# 一、基本概念

数据挖掘就是在数据中寻找模式的过程。（如客户忠诚度分析、市场购物篮分析等）

这个寻找过程可以是自动的或者是半自动的。（数据量特别大的时候只能是自动的）

而自动的数据分析方法则来自机器学习的结果。（机器学习就是自动找出数据模式的一套方法）

**分类与回归**

分类函数或者分类模型，预测数据的类别。区别：分类是离散值，回归是连续值。

**聚类分析**

把数据划分成不同的簇

**关联分析**

数据的联系，发现关联规则

# 二、关联规则挖掘

## 定义

**关联规则：**分为两部分，一个是规则的前提，一个是规则的结论。即由前提推导出结论。

**覆盖率=支持度：**可以表示为0-1之间的一个比值（小数），指的是满足整个规则（前提和结论）的实例数与实例总数的比例。

**最低支持度：**就是支持度的下限的意思。

**准确率=置信度：**书面上的解释为：表示支持度数值占应用规则后的实例数量的比例。这段话估计没几个人能看的明白，其实很简单。就是满足某个规则的前提，又满足该规则的结论的实例数（双满足），与 满足这个规则前提（单满足）的实例数之比。它的目的其实就是告诉用户，符合了这个规则的前提，有多大比例能导致这个规则推导出的结论。

**项集 包含 多个 项：**每个项都是 字段名=字段值 的形式

**提升度：**

提升度指的是： 符合规则前提的实例，其置信度 除以 所有实例的置信度（指符合当前关联规则的结论），所得的倍数。 P(L,R)/(P(L)P(R))  4/14 / 4/14 \* 9/14

这是一个衡量关联价值的指标，独立于支持度

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。

英文原文定义如下：

Lift is confidence divided by the proportion of all examples that are covered by the consequence.

This is a measure of the importance of the association that is independent of support.

**杠杆率：**指的是新增实例符合关联规则的前提及结论的比例是多少。后面括号中的数字就是符合这个杠杆率的实例数量。P(L,R)-P(L)P(R) 4/14 - 4/14 \* 9/14

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。

Leverage is the proportion of additional examples covered by both the premise and consequence above those expected if the premise and consequence were independent of each other. The total number of examples that this represents is presented in brackets following the leverage.

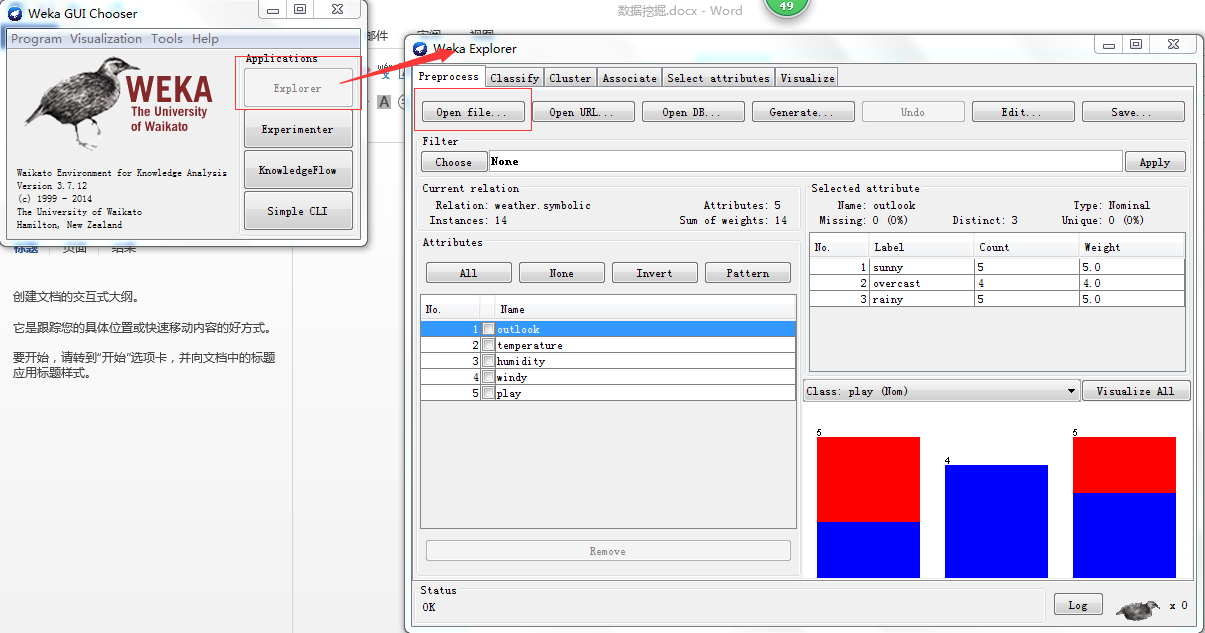
**确信度：**

确信度= P(L)P(!R)/P(L,!R) （!R表示R没有发生）,P(L,!R)  最小值为 P(1),而不是实际的P(0) 4/14 \* 5/14 / 1/14 =

Conviction is another measure of departure from independence. Conviction is given by P(premise)P(!consequence) / P(premise, !consequence).

