding to the latest edition of the data mining book, which only receives bug fixes, and the development version, which receives new features and exhibits a package management system that makes it easy for the Weka community to add new functionality to Weka. For the bleeding edge, it is also possible to download nightly snapshots.

- **Snapshots**
Every night a snapshot of the Subversion repository is taken, compiled and put together in ZIP files. For those who want to have the latest bugfixes, they can download these snapshots [here](#).
- **Stable book 3rd ed. version**
Weka 3.6 is the latest stable version of Weka, and the one described in the 3rd edition of the **data mining book**. This branch of Weka receives bug fixes only (for new features in Weka see the developer version). There are different options for downloading and installing it on your system:
 - **Windows x86**
Click [here](#) to download a self-extracting executable that includes Java VM 1.7 (weka-3-6-11jre.exe; 53.6 MB)



依次执行安装步骤，完成安装环节。

3.3 Weka 的界面



以上为 Weka 运行主界面，分别包含四项应用：Explorer，Experimenter，KnowledgeFlow，Simple CLI。

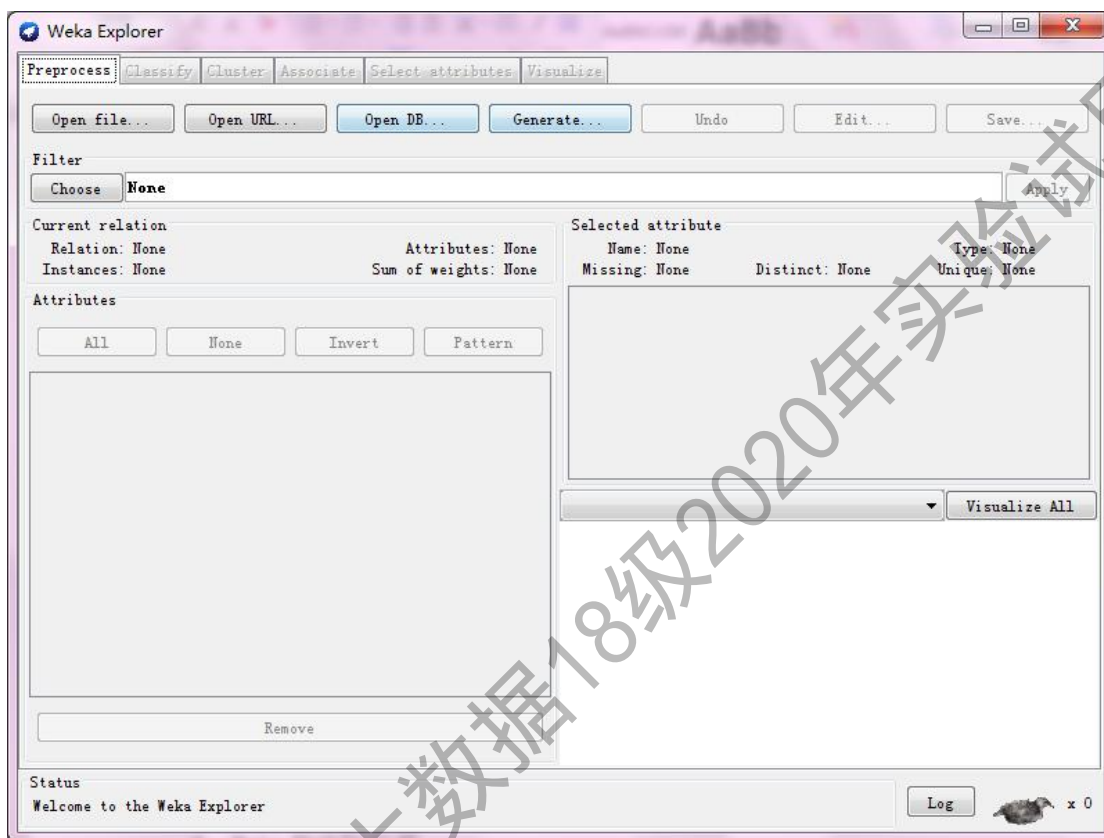
(1) Explorer：使用 Weka 探索数据的环境，包括获取关联项，分类预测，聚类等；

(2) Experimenter：运行算法试验、管理算法方案之间的统计检验的环境；

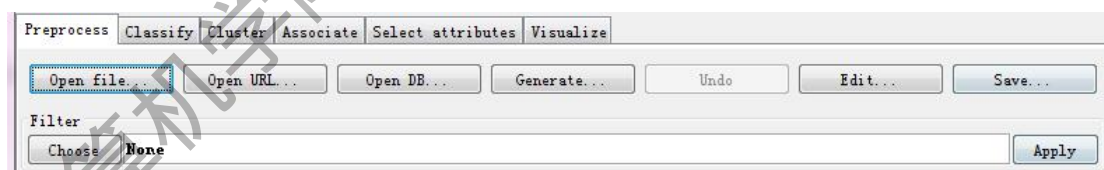
(3) KnowledgeFlow: 这个环境本质上和 Explorer 所支持的功能是一样的,但是它有一个可以拖放的界面。它有一个优势,就是支持增量学习;

(4) SimpleCLI: 提供了一个简单的命令行界面,从而可以在没有自带命令行的操作系统中直接执行 Weka 命令(某些情况下使用命令行功能更好一些)。

3.4 Explorer 数据探索环境的使用



该图为点击 Explorer 进入的界面



该图为 Explorer 界面内的标题栏,依次作用如下:

(1) Preprocess: 数据预处理

Classify: 分类

Cluster: 聚类

Associate: 关联规则

Select Attributes: 属性选择

Visualize: 图形可视化

(2) Open file : 用于打开执行文件(csv 或者 arff 格式)

Open URL: 用于打开网站

Open DB: 用于打开数据库

Generate: 从数据生成器中生成一些人造数据

Undo: 撤消上一步操作

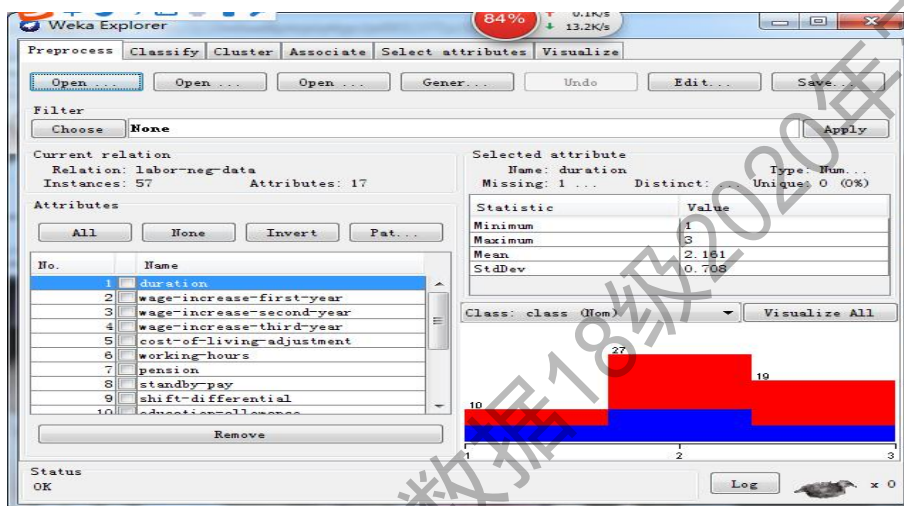
Edit: 编辑

Save: 对分析结果进行保存

(3) Choose: 预处理阶段使用 Filter 筛选器可以对数据进行预处理, 点击 Apply, 即对预处理结果进行保存

使用 weka 数据预处理:

打开 Explorer 界面, 点 “open file”, 在 weka 安装目录下, 选择 data 目录里的 “labor.arff” 文件, 将会看到如下的界面。我们将整个区域分为 7 部分, 下面将分别介绍每部分的功能、



最上面的一行, 一共有 6 个选项, 可以选择不同的数据挖掘功能, 依次是 Preprocess (预处理)、Classify (分类)、Cluster (聚类)、Associate (关联规则)、Select attribute (特征选择) 和 Visualize (可视化)。

第二行, 提供了打开、保存, 编辑文件的功能。打开文件不仅仅可以直接从本地选择, 还可以使用 url 和 db 来做数据源。Generate 按钮提供了数据生成的功能, weka 提供了几种生成数据的方法。点开 Edit, 将看到如下界面

No.	duration Numeric	wage-increase-first-year Numeric	wage-increase-second-year Numeric	wage
1	1.0	5.0		
2	2.0	4.5	5.8	
3				
4	3.0	3.7	4.0	
5	3.0	4.5	4.5	
6	2.0	2.0	2.5	
7	3.0	4.0	5.0	
8	3.0	6.9	4.8	
9	2.0	3.0	7.0	
10	1.0	5.7		
11	3.0	3.5	4.0	
12	2.0	6.4	6.4	
13	2.0	3.5	4.0	
14	3.0	3.5	4.0	
15	1.0	3.0		
16	2.0	4.5	4.0	
17	1.0	2.8		
18	1.0	2.1		
19	1.0	2.0		
20	2.0	4.0	5.0	
21	2.0	4.3	4.4	
22	2.0	2.5	3.0	
23	3.0	3.5	4.0	

在这个界面，可以看到各行各列对应的值，右键每一列的名字(先点击列名)，可以看到一些编辑数据的功能，这些功能还是比较实用的。

第3行，Filter 针对特征 (attribute) 和样本 (instance) 提供了大量的操作方法，功能十分强大。第4行，可以看到当前的特征、样本信息，并提供了特征选择和删除的功能。在4行中用鼠标选择单个特征后第5行将显示该特征的信息。包括最小值、最大值、期望和标准差。第六行，提供了可视化功能，选择特征后，该区域将显示特征值在各个区间的分布情况，不同的类别标签以不同的颜色显示。第七行，是状态栏，没有任务时，小鸟是坐着的，任务运行时，小鸟会站起来左右摇摆。如果小鸟站着但不转动，表示任务出了问题。

3.5 Weka 实现基本的数据挖掘

(1) 关联规则 (购物篮分析)

关联规则与相关概念

◆关联规则

关联规则是通过分析数据信息的相关性进行知识获取的方法。

关联规则反映了一个事物与其它事物之间的相互依存性和关联性。如果两个或者多个事物之间存在一定的关联关系，那么，其中一个事物的相关资料就能够通过与其相关联事物的信息

进行预测分析。

关联规则的形式如“在购买面包顾客中,有 70%的人同时也买了黄油”,可以表示成: 面包 \rightarrow 黄油,也就是说面包与黄油关联。

那么我们如何发现这些关联规则呢?

对超市中的货篮数据 (Market-Basket Data) 进行分析。具体为:零售部门可以利用前端收款机收集存储大量的售货数据,发现顾客放入货篮中的不同商品之间的关系 (指普遍存在的具有一定规律的关系,例如卓越购物网会提示你购买某商品的顾客同时也购买了什么商品就是关联挖掘的结果),则可获取顾客的购买行为习惯(Profile)等极有价值的关联规则(知识),并指导部门制定相应的经销策略。

例如,可以帮助如何摆放货架上的商品(如把顾客经常同时买的商品放在一起),帮助如何规划市场(怎样相互搭配进货)。

用于关联规则发现的主要对象是针对事务型数据库中的数据,如:购物事务数据库中的货篮数据。一个购物事务一般由如下几个部分组成:购物事务发生的时间(圣诞节的巧克力、情人节的玫瑰等),顾客购买的物品名称,顾客标识号(如信用卡号、积分卡号等信息)。

IBM 公司 Almaden 研究中心的 R.Agrawal 首先提出了关联规则模型,并给出求解算法 AIS。随后又出现了 SETM 和 Apriori 等算法。其中, Apriori 是关联规则模型中的经典算法。

◆关联规则的相关概念

支持度 (统计概率)

1000 个顾客购物,其中 200 个顾客购买了面包(物品集 A),面包 (物品集 A) 的支持度为: $P(A)=20\% (200/1000)$ 。

总结:物品集(项集)A 的支持度:如果物品集 A 的支持度为 s%,则 $P(A)$ 称物品集 A 在事务数据集 W 中具有大小为 s%的支持度。

1000 个顾客购物,100 个顾客购买了面包和黄油,则我们认为蕴涵关系:面包 \rightarrow 黄油的支持度为: $P(A \cup B)=10\% (100/1000)$

总结:蕴涵关系 $A \rightarrow B$ 的支持度:如果物品集(A+B)的支持度为 s%,则 $P(A \cup B)$ 表示蕴涵关系 $A \rightarrow B$ 在事务数据库 W 中具有大小为 s%的支持度。

可信度 (条件概率)

实际上就是表示在购买物品 A 的前提下又购买物品 B 的概率,求蕴涵关系 $A \rightarrow B$ 的可信度就是求: $P(B|A)$

例如:1000 个顾客购物,200 个顾客购买了面包,其中 140 个买了黄油,则面包 \rightarrow 黄油的可信度是:

$$P(B|A) = P(A \cup B) / P(A) = 140/1000 / 200/1000 = 70\%。$$

若项集 A 的支持度记为 support(A),可信度记为 confidence(A)

则: $\text{support}(A \rightarrow B) = P(A \cup B)$ ($A \rightarrow B$ 的支持度)

$\text{confidence}(A \rightarrow B) = P(B|A) = P(A \cup B) / P(A)$

$= \text{support}(A \rightarrow B) / \text{support}(A)$ ($A \rightarrow B$ 的可信度)

最小支持度 minsup 的定义: 用户规定的关联规则必须满足的最小支持度。

最小可信度 minconf 的定义: 用户规定的关联规则必须满足的最小可信度。

关联规则就是支持度和信任度分别满足用户给定阈值的那些蕴涵关系($A \rightarrow B$)。

关联规则基本模型

大项集(频繁项集大物品集 large item set)的定义: 支持度 \geq 最小支持度 minsup 的物品集发现关联规则需要经历如下两个步骤:

1 找出所有频繁项集或大项集 (也就是满足最小支持度的项集)。

2 由频繁项集或大项集生成满足最小信任度阈值的关联规则。

由于可信度为: $\text{support}(A \cup B) / \text{support}(A)$ 显然第二步不是难题,目前大多数研究集中在第一个问题上,即如何高效地求出大项集。

◆思路的正确性

大项集的特点:

显然大项集具有向下封闭性,即大项集 A 的任意子集一定也是大项集(如: {牛奶、面包、黄油}是大项集,显然{牛奶、黄油}、{面包、牛奶}、{黄油、面包}都是大项集)。

反过来说,如果 A 有一子集不是大项集,则 A 肯定不是大项集。(如: {牛奶、面包、黄油、袜子}中,显然{牛奶、袜子} 不是大项集,所以{牛奶、面包、黄油、袜子} 不是大项集)。

利用大项集的特点求大项集的方法如下:

◆求大项集的基本思路

1、首先找寻长度为 1 的大项集 $L[1]$ (即单个物品), 记为 $L[1]$ (通过看它是否大于最小支持度来判断);

2、在 $L[k]$ 的基础上生成候选物品集 $C[k+1]$, 候选物品集 $C[k+1]$ 必须保证包括所有的 $L[k]$ 大项集

3、用事务数据库 D 中的事务对所有 $C[k+1]$ 进行支持度测试以找寻长度为 $k+1$ 的大项集 $L[k+1]$, 计算每个候选物品集的支持度, 如果大于 minsup , 则加入到 $L[k+1]$

4、如果 $L[k+1]$ 为空集, 则结束, $L[1] \cup L[2] \cup \dots$ 即为结果; 否则转(2), 继续。

◆Apriori 算法—实例:

假设: $\text{minsup}=2$, 求所有大项集或频繁项集

顾客号 购买物品

A 1, 3, 4
B 2, 3, 5
C 1, 2, 3, 5
D 2, 5

顾客号 购买单个物品集

A {1}, {3}, {4} {支持度 1/4}
B {2}, {3}, {5}
C {1}, {2}, {3}, {5}
D {2}, {5}

$C[k]$ 必须保证联合(Join)所有的 $L[k-1]$ 的大项集

L1	support	C2	support
{1}	2	{1, 2}	1
{2}	3	{1, 3}	2
{3}	3	{1, 5}	1
{5}	3	{2, 3}	2
		{2, 5}	3
		{3, 5}	2

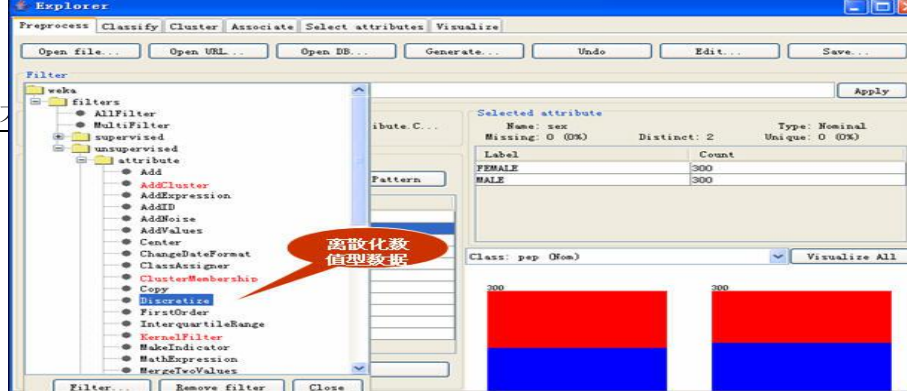
从 $C[k]$ 中除去(Prune)大小为 $k-1$ 且不在 $L[k-1]$ 中的子集, 找出支持度大于 $\text{minsup}=2$ 的 $L[k]$, 当然在本例的 $C2$ 中没有这样的不在 $L1$ 中的子集, 但支持度有小于 $\text{minsup}=2$ 的集合: {1, 2} 和 {1, 5}

先通过 join 所有的 $L[k-1]$ 大项集找出 $C[k]$, 然后再从 $C[k]$ 中除去大小为 $k-1$ 且不在 $L[k-1]$ 中的子集, 找出支持度大于 $\text{minsup}=2$ 的 $L[k]$

L2	support	C3	support	事务号	购买物品号
{1, 3}	2	{1, 2, 3}	1	1	1, 3, 4
{2, 3}	2	{1, 2, 5}	1	2	2, 3, 5
{2, 5}	3	{1, 3, 5}	1	3	1, 2, 3, 5
{3, 5}	2	{2, 3, 5}	2	4	2, 5

L3	support	L4	support
{2, 3, 5}	2	空集	

最大项集为: $L1 \cup L2 \cup L3$



离散化数值型数据

```

{1}      {5}      {1, 3}
{2}      {2, 3}    {2, 5}
{3}      {3, 5}    {2, 3, 5}
(1) L[1]={large 1-itemsets};
(2) for (k=2; L[k-1]不为空; k++) do begin
(3) C[k]=apriori-gen(L[k-1]); // 新候选物品集
(4) For all transactions t∈D do begin
(5) C=subset(C[k],t); // t 中的候选物品集
(6) For all candidates c∈C do
(7) c.count++;
(8) end;
(9) L[k]={c∈C[k]|c.count>=minsup};
(10) end;
(11) Answer = L[1] ∪ L[2] ∪ ...
apriori-gen(L[k-1]) 分成两步:
join 算法: 从任意两个 L[k-1]最大项集生成候选物品集 C[k]
insert into C[k]
select p.item1,p.item2,...,p.item(k-1), q.item(k-1)
from L[k-1] p, L[k-1] q
where p.item1=q.item1, ...,p.item(k-2)=q.item(k-2), p.item(k-1)小于 q.item(k-1)

```

Prune 算法: 从 C[k]中除去大小为 k-1 且不在 L[k-1]中的子集

```

(1) For all itemsets c∈C[k] do
(2) For all (k-1)-subsets s of c do
(3) if (s∉L[k-1])
(4) then delete c from C[k]

```

关联挖掘

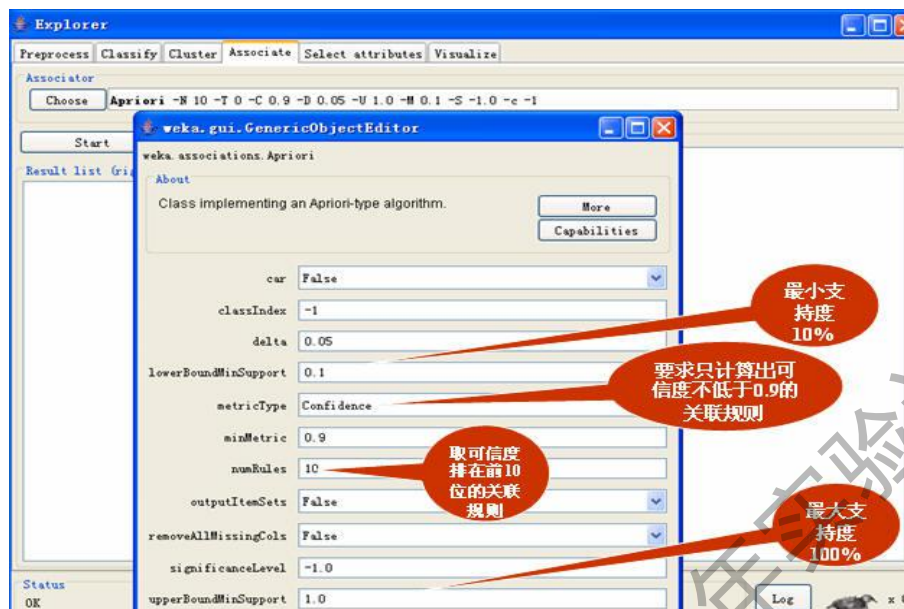
◆bank-data 实例关联挖掘

注意: 目前, WEKA 的关联规则分析功能仅能用来作示范, 不适合用来挖掘大型数据集。

我们打算对前面的“bank-data”数据作关联规则的分析。

用“Explorer”打开“bank-data-final.arff”后, 切换到“Associate”选项卡。默认关联规则分析是用 Apriori 算法。

点“Choose”右边的文本框修改默认的参数弹出的窗口中点“More”可以看到各参数的说明



首先我们来温习一下 Apriori 的有关知识。对于一条关联规则 $L \rightarrow R$ ，我们常用支持度（Support）和置信度（Confidence）来衡量它的重要性。规则的支持度是用来估计在一个购物篮中同时观察到 L 和 R 的概率 $P(L, R)$ ，而规则的置信度是估计购物篮中出现了 L 时也出现 R 的条件概率 $P(R|L)$ 。关联规则的目标一般是产生支持度和置信度都较高的规则。