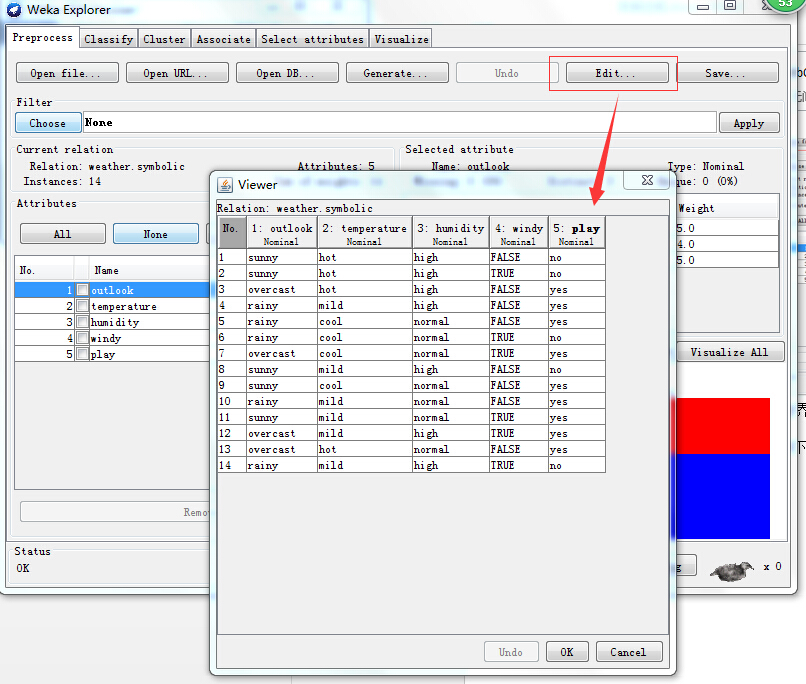
## 实例1

Weather.nominal.arff



单击 Explorer 打开 探索者界面。 单击 open file ，然后选择Weather.nominal.arff 文件。

（这个文件在你weka的安装文件夹下，我的是 C:\Program Files\Weka-3-7\data 下）

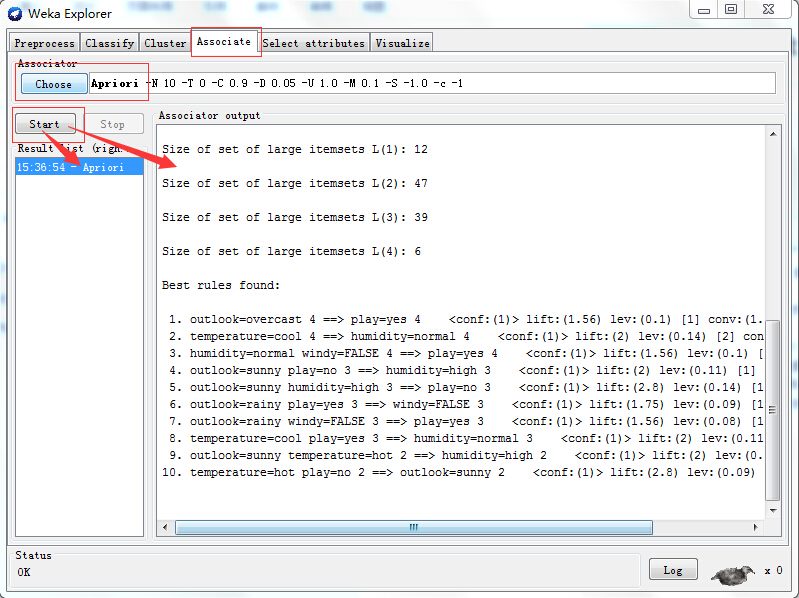


单击 Edit 可以打开 Weather.nominal.arff 中的数据，我们可以查看、排序、修改、删除数据，也可以删除字段、修改字段名等操作。

注意一点：这里的修改只影响内存中的数据，对数据文件本身没有影响。

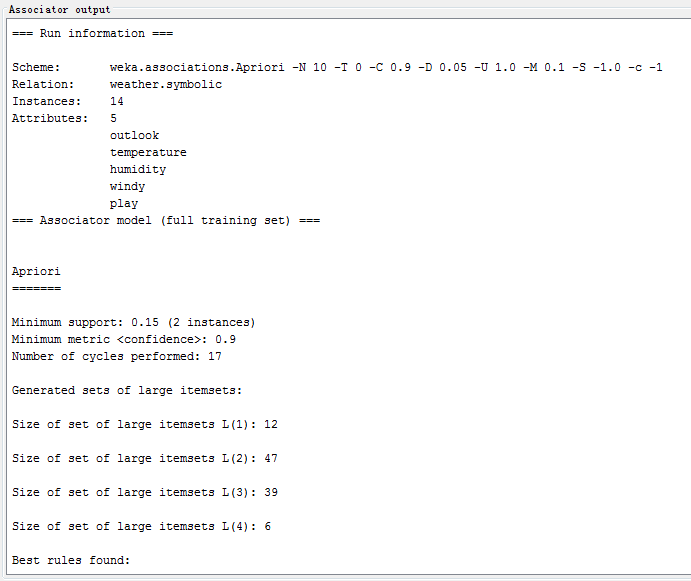
最上面的一行是字段名、字段类型。

因为关联算法只支持字段类型为nominal（就是字符型） 的字段数据，所以如果遇到numeric（就是数字）的字段类型，得先转化。转化的方式就是分类（比如小于10的属于一类， 10 – 20 的一类等等），如果不想分类的话，就直接删掉这个数字字段。（分类其实也是一个复杂的数据挖掘大功能，但我们就采用自定义分类的方式简单处理好了）



切换到 associate 面板，在关联面板中我们可以发现系统已经帮我们默认选择了apriori 关联算法，还指定了一大堆默认参数。 其实关联算法就两个，默认的这个就是我们要用的那个，所以什么都不用设，直接单击 start， 开始挖掘吧 （其实，为了更好的挖掘，那些参数是需要改动的）。

挖掘的结果都在右边栏那一大堆英文上显示出来了。下面一个一个解释。（左边栏那个就是什么时候开始，用什么算法挖掘的意思）



=== Run information === //运行信息

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

//用了什么算法 + 参数

Relation: weather.symbolic //数据集名称

Instances: 14 //实例数，其实就是数据行数

Attributes: 5 //属性数，就是字段数

Outlook // 字段列表，5个字段名称被一一列出

temperature

humidity

windy

play