有几个类似的度量代替置信度来衡量规则的关联程度，它们分别是

Lift（提升度）：是一种简单的相关度量，定义如下：项集 A 的出现独立于项集 B 的出现，如果 $P(A \cup B) = P(A)P(B)$ ，否则作为事件项集 A 和 B 是依赖的（dependent）和相关的（correlated）。

E. G:

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}$$

Lift < 1 时表示 A 的出现和 B 的出现是负相关（ A 发生时实际上减少了 B 的可能性）的。

Lift > 1 时表示 A 的出现和 B 的出现是正相关的，意味着一个的出现蕴涵另一个出现。

Lift = 1 时表示 L 和 R 独立。这个数越大，越表明 L 和 R 存在在一个购物篮中不是偶然现象。

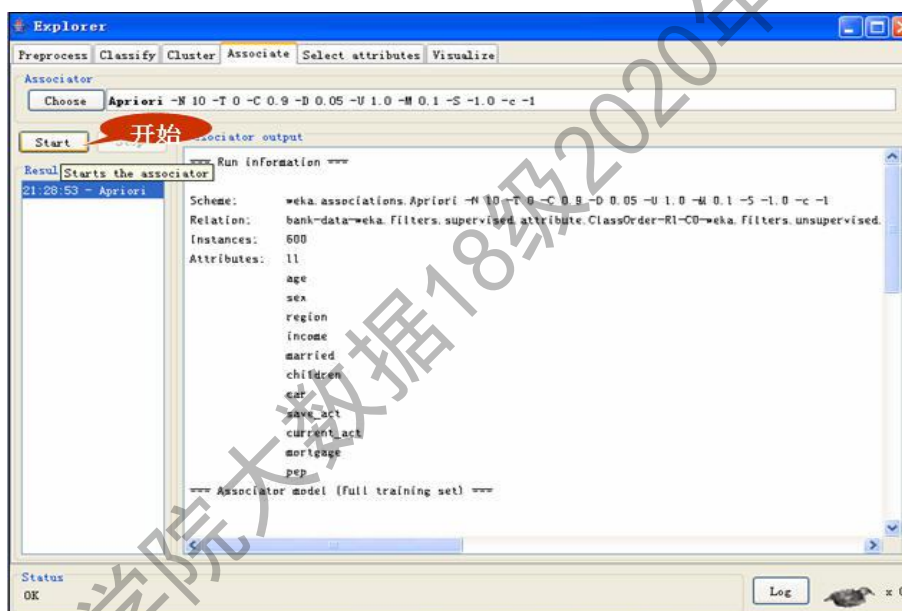
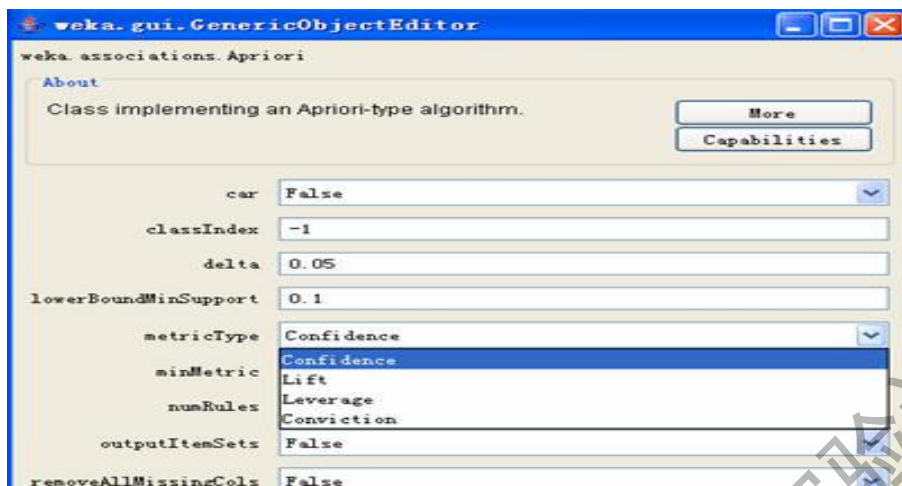
Leverage（不知道怎么翻译）： $P(L, R) - P(L)P(R)$

它和 Lift 的含义差不多。Leverage = 0 时 L 和 R 独立，Leverage 越大 L 和 R 的关系越密切。

Conviction（更不知道译了）： $P(L)P(!R)/P(L, !R)$ （ $!R$ 表示 R 没有发生）

Conviction 也是用来衡量 L 和 R 的独立性。从它和 lift 的关系（对 R 取反，代入 Lift 公式后求倒数）可以看出，我们也希望这个值越大越好。

值得注意的是，用 Lift 和 Leverage 作标准时， L 和 R 是对称的，Confidence 和 Conviction 则不然。



◆ 以 confidence 作为 minMetric, 分析挖掘出来的 confidence 排前 5 的规则。

Best rules found:

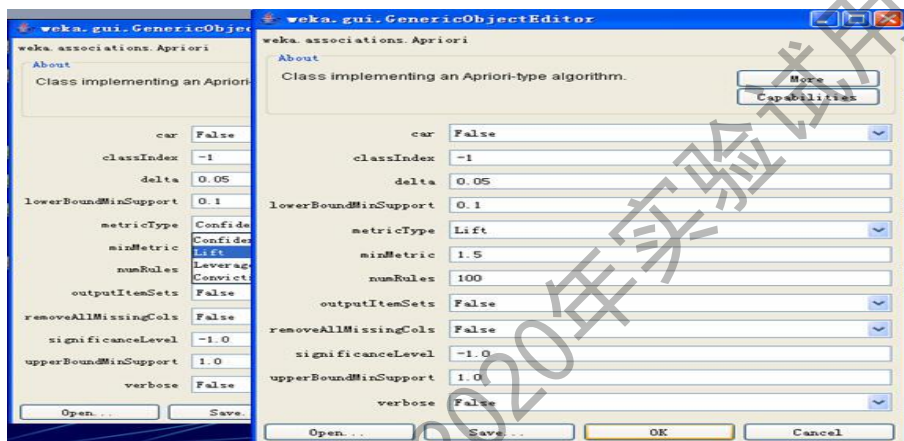
1. income=(43758.136667-inf)' 80 ==> save_act=YES 80 conf:(1)
2. age=(50.666667-inf)' income=(43758.136667-inf)' 76 ==> save_act=YES 76 conf:(1)
3. income=(43758.136667-inf)' current_act=YES 63 ==> save_act=YES 63 conf:(1)
4. age=(50.666667-inf)' income=(43758.136667-inf)' current_act=YES 61 ==>
save_act=YES 61 conf:(1)
5. income=(43758.136667-inf)' current_act=YES 63 ==> age=(50.666667-inf)' 61 conf:
(0.97)
6. income=(43758.136667-inf)' save_act=YES current_act=YES 63 ==> age=(50.666667-
inf)' 61 conf:(0.97)

◆ 下面分析挖掘出来的 lift 排前 5 的规则。

Best rules found:

1. income=(43758-max)' 80 ==> age=(51-max)' save_act=YES current_act=YES 61 conf:

- (0.76) < lift:(4.05)> lev:(0.08) [45] conv:(3.25)
2. age='(51-max)' save_act=YES current_act=YES 113 ==> income='(43758-max)' 61 conf:(0.54) < lift:(4.05)> lev:(0.08) [45] conv:(1.85)
3. age='(51-max)' save_act=YES 151 ==> income='(43758-max)' current_act=YES 61 conf:(0.4) < lift:(3.85)> lev:(0.08) [45] conv:(1.49)
4. income='(43758-max)' current_act=YES 63 ==> age='(51-max)' save_act=YES 61 conf:(0.97) < lift:(3.85)> lev:(0.08) [45] conv:(15.72)
5. income='(43758-max)' 80 ==> age='(51-max)' save_act=YES 76 conf:(0.95) < lift:(3.77)> lev:(0.09) [55] conv:(11.97)



命令行方式

我们也可以利用命令行来完成挖掘任务，在“Simple CLI”模块中输入如下格式的命令：

```
java weka.associations.Apriori options -t directory-path\bank-data-final.arff
```

即可完成 Apriori 算法。注意，“-t”参数后的文件路径中不能含有空格。

在前面我们使用的 option 为

-N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 命令行中使用这些参数得到的结果和前面利用 GUI 得到的一样。

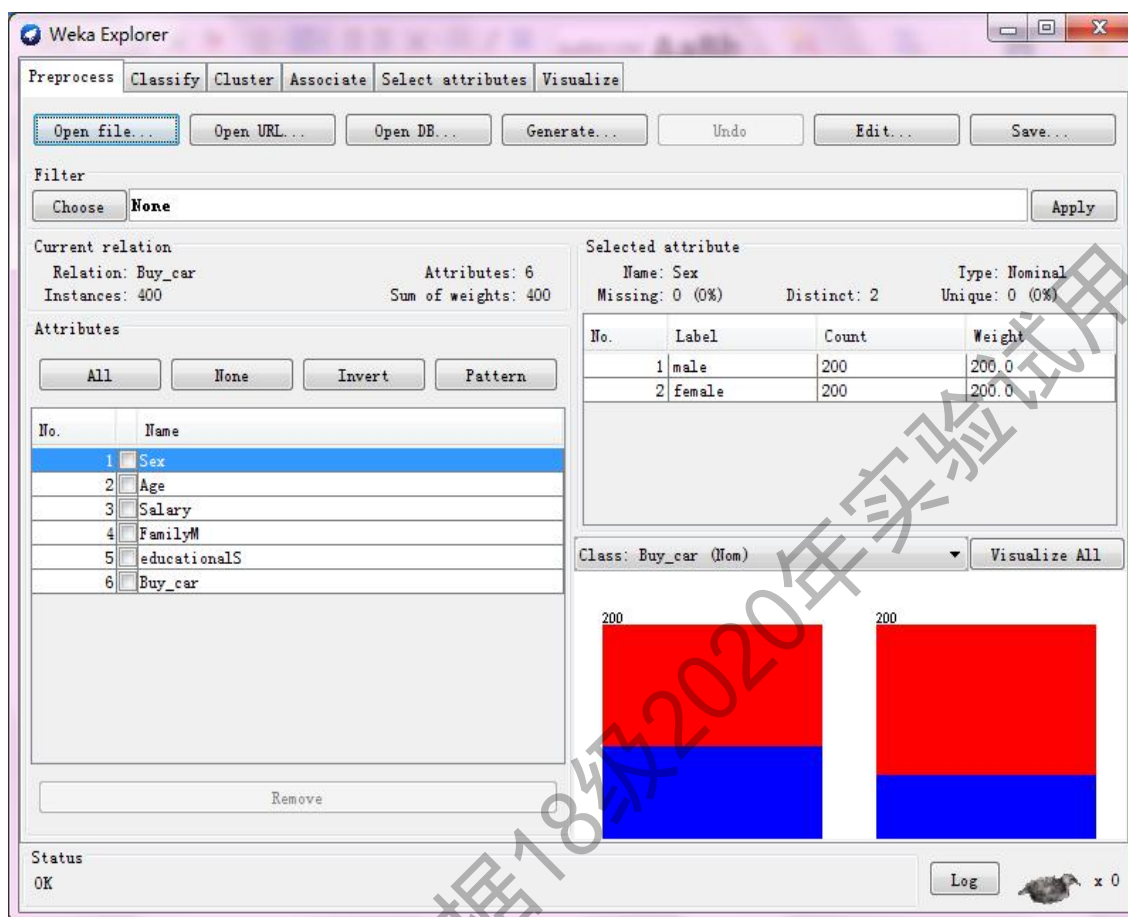
我们还可以加上“-I”参数，得到不同项数的频繁项集。我用的命令如下：

```
java weka.associations.Apriori -N 100 -T 1 -C 1.5 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -I -t d:\weka\bank-data-final.arff
```

挖掘结果在上方显示，应是这个文件的样子。

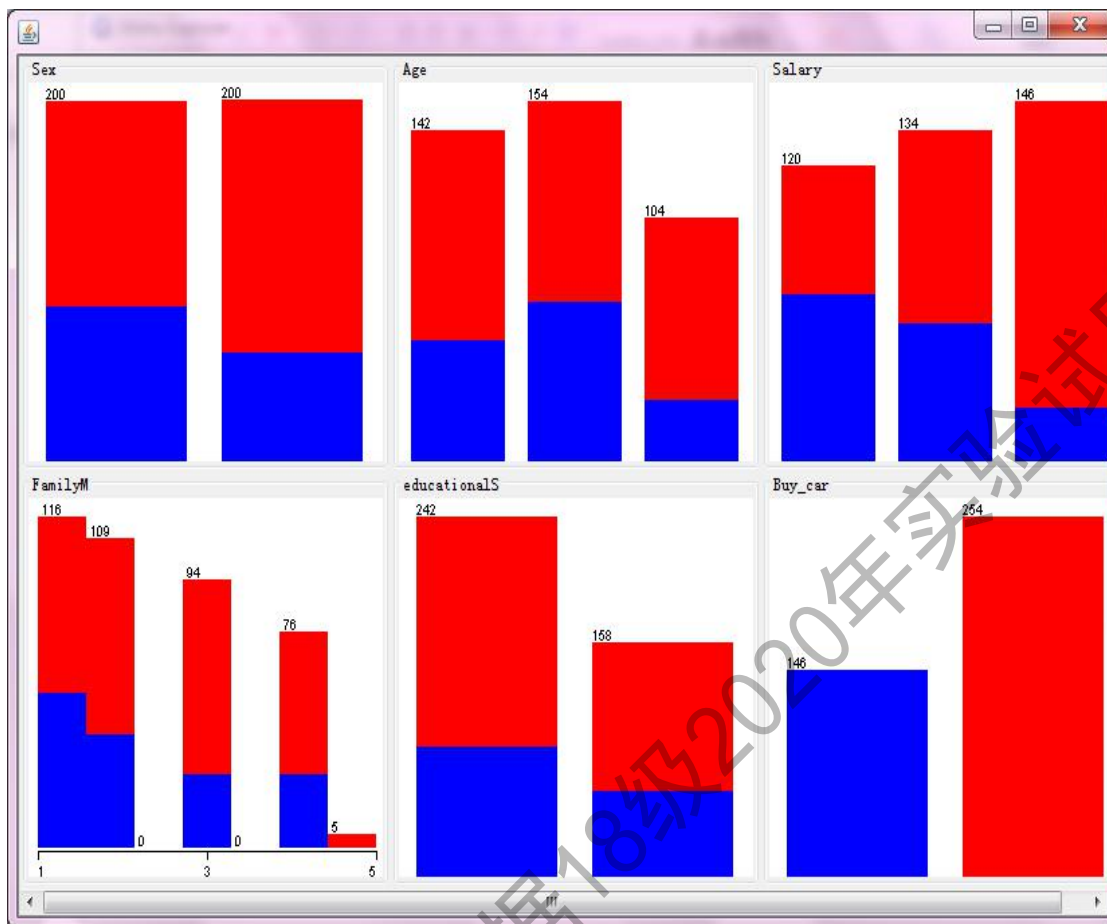
(2) 用于分类——J48 算法

导入数据源，该数据源如图所示：



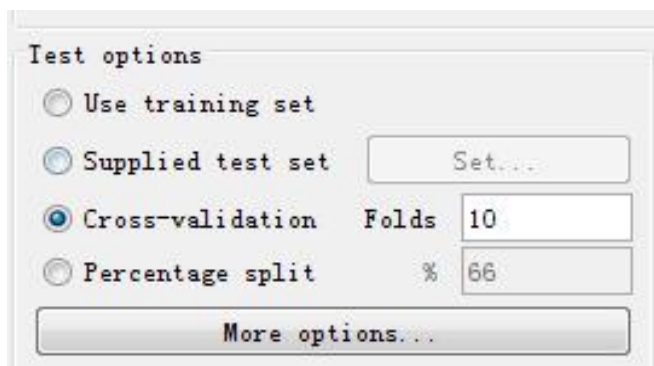
该数据源分为：Sex、Age、Salary、FamilyM、EducationS、Buy_car。
 分别表示性别，年龄，工资，家庭成员数，教育情况，是否买车。
 Sex: female 为女士，male 为男士。
 Age: A1 代表 20 岁以下，A2 代表 20 到 50 岁、A3 代表 50 岁以上。
 Salary: S1 代表工资高，S2 代表工资中等、S3 代表工资低。
 FamilyM: 含有几个家庭成员。
 EducationS: E1 表示受过低等教育，E2 表示受过中等教育、E3 表示受过高等教育。
 Buy car: YES 表示买车，NO 表示不买车。

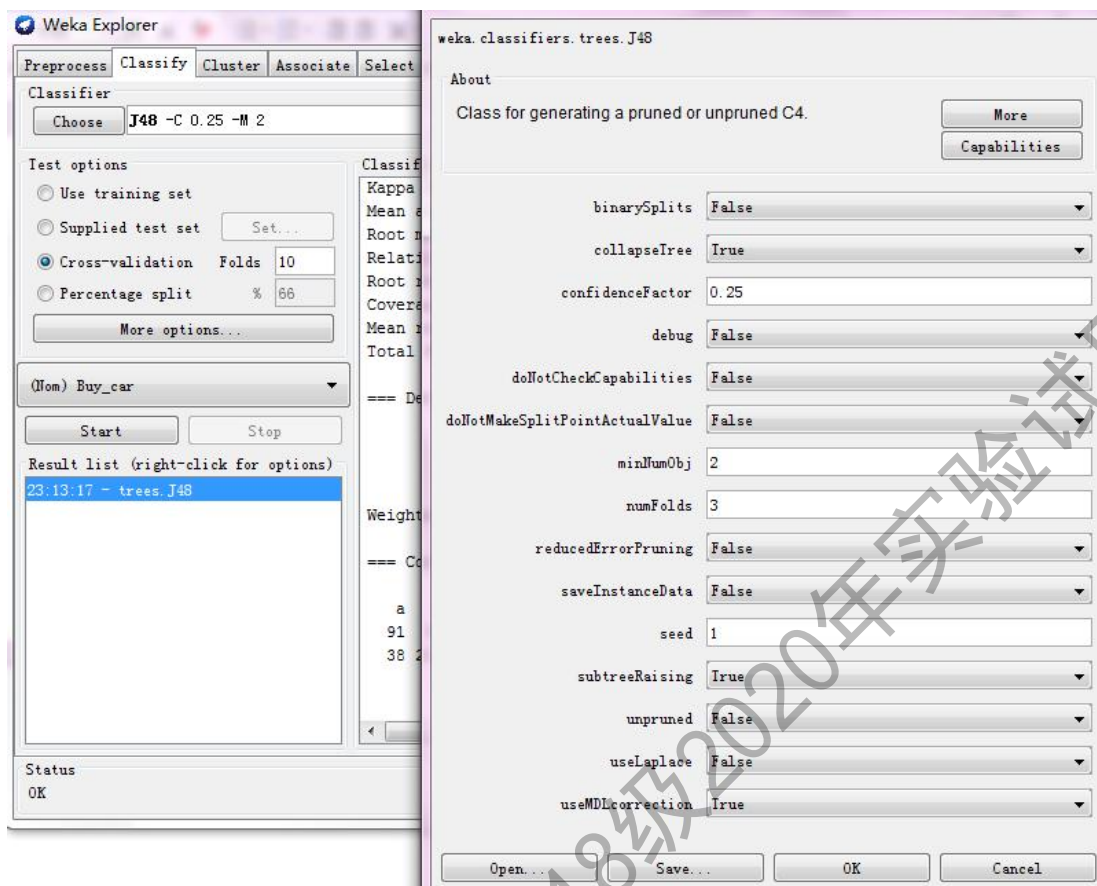
点击 Visualize All 即可查看可视化图形，如下：



Buy_car 反应出最终是否买车的情况，蓝色为买车，红色为不买车。其余属性条中均为蓝色买车，红色不买车。

点击 Classify, choose 算法，双击 J48 出现右图中的属性面板，对属性进行相应的调整（点击 More 即可对所选算法的详细信息进行查看），若选择 percentage split 66%，即用百分之 66 数据作为训练集，剩余百分之 34 用于测试集。选择 Cross_validation，点击 Start 按钮如下：