=== Associator model (full training set) === // 关联模型 （完整的训练集），机器学习的过程又叫训练，意思就是完整的机器学习，找出了下面这些模式规律

Apriori //算法名字

=======

Minimum support: 0.15 (2 instances) //最低支持度 0.15，也就是15%。总数的15%也就是14\*0.15=2.1 约为2 个实例

Minimum metric <confidence>: 0.9 // 最低置信度 0.9

Number of cycles performed: 17 //算法循环执行了17次，这个次数是由两个算法参数决定的，后面会解释

Generated sets of large itemsets: //生成的频繁项集列表（只要符合最低支持度0.15，即2个实例满足即可，此时还不涉及结论，只是前提）

Size of set of large itemsets L(1): 12  //12 个大小为1 的项集

Size of set of large itemsets L(2): 47 //47 个大小为2 的项集

Size of set of large itemsets L(3): 39 //39 个大小为3 的项集

Size of set of large itemsets L(4): 6 //6 个大小为4 的项集

如果你想看看到底都是些什么项集的话，可以把算法参数的outputitemsets 设置成 true。你就会看到这些东西：

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Large Itemsets L(1):

outlook=sunny 5 // 后面的数字表示符合这个项集的实例数

outlook=overcast 4

outlook=rainy 5

temperature=hot 4

temperature=mild 6

temperature=cool 4

humidity=high 7

humidity=normal 7

windy=TRUE 6

windy=FALSE 8

play=yes 9

play=no 5

Size of set of large itemsets L(2): 47

Large Itemsets L(2):