如下所示为改组数据的测试结果，可看出本组数据源有 400 个数据，六个属性项，分别为 Sex、Age、Salary、FamilyM、EducationIS、Buy_car，其中最后一组数据为最终测试评估结果。后图可以看出该用例最终的正确率为 80%。

=== Run information ===

```

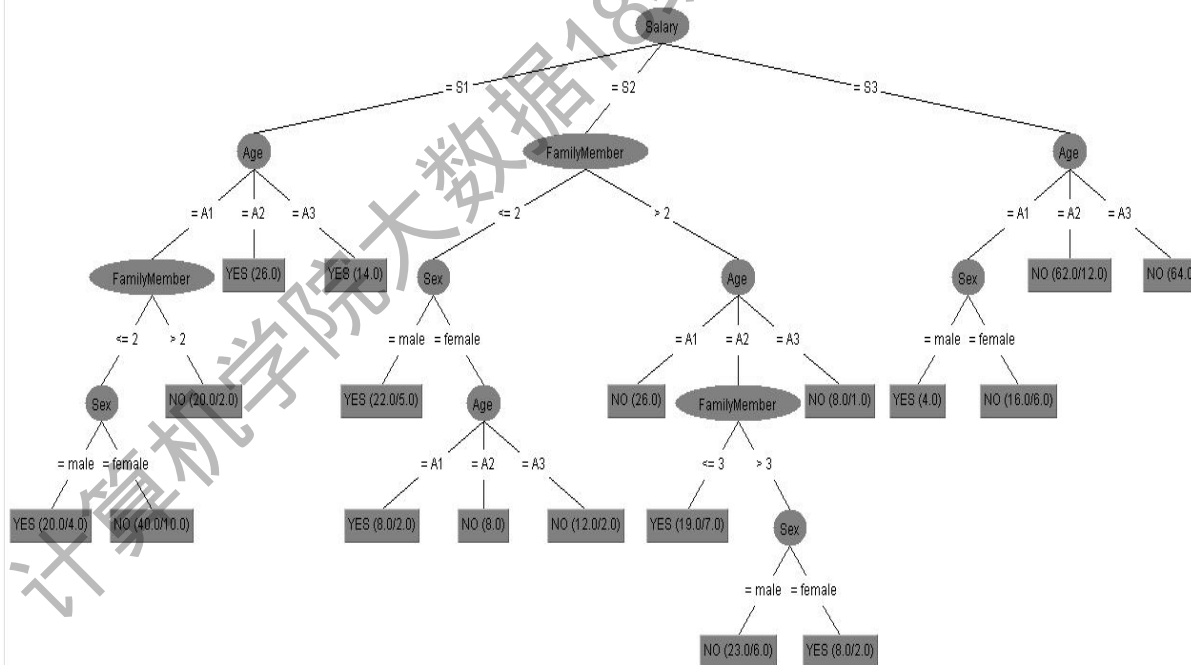
Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    Buy_car
Instances:    400
Attributes:   6
              Sex
              Age
              Salary
              FamilyM
              educationIS
              Buy_car
Test mode:    10-fold cross-validation

```

=== Classifier model (full training set) ===

Correctly Classified Instances	320	80	%
Incorrectly Classified Instances	80	20	%
Kappa statistic	0.5439		
Mean absolute error	0.2562		
Root mean squared error	0.3816		
Relative absolute error	56.2752	%	
Root relative squared error	80.0104	%	
Coverage of cases (0.95 level)	99.25	%	
Mean rel. region size (0.95 level)	84.625	%	
Total Number of Instances	400		

生成的可视化决策树如下图，由该图可以看出年龄、性别、家庭成员数、教育程度和工资收入对该人是否买车的作用，从而帮助商家更好的预测购买群体，做相应的宣传或规划。

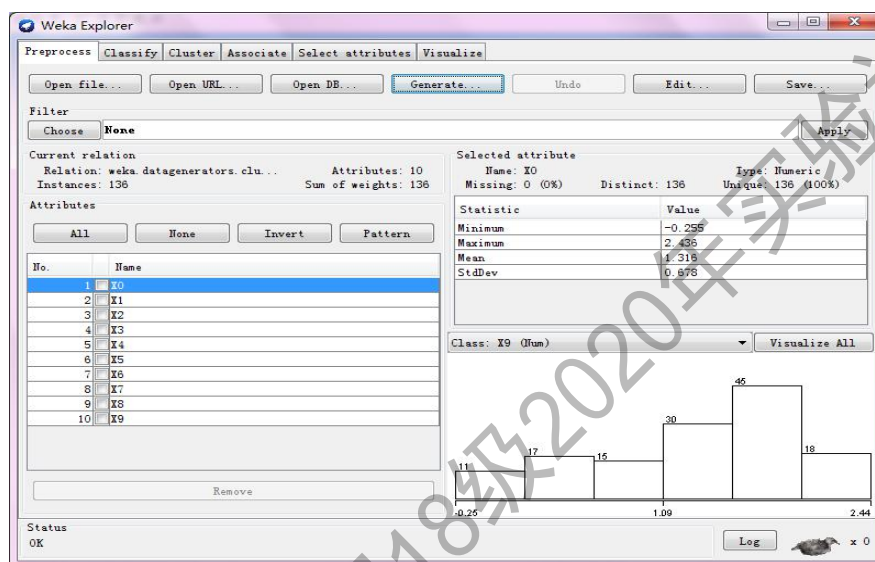


(3) 用于聚类——SimpleKMeans 算法

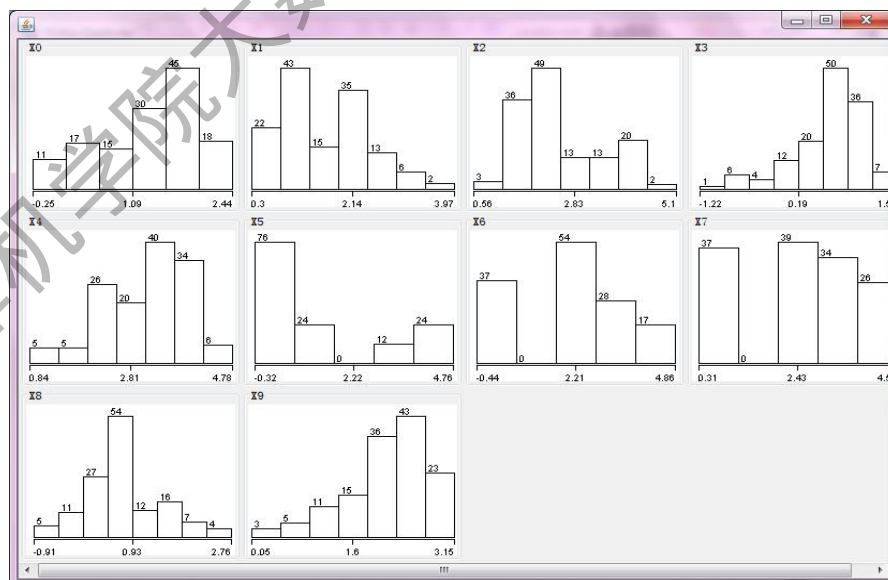
该实例利用 Generate 生成器直接生成一组粒子数据，进行聚类分析，具体操作如下：



选择生成聚类数据 BIRCHCluster 数据，点击 Generate，即完成数据的生成。以下为软件自动生成数据图：

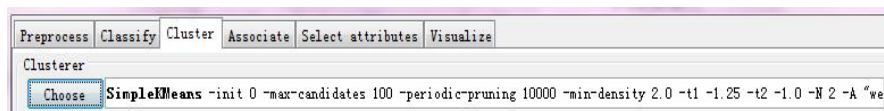


点击 Visualize All，即可看到十个属性各自的情况：

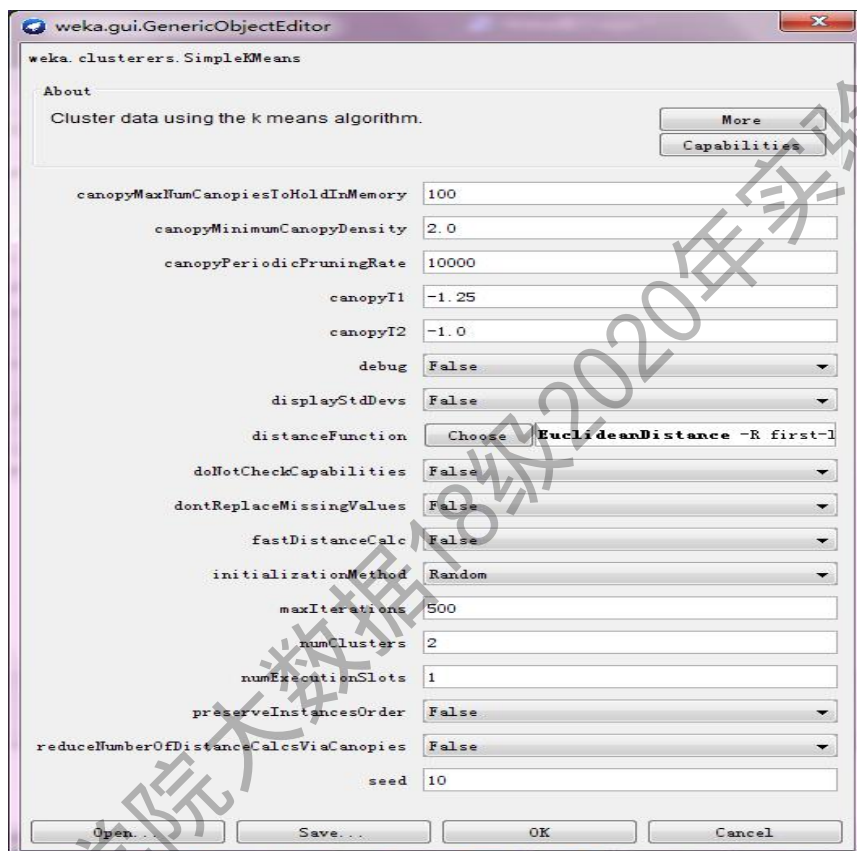


对这十组属性进行聚类分析，我们选择 SimpleKMeans 方法进行操作点击

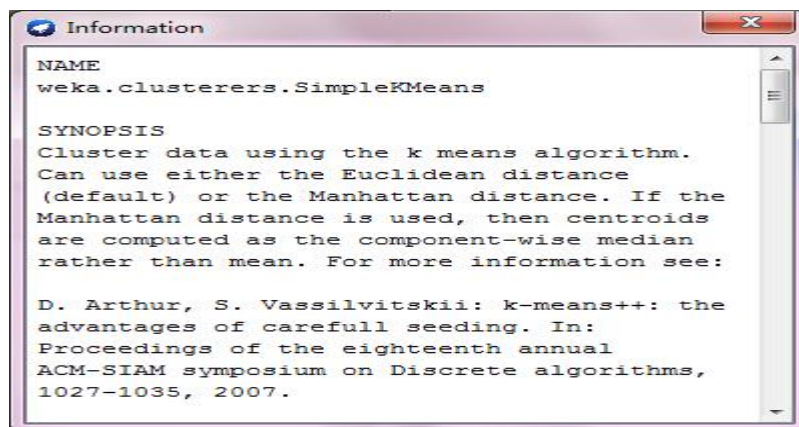
Cluster, 在 Choose 中选择 SimpleKMeans 方法如下:



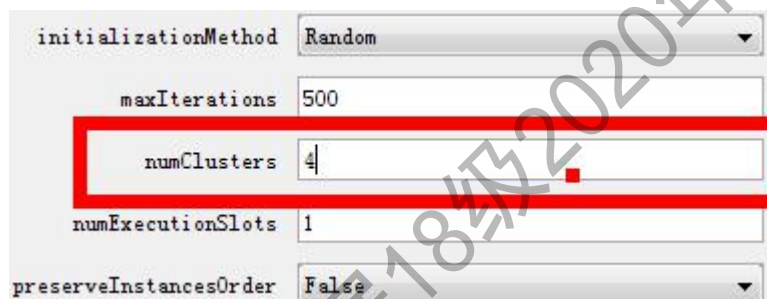
双击 MakeDensityBasedClusterer 出现如下图所示, 可以更改该方法的属性设置。



点击 more 可以查看更多相关 SimpleKMeans 算法的情况, 如下:



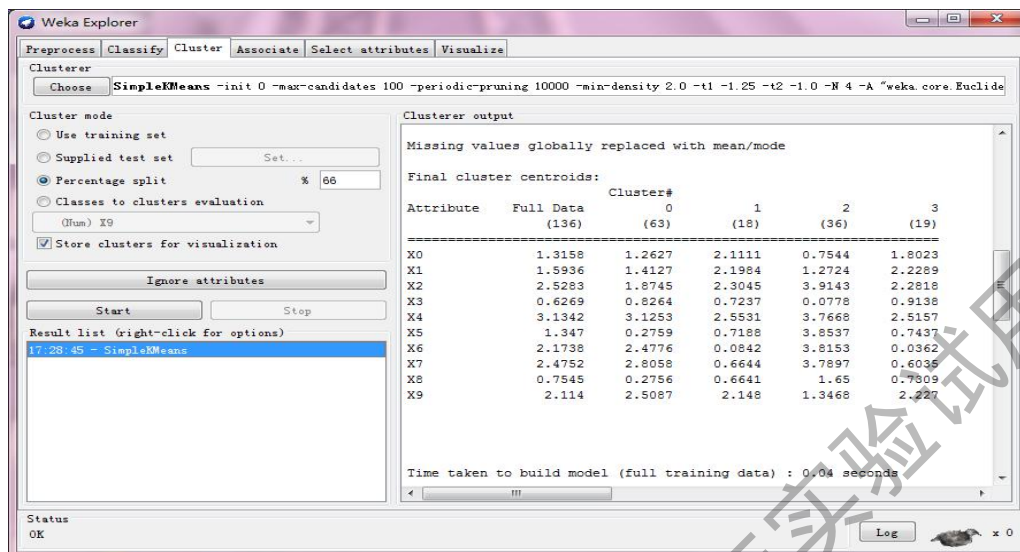
对属性进行相应的修改，如下图，将系统自带的 numCluster 为 2 改为 numCluster 为 4:



点击 percentage split %66 用于作为训练集，剩余 34%用于作为测试集，如下图:

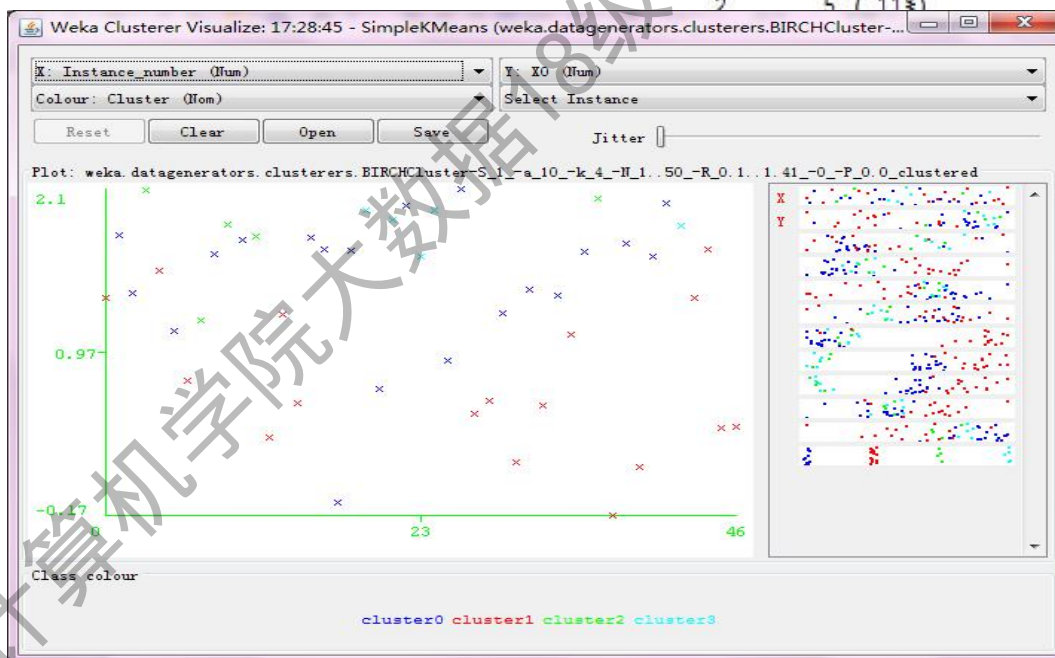


点击 Start 即完成对数据集的聚类操作，由于刚才对 numCluster 的修改，所以我们可以看到下图右侧中产生了对应的四个簇:

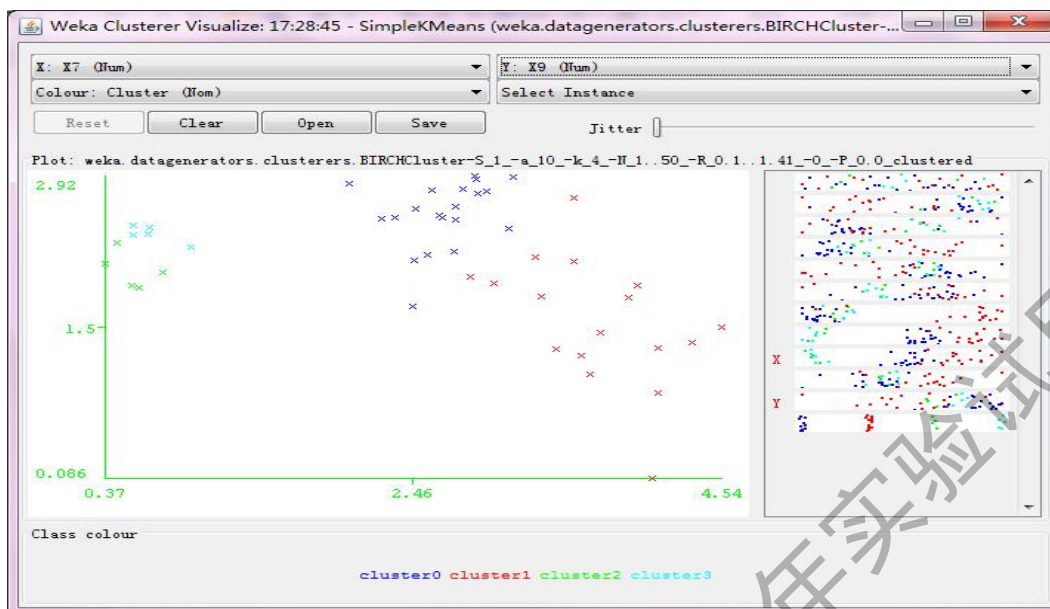


各个簇包含的点数和所占的比例如右图：

在 SimpleKMeans 上单击右键，选择 Visualize cluster assignments, 即可看到如下图：



该图对性的 X,Y 轴可以有用户自己选择，调整已达到最明显的效果，我们发现当选择 X 轴为 x7,Y 轴为 x9 时，可以清楚的看到四种颜色的簇有自己明显的边界，进而我们可以对这些簇进行观察：



Jitter 可用于对可视图进行放大缩小，点击图中任意一点，即可得到该点的相关信息，如下：



类似的我们可以分别观察各个点，找到该点的相关属性和该点包含于哪个簇，通过这样的方法，我们可以实现对一组数据使用 SimpleKmeans 方法进行聚类分析。

实验一 数据预处理实验

一、实验目的

- 1、掌握数据探索统计特征计算、数据可视化等基本方法
- 2、掌握数据集缺失值、含噪数据的平滑处理、数据变换、数据集成等预处理方法。
- 3、掌握PCA主成分分析等降维方法

实验类型：验证

计划课间：6学时

二、实验内容

- 1、利用可视化工具实现选择数据的统计特征计算、箱体图和小提琴图的绘制；
- 2、对选择的数据进行缺失值、含噪数据的平滑处理、数据变换、数据集成等；
- 3、对选择的数据编程实现PCA主成分分析，实现数据降维；

三、实验步骤

- 1、对某县广电宽带用户的5000条数据（或者自己感兴趣的其他领域的的数据）进行探索，通过统计特征和可视化进行数据分析，探索发现你感兴趣的知识。
- 2、对北京西安的年薪数据（或者自己感兴趣的其他领域的的数据）计算均值，方差等统计特征，

绘制数据箱体图和小提琴图等图，分析北京西安年薪的差异。

3、数据清洗

用“movie_metadata.csv”数据集（或者自己感兴趣的其他领域的数据）进行案例分析，这个数据集包含了包括演员、导演、预算、总输入，以及 IMDB 评分和上映时间等信息，进行处理缺失数据，可以是添加默认值，删除不完整的行，异常值处理，重复数据处理，规范化数据类型等等。

4、数据集成

合并两个给定数据集：ReaderRentRecode.csv和ReaderInformation.csv（或者自己感兴趣的其他领域的数据），其中两个数据集的共同点是具有相同的num属性，最终生成一个综合的数据集。

5、数据归约：PCA 主成分分析

使用鸢尾花数据集（或者自己感兴趣的其他领域的数据），这个数据集有150个样本，其中每个样本有五个变量，其中四个为特征变量，分别为萼片长度(Sepal length)，萼片宽度(Sepal width)，花瓣长度(Petal length)，花瓣宽度(Petal width)，还有一个变量是其所属的品种类别变量(Species)，这个鸢尾花内别共有3种类别分别是山鸢尾(Iris-setosa)、变色鸢尾(Iris-versicolor)和维吉尼亚鸢尾(Iris-virginica)，首先对4维的原始数据集实现可视化，可视化一组数据来观察数据分布，然后对数据集进行标准化（归一化），接着利用PCA主成分分析将数据降到二维。