outlook=sunny temperature=hot 2

outlook=sunny temperature=mild 2

outlook=sunny humidity=high 3

outlook=sunny humidity=normal 2

outlook=sunny windy=TRUE 2

outlook=sunny windy=FALSE 3

outlook=sunny play=yes 2

outlook=sunny play=no 3

outlook=overcast temperature=hot 2

outlook=overcast humidity=high 2

outlook=overcast humidity=normal 2

outlook=overcast windy=TRUE 2

outlook=overcast windy=FALSE 2

outlook=overcast play=yes 4

outlook=rainy temperature=mild 3

outlook=rainy temperature=cool 2

outlook=rainy humidity=high 2

outlook=rainy humidity=normal 3

outlook=rainy windy=TRUE 2

outlook=rainy windy=FALSE 3

outlook=rainy play=yes 3

outlook=rainy play=no 2

temperature=hot humidity=high 3

temperature=hot windy=FALSE 3

temperature=hot play=yes 2

temperature=hot play=no 2

temperature=mild humidity=high 4

temperature=mild humidity=normal 2

temperature=mild windy=TRUE 3

temperature=mild windy=FALSE 3

temperature=mild play=yes 4

temperature=mild play=no 2

temperature=cool humidity=normal 4

temperature=cool windy=TRUE 2

temperature=cool windy=FALSE 2

temperature=cool play=yes 3

humidity=high windy=TRUE 3

humidity=high windy=FALSE 4

humidity=high play=yes 3

humidity=high play=no 4

humidity=normal windy=TRUE 3

humidity=normal windy=FALSE 4

humidity=normal play=yes 6

windy=TRUE play=yes 3

windy=TRUE play=no 3

windy=FALSE play=yes 6

windy=FALSE play=no 2

Size of set of large itemsets L(3): 39

Large Itemsets L(3):

outlook=sunny temperature=hot humidity=high 2

outlook=sunny temperature=hot play=no 2

outlook=sunny humidity=high windy=FALSE 2

outlook=sunny humidity=high play=no 3

outlook=sunny humidity=normal play=yes 2

outlook=sunny windy=FALSE play=no 2

outlook=overcast temperature=hot windy=FALSE 2

outlook=overcast temperature=hot play=yes 2

outlook=overcast humidity=high play=yes 2

outlook=overcast humidity=normal play=yes 2

outlook=overcast windy=TRUE play=yes 2

outlook=overcast windy=FALSE play=yes 2

outlook=rainy temperature=mild humidity=high 2

outlook=rainy temperature=mild windy=FALSE 2

outlook=rainy temperature=mild play=yes 2

outlook=rainy temperature=cool humidity=normal 2

outlook=rainy humidity=normal windy=FALSE 2

outlook=rainy humidity=normal play=yes 2

outlook=rainy windy=TRUE play=no 2

outlook=rainy windy=FALSE play=yes 3

temperature=hot humidity=high windy=FALSE 2

temperature=hot humidity=high play=no 2

temperature=hot windy=FALSE play=yes 2

temperature=mild humidity=high windy=TRUE 2

temperature=mild humidity=high windy=FALSE 2

temperature=mild humidity=high play=yes 2

temperature=mild humidity=high play=no 2

temperature=mild humidity=normal play=yes 2

temperature=mild windy=TRUE play=yes 2

temperature=mild windy=FALSE play=yes 2

temperature=cool humidity=normal windy=TRUE 2

temperature=cool humidity=normal windy=FALSE 2

temperature=cool humidity=normal play=yes 3

temperature=cool windy=FALSE play=yes 2

humidity=high windy=TRUE play=no 2

humidity=high windy=FALSE play=yes 2

humidity=high windy=FALSE play=no 2

humidity=normal windy=TRUE play=yes 2

humidity=normal windy=FALSE play=yes 4

Size of set of large itemsets L(4): 6

Large Itemsets L(4):

outlook=sunny temperature=hot humidity=high play=no 2

outlook=sunny humidity=high windy=FALSE play=no 2

outlook=overcast temperature=hot windy=FALSE play=yes 2

outlook=rainy temperature=mild windy=FALSE play=yes 2

outlook=rainy humidity=normal windy=FALSE play=yes 2

temperature=cool humidity=normal windy=FALSE play=yes 2

下面就是真正的结论，干货了。

Best rules found: //发现的最好规则

1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)

outlook=overcast 4 个实例符合这个前提，play=yes 4 个实例符合这个结论。

置信度 1 ， 4除以4 不就是1 吗，表示只要符合这个前提，百分百得出这个结论。

Conf：置信度

Lift：提升度

Lev：杠杠率

Conv：确信度

2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)

3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)

4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)

5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)

6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)