四、实验报告要求

- 1、尽量使用高级语言实现上述相关步骤（有困难的同学可以使用工具实现）。
- 2、实验操作步骤和实验结果，实验中出现的问题和解决方法。

五、注意事项

- 1、集合的表示及相关操作的实现；
- 2、PCA主成分分析高级语言实现要借助于一些数学库；

实验二 Apriori 算法实现

一、实验目的

- 1、掌握Apriori算法对于关联规则挖掘中频繁集的产生以及关联规则集合的产生过程；
 - 2、根据算法描述编程实现算法，调试运行。并结合相关实验数据进行应用，得到分析结果。
- 数据和删除数据的操作。

实验类型：验证

计划课间：6学时

二、实验内容

- 1、频繁项集的生成与Apriori算法实现；
- 2、关联规则的生成过程与Rule-generate算法实现；
- 3、结合样例对算法进行分析；

三、实验步骤

编写程序完成下列算法：

1、Apriori算法

输入：数据集D；最小支持数minsup_count；

输出：频繁项目集L

$L_1 = \{\text{large 1-itemsets}\}$

For ($k=2$; $L_{k-1} \neq \Phi$; $k++$)

$C_k = \text{apriori-gen}(L_{k-1})$; // C_k 是k个元素的候选集

For all transactions $t \in D$ do

begin $C_t = \text{subset}(C_k, t)$; // C_t 是所有t包含的候选集元素

for all candidates $c \in C_t$ do $c.\text{count}++$;

end

$L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup_count}\}$

End

$L = \cup L_k$;

2、apriori-gen (L_{k-1}) 候选集产生算法

输入: $(k-1)$ -频繁项目集 L_{k-1}

输出: k -频繁项目集 C_k

For all itemset $p \in L_{k-1}$ do

For all itemset $q \in L_{k-1}$ do

If $p.item_1 = q.item_1, p.item_2 = q.item_2, \dots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
then

begin $c = p \cup q$

if $has_infrequent_subset(c, L_{k-1})$

then delete c

else add c to C_k

End

Return C_k

3、 $has_infrequent_subset(c, L_{k-1})$

功能: 判断候选集的元素

输入: 一个 k -频繁项目集 L_{k-1} , $(k-1)$ -频繁项目集 L_{k-1}

输出: c 是否从候选集中删除的布尔判断

For all $(k-1)$ -subsets of c do

If Not $(S \in L_{k-1})$ THEN return TRUE;

Return FALSE;

4、Rule-generate($L, minconf$)

输入: 频繁项目集; 最小信任度

输出: 强关联规则

算法:

FOR each frequent itemset lk in L

generules(lk, lk);

5、Genrules递归算法:

Genrules(lk :frequent k -itemset, xm :frequent m -itemset)

$X = \{(m-1)\text{-itemsets } xm-1 \mid xm-1 \text{ in } xm\}$;

For each $xm-1$ in X

```
BEGIN    conf=support(lk)/support(xm-1);  
        IF (conf  $\geq$  minconf) THEN  
        BEGIN  
            输出规则: xm-1  $\rightarrow$  (lk-xm-1), support, confidence;  
            IF (m-1)>1) THEN genrules(lk, xm-1);  
        END;  
    END;
```

结合相关样例数据对算法进行调试，并根据相关实验结果对数据进行分析，

四、实验报告要求

- 1、用高级语言实现上述相关算法。
- 2、实验操作步骤和实验结果，实验中出现的問題和解决方法。

五、注意事项

- 1、集合的表示及相关操作的实现；
- 2、项目集的数据结构描述；

实验三 K-Means 聚类算法实现

一、实验目的

通过分析K-Means聚类算法的聚类原理，利用高级语言编程实现K-Means聚类算法，并通过对样本数据的聚类过程，加深对该聚类算法的理解与应用过程。

实验类型：**验证**

计划课间：**4学时**

二、实验内容

- 1、分析K-Means聚类算法；
- 2、分析距离计算方法；
- 3、分析聚类的评价准则；
- 4、编程完成K-Means聚类算法，并基于相关实验数据实现聚类过程；

三、实验方法

1、K-means聚类算法原理

K-means聚类算法以k为参数，把n个对象分为k个簇，以使簇内的具有较高的相似度。相似度的计算根据一个簇中对象的平均值来进行。

算法描述：

输入：簇的数目k和包含n个对象的数据库

输出：使平方误差准则最小的k个簇

过程：

任选k个对象作为初始的簇中心；

Repeat

for j=1 to n DO

根据簇中对象的平均值，将每个对象赋给最类似的簇

for i=1 to k DO

更新簇的平均值

计算E

Until E不再发生变化

按簇输出相应的对象

2、聚类评价准则:

$$E \text{ 的计算为: } E = \sum \sum_k |x - \bar{x}_i|^2$$

四、实验步骤

4.1 实验数据

4.2 初始簇中心的选择

选择k个样本作为簇中心

```
For (i=0; i<k; i++)
    For (j=0; j<AttSetSize; j++)
        ClusterCenter[i][j]=DataBase[i][j]
```

4.3 数据对象的重新分配

```
Sim=某一较大数; ClusterNo=-1;
For (i=0; i<k; i++)
    If (Distance(DataBase[j], ClusterCenter[i])<Sim)
        {Sim=Distance(DataBase[j], ClusterCenter[i]);
        ClusterNo=i;}
ObjectCluster[j]=ClusterNo;
```

4.4 簇的更新

```
For (i=0; i<k; i++)
    {Temp=0; Num=0;
    For (j=0; j<n; j++)
        If (ObjectCluster[j]==i) {Num++; Temp+=DataBase[j];}
        If (ClusterCenter[i]!=Temp) HasChanged=TRUE;
        ClusterCenter[i]=Temp;
    }
```

4.5 结果的输出

```
For (i=0; i<k; i++)
```

```
{  
Printf(“输出第%d个簇的对象:”, i);  
For (j=0;j<n;j++)  
    If (ObjectCluster[j]==i) printf(“%d ”, j);  
Printf(“\n”);  
Printf(“\t\t\t 簇平均值为(%d,%d)\n”, ClusterCenter[i][0], ClusterCenter[i][1]);  
}
```

五、注意事项

- 1、距离函数的选择
- 2、评价函数的计算

实验四 DBSACN 聚类算法实现

一、实验目的

通过分析DBSACN聚类算法的聚类原理，利用高级语言编程实现DBSACN算法，并通过对样本数据的聚类过程，加深对该聚类算法的理解与应用过程。

实验类型：**验证**

计划课间：**4学时**

二、实验内容

- 1、分析DBSACN算法；
- 2、分析密度涉及的两个参数；
- 3、分析聚类的评价准则；
- 4、编程完成DBSACN算法，并基于相关实验数据实现聚类过程；

三、实验方法

- 1、DBSACN算法原理

四、实验步骤

4.1 实验数据

五、注意事项

- 1、密度参数的选择
- 2、评价函数的计算

实验五 ID3 算法实现

一、实验目的

通过编程实现决策树算法，信息增益的计算、数据子集划分、决策树的构建过程。加深对相关算法的理解过程。

实验类型：**验证**

计划课间：**6学时**

二、实验内容

- 1、分析决策树算法的实现流程；
- 2、分析信息增益的计算、数据子集划分、决策树的构建过程；
- 3、根据算法描述编程实现算法，调试运行；
- 4、对作业数据进行验算，得到分析结果。

三、实验方法

算法描述：

以代表训练样本的单个结点开始建树；

若样本都在同一个类，则该结点成为树叶，并用该类标记；

否则，算法使用信息增益作为启发信息，选择能够最好地将样本分类的属性；

对测试属性的每个已知值，创建一个分支，并据此划分样本；

算法使用同样的过程，递归形成每个划分上的样本决策树

递归划分步骤，当下列条件之一成立时停止：

给定结点的所有样本属于同一类；

没有剩余属性可以进一步划分样本，在此情况下，采用多数表决进行

四、实验步骤

- 1、算法实现过程中需要使用的数据结构描述：

```
Struct
{int Attrib_Col;           // 当前节点对应属性
int Value;                 // 对应边值
Tree_Node* Left_Node;     // 子树
Tree_Node* Right_Node     // 同层其他节点
Boolean IsLeaf;           // 是否叶子节点
int ClassNo;              // 对应分类标号
```

```
}Tree_Node;
```

2、整体算法流程

主程序：

```
InputData();
T=Build_ID3(Data,Record_No, Num_Attrib);
OutputRule(T);
释放内存;
```

3、相关子函数：

3.1、 InputData()

```
{
    输入属性集大小Num_Attrib;
    输入样本数Num_Record;
    分配内存Data[Num_Record][Num_Attrib];
    输入样本数据Data[Num_Record][Num_Attrib];
    获取类别数C(从最后一列中得到);
}
```

3.2、 Build_ID3(Data,Record_No, Num_Attrib)

```
{
    Int Class_Distribute[C];
    If (Record_No==0) { return Null }
    N=new tree_node();
    计算Data中各类的分布情况存入Class_Distribute
    Temp_Num_Attrib=0;
    For (i=0;i<Num_Attrib;i++)
        If (Data[0][i]>=0) Temp_Num_Attrib++;
        If Temp_Num_Attrib==0
        {
            N->ClassNo=最多的类;
            N->IsLeaf=TRUE;
            N->Left_Node=NULL;N->Right_Node=NULL;
            Return N;
        }
    If Class_Distribute中仅一类的分布大于0
    {
        N->ClassNo=该类;
        N->IsLeaf=TRUE;
        N->Left_Node=NULL;N->Right_Node=NULL;
        Return N;
    }
}
```

```

        InforGain=0;CurrentCol=-1;

For i=0;i<Num_Attrib-1;i++)
{
    TempGain=Compute_InforGain(Data,Record_No,i,Num_Attrib);
    If (InforGain<TempGain)
        { InforGain=TempGain; CurrentCol=i;}
}
N->Attrib_Col=CurrentCol;
//记录CurrentCol所对应的不同值放入DiferentValue[];
I=0;Value_No=-1;
While i<Record_No {
    Flag=false;
    For (k=0;k<Value_No;k++)
    if (DiferentValu[k]=Data[i][CurrentCol]) flag=true;
    if (flag==false)
    {Value_No++;DiferentValue[Value_No]=Data[i][CurrentCol] }
    I++;
}
SubData=以Data大小申请内存空间;
For (i=0;i<Value_No;i++)
{
    k=-1;
    for (j=0;j<Record_No-1;j++)
        if (Data[j][CurrentCol]==DiferentValu[i])
            {k=k++;
For(int il=0;il<Num_Attrib;il++)
If (il<>CurrentCol)SubData[k][il]=Data[j][il];
    Else SubData[k][il]=-1;
        }
    N->Attrib_Col=CurrentCol;
    N->Value=DiferentValu[i];
    N->Isleaf=false;
    N->ClassNo=0;
    N->Left_Node=Build_ID3(SubData,k+1, Num_Attrib);
    N->Right_Node=new Tree_Node;
    N=N->Right_Node;
}
}

```

3.3、计算信息增益

```

Compute_InforGain(Data, Record_No, Col_No, Num_Attrib)
{
    Int DifferentValue[MaxDifferentValue];
    Int Total_DifferentValue;
    Int s[ClassNo][MaxDifferentValue];

    s=0; // 数组清0;
    Total_DifferentValue=-1;
    For (i=0; i<Record_No; i++)
    {
        J=GetPosition(DifferentValue,
            Total_DifferentValue, Data[i][Col_no]);
        If (j<0) {Total_DifferentValue++;
            DifferentValue[Total_DifferentValue]=Data[i][Col_no];
            J=Total_DifferentValue;}
        S[Data[i][Num_Attrib-1]][j]++;
    }
    Total_I=0;
    For (i=0; i<ClassNo; i++)
    {
        Sum=0;
        For(j=0; j<Record_No; j++) if Data[j][Num_Attrib-1]==i sum++;
        Total_I=Compute_PI(Sum/Record_No);
    }
    EA=0;
    For (i=0; i<Total_DifferentValue; i++);
    {
        temp=0; sj=0; //sj是数据子集中属于类j的样本个数;
        For (j=0; j<ClassNo; j++)
            sj+=s[j][i];
        For (j=0; j<ClassNo; j++)
            EA+=sj/Record_No*Compute_PI(s[j][i]/sj);
    }
    Return total_I-EA;
}

```

3.4、得到某数字在数组中的位置

```

GetPosition(Data, DataSize, Value)
{
    For (i=0; i<DataSize; i++) if (Data[i]=value) return I;
    Return -1;
}

```

```
}
```

3.5、计算 $Pi * \log Pi$

```
Float Compute_PI(float pi)
{
    If pi<=0 then return 0;
    If pi>=1 then return 0;
    Return 0-pi*log2(pi);
}
```

五、实验报告要求

- 1、用高级语言实现上述相关算法。
- 2、实验操作步骤和实验结果，实验中出现的问题和解决方法。

六、注意事项

- 1、信息增益的计算；
- 2、选择相关字段后根据相关字段的取值对数据集合进行划分。
- 3、决策树构建的终止条件

实验六 贝叶斯算法实现

一、实验目的

通过对贝叶斯算法的编程实现，加深对贝叶斯算法的理解，同时利用贝叶斯算法对简单应用实现预测分类

实验类型：**验证**

计划课间：**4学时**

二、实验内容

- 1、分析贝叶斯算法；
- 2、计算条件概率；
- 3、预测精度的计算与评估；
- 4、编程实现贝叶斯分类算法，并对简单应用样本数据实现预测分类

三、实验方法

- 1、实现贝叶斯算法
- 2、利用实验数据对贝叶斯算法进行检测
- 3、求解精确度计算
- 4、调试程序
- 5、完成整个分类与评估的过程

四、实验步骤

4.1 算法过程描述：

- 1) 输入训练数据，将数据保存在DataBase二维数组中(数组的最后一个属性对应类别标号)
- 2) 设定训练数据集与测试数据集大小(指定从数组下标0开始到TrainSetSize-1所对应的数据为训练数据，其余为测试数据)；
- 3) 计算训练数据集数据中各属性在各类中的概率分布情况；
- 4) 利用测试数据计算贝叶斯算法的分类精度；
- 5) 输出分类结果；

4.2 数据处理

A、实验数据

RID	age	income	student	Credit_rating	BuyComputer
1	≤30	High	No	Fair	No

2	≤ 30	High	No	Excellent	No
3	31~40	High	No	Fair	Yes
4	>40	med	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31~40	Low	Yes	Excellent	Yes
8	≤ 30	Med	No	Fair	No
9	≤ 30	Low	Yes	Fair	Yes
10	>40	Med	Yes	Fair	Yes
11	≤ 30	Med	Yes	Excellent	Yes
12	31~40	Med	No	Excellent	Yes
13	31~40	High	Yes	Fair	Yes
14	>40	med	No	Excellent	No

B、对数据中的枚举类型数据进行转换以便于数据处理：

	0	1	2	3	ClassNo
1	0	0	0	0	0
2	0	0	0	1	0
3	1	0	0	0	1
4	2	1	0	0	1
5	2	2	1	0	1
6	2	2	1	1	0
7	1	2	1	1	1
8	0	1	0	0	0
9	0	2	1	0	1
10	2	1	1	0	1
11	0	1	1	1	1
12	1	1	0	1	1
13	1	0	1	0	1
14	2	1	0	1	0

4.3 计算训练数据集数据中各属性在各类中的概率分布情况如图5-1所示

4.4 利用测试数据计算贝叶斯算法的分类精度如图5-2所示

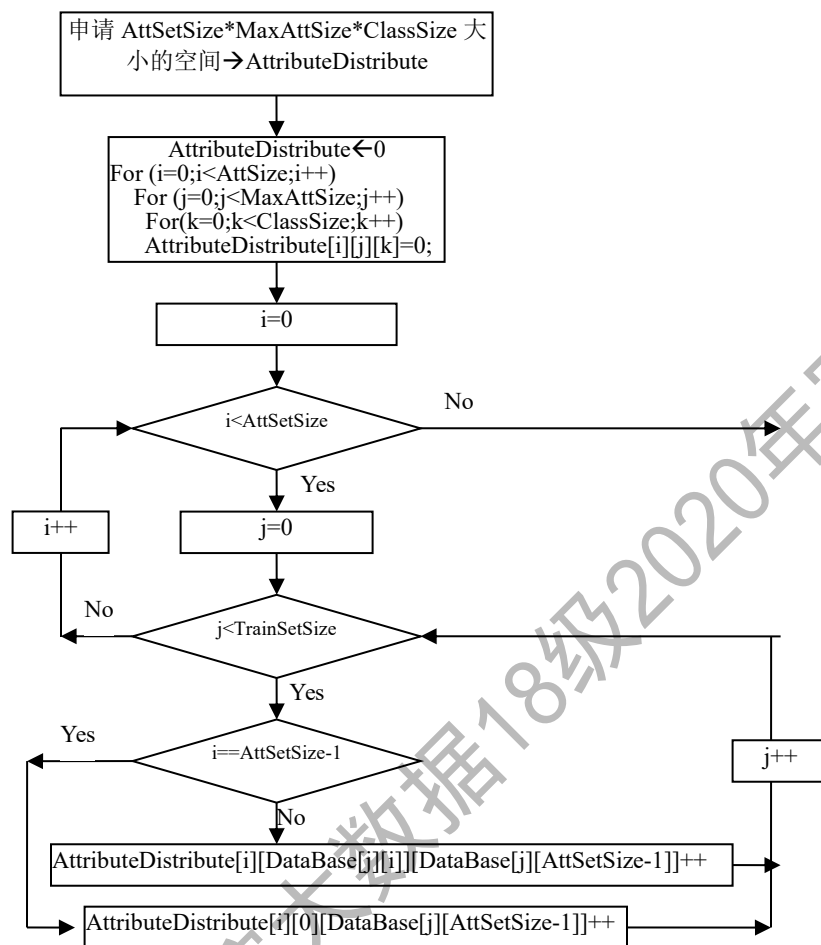


图5-1 训练数据集各属性的概率分布计算

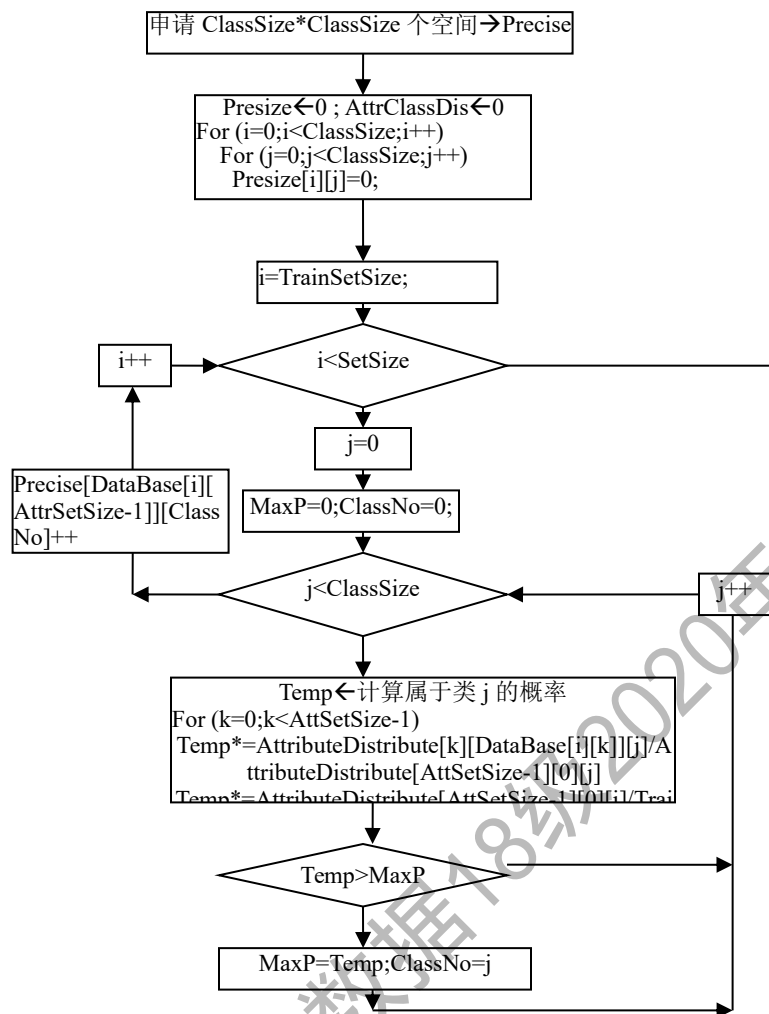


图5-2 贝叶斯算法的分类精度计算

4.5 输出分类结果

```

For (i=0; i < ClassSize; i++)
{
    printf(“\n”);
    For (j=0; j < ClassSize; j++)    printf(“\t%d”, Precise[i][j]);
    TotalCorrect += Precise[i][i];
}
printf(“\n\nTotal Correct is %d”, TotalCorrect);

```

五、注意事项

注意单个样例数据的概率计算与各字段的概率计算的关系