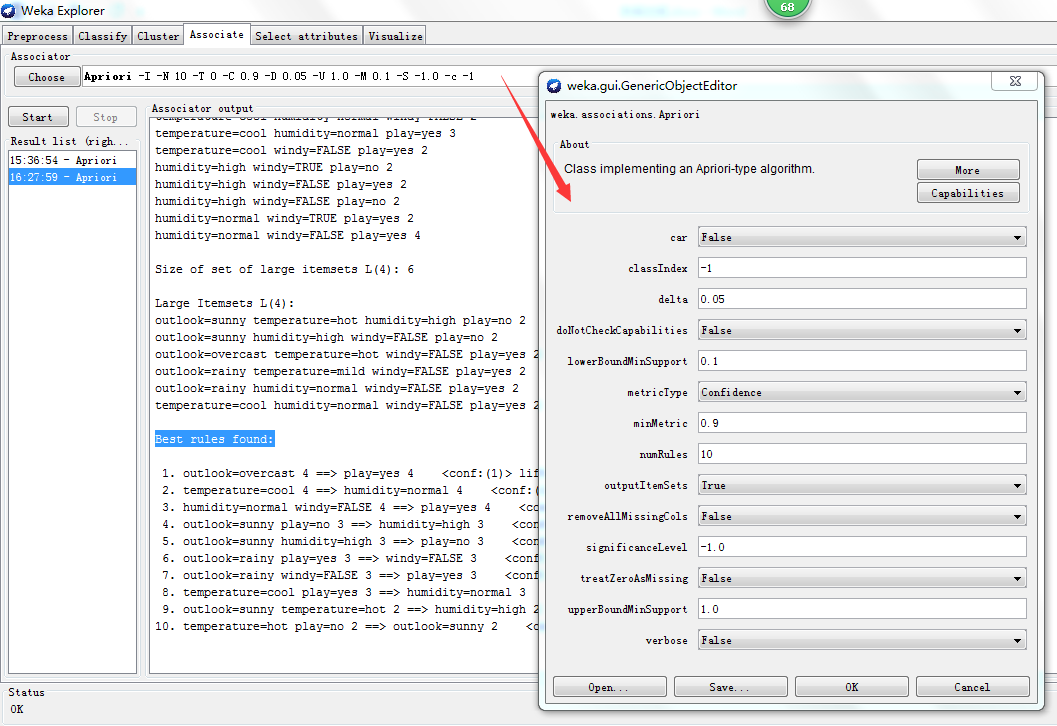
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)

8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)

9. outlook=sunny temperature=hot 2 ==> humidity=high 2 <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)

10. temperature=hot play=no 2 ==> outlook=sunny 2 <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)

参数解释



单击choose 后面的编辑框，就弹出参数设置界面了。重要的红色标注。

Car : 默认false ，分类关联规则。设置为true 的话会采用分类关联规则挖掘，以取代默认的普通关联规则。

ClassIndex：类别属性的索引，默认-1，表示最后一个属性作为类别属性。

**Delta：算法从支持度1- delta（0.05）开始计算运行 ，每运行一次减一次，直到达到最低支持度（0.10）+delta（0.05）为止（再运行下去就不符合大于0.10的要求了）。 1- 17\*0.05 = 0.15 ，这就是为什么算法会运行17次的原因。（运行之前先减delta）**

**LowerBoundMinSupport：最低支持度**

**MetricType：衡量标准 （默认置信度）**

**MinMetric ： 最低置信度 （这个值跟metricType相关联的）**

**NumRules：生成规则数**

**OutputItemSets: 是否显示频繁项集**

**UpperBoundMinSupport:最高支持度**

RemovAllMissingCols：是否移除所有的空字段

TreatZeroAsMissing：是否把0作为空值来处理

Significancelevel:显著性水平

Donotcheckcapabilities:是否检查算法的能力

Verbose：是否启用详细模式

## 实例2

vote.arff

## 实例3

supermarket.arff

=== Run information ===

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Relation: supermarket

Instances: 4627

Attributes: 217

[list of attributes omitted]

=== Associator model (full training set) ===

Apriori

=======

Minimum support: 0.15 (694 instances)

Minimum metric <confidence>: 0.9

Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)

2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)

3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)

4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)

5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779 <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)

6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725 <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)

7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701 <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)

8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866 <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)

9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757 <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)

10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877 <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)

可以发现

1. 购买饼干或冷冻食品 ，会 购买水果 或 蔬菜 （水果 8 条规则，蔬菜 4条规则）
2. 购买饼干、冷冻食品 、水果、蔬菜，会购买面包、蛋糕（多条）
3. 购买上述食品的，采购量会很大（金额高） （9条）
4. 采购量会很大（金额高）的，一般会购买面包、蛋糕（10条规则都包含这两项）

# 三、项目功能设计

挖掘结论可以做一个图表形式，一目了然，且有助于自动分析。

自动分析可以在挖掘结论图表的基础上进行一些计算，得出一些总结建议。也可以弄一个图表的形式。

# 四、WEKA 其它功能

## 连接sql 2012 数据库

1. 下载sql jdbc 驱动jar包 。微软官网去下载就行了
2. 下载下来的压缩包有4个jar文件。

sqljdbc.jar

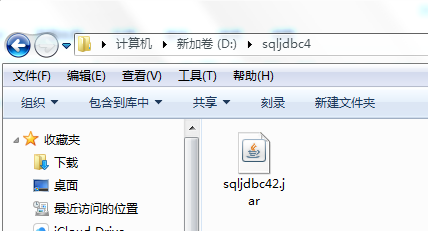
sqljdbc4.jar

sqljdbc41.jar

sqljdbc42.jar

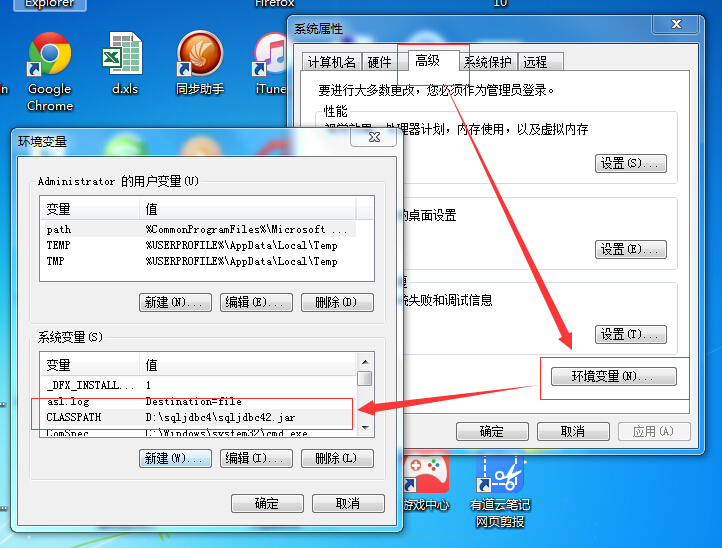
在看了文档之后，我们就知道应该选择sqljdbc42.jar 了

1. 把sqljdbc42.jar 拷贝到一个没有中文路径的文件夹下（建议不要藏太深）



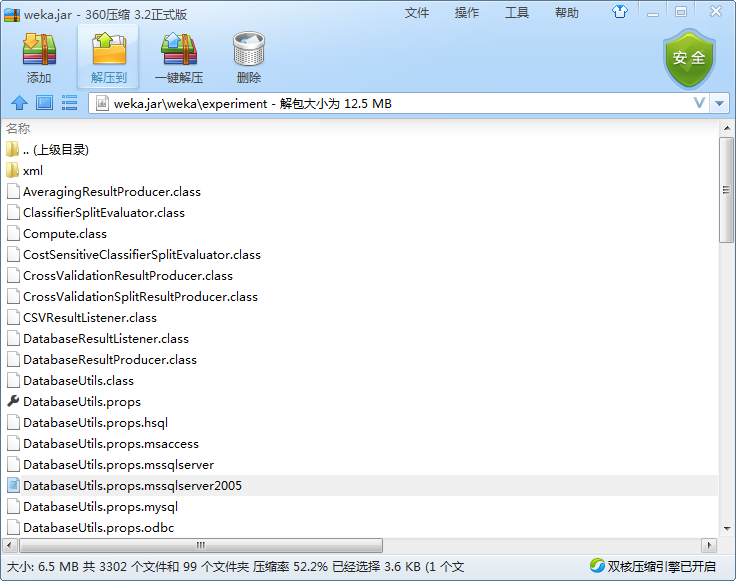
我就这样

1. 加上环境变量



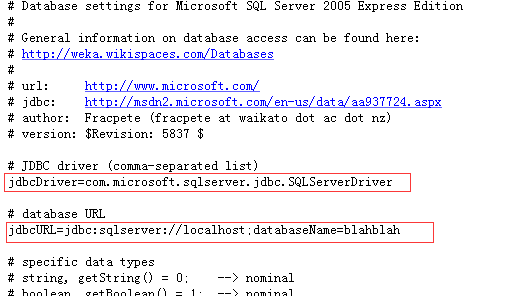
Java是通过环境变量找需要的jar包的

1. 去weka.jar\weka\experiment 中找到 DatabaseUtils.props.mssqlserver2005 文件



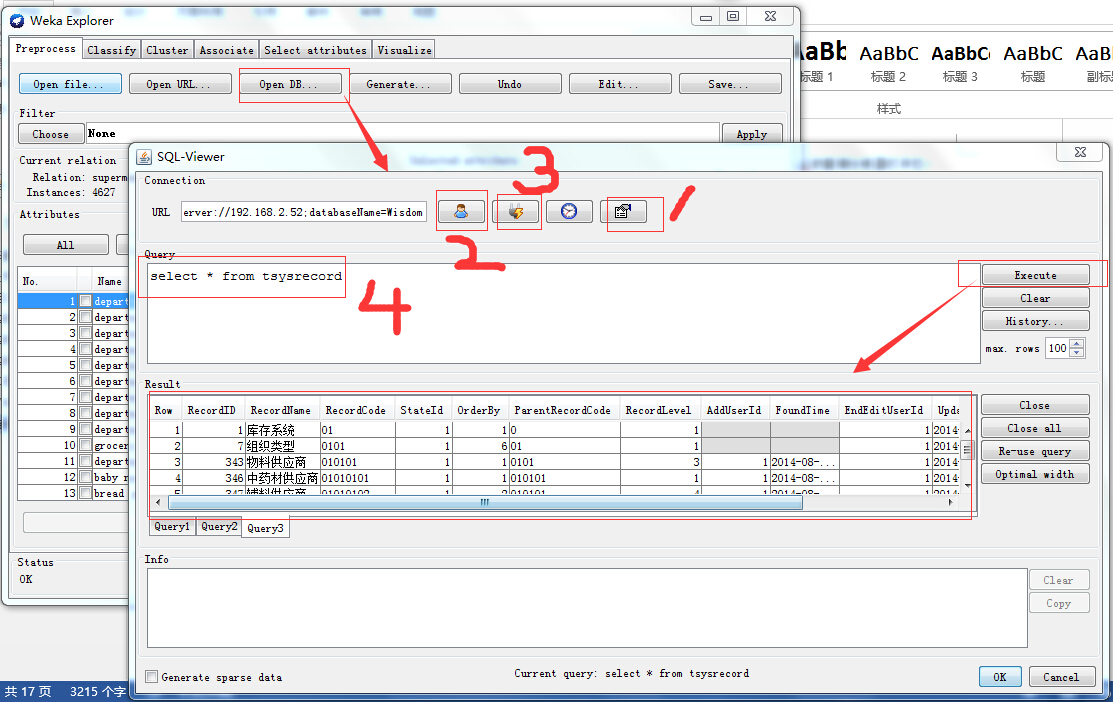
拷贝出来，放到C:\Program Files\Weka-3-7文件夹下，用sql的查询分析器打开它

注意这两行



下面的改成 jdbc:sqlserver://192.168.2.52;databaseName=Wisdom

1. 回到weka



1. 单击 1 ： 选择刚才的那个DatabaseUtils.props.mssqlserver2005 文件

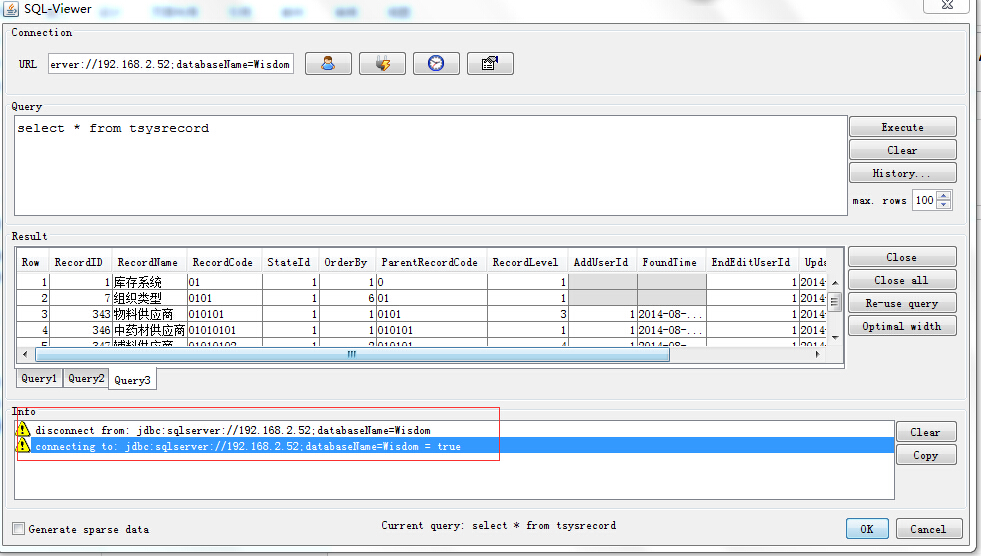
单击2 ：输入用户名，密码

单击3 : 连接数据库

单击4:输入sql

单击 execute ，可以看到结果了。

最下面的info 栏，其实也是可以给出连接提示的。



懂点英文的都看的明白。

## Weka源代码导入项目

因为接下来的开发工作很大程度上得参考weka的源代码，所有必须得把weka的源代码跑起来。

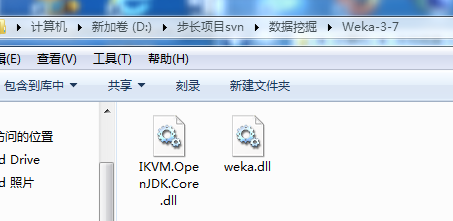
1. 下载eclipse，32位的eclipse最新版下一个就行。（我的jdk是32位的，所以eclipse也得是32位的，当然如果jdk64位的，当然eclipse也得是64位了）
2. 解压即可（所有的java程序都是解压就能用的）。
3. 在Eclipse下新建工程，可以命名为weka,在src下新建名为weka的包；
4. 在此项目下import-->File System-->选择.../weka/src/main/java/weka，并全部导入；
5. 导入库文件，在项目上右键，biuldpath-->add external Archive-->选择lib里的

java-cup.jar  JFlex.jar  junit.jar packageManager.jar;

1. 运行weka.gui.main即可；

# 五、开发说明

首先项目需要引入两个dll



Weka.dll 其实就是拿weka的java源代码，由IKVM 转化而来的。这也是为什么要同时引入IKVM 的dll的原因。

　IKVM转换weka.jar的命令是：ikvmc -target:library weka.jar。运行后，会产生一个新的文件weka.dll。

IKVM 是绿色的，下载解压就行。

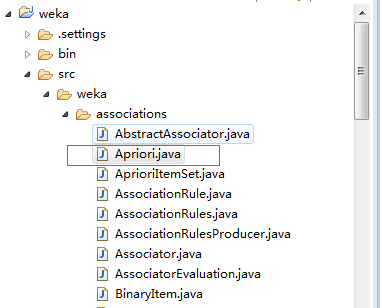
从arff文件中加载数据

new weka.core.Instances(newjava.io.FileReader("G:\\test.arff"));

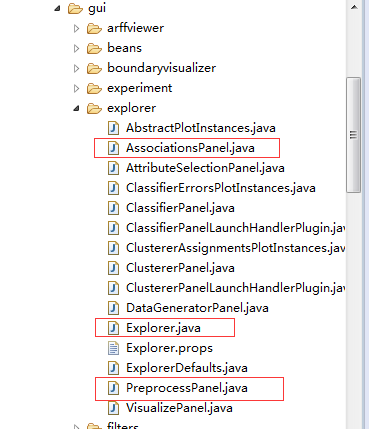
保存数据至arff文件

DataSink.write(‘/some/where/data.arff’,data)

Apriori 算法代码



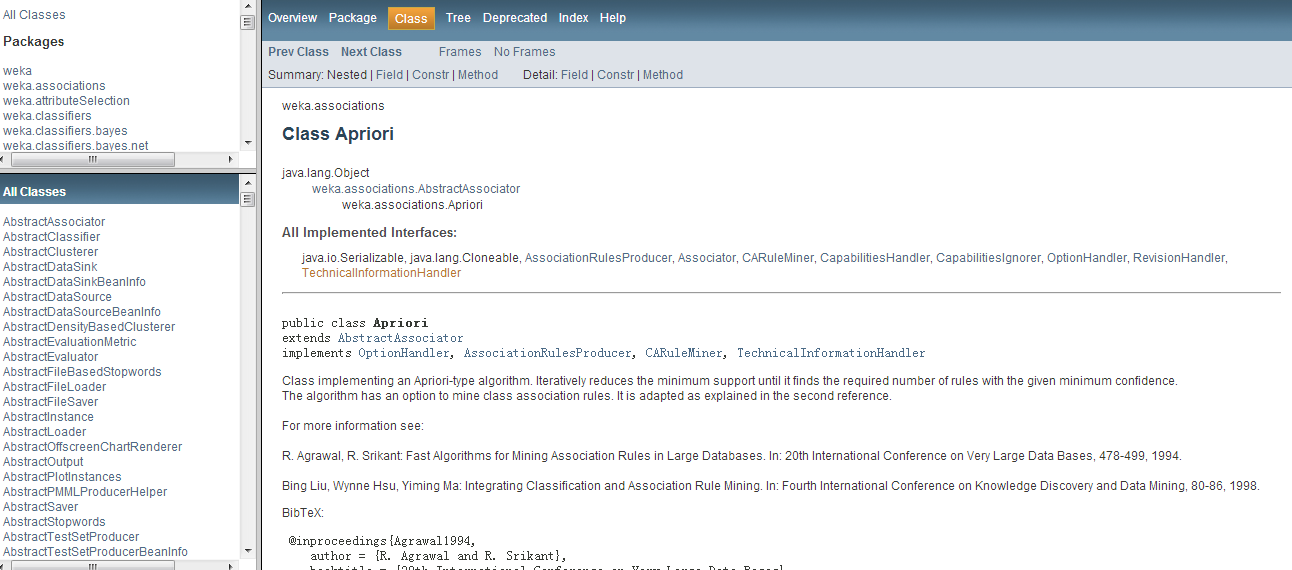
Explorer 代码



我们其实只用关注这3个代码文件就行了。

基本功能代码都在这3个代码文件中。

其它功能请参考api文档



# 六、回归分析

## 定义

回归分析属于分类的一种，分类：得到一个分类函数或分类模型（即分类器），通过分类器将未知的数据对象映射到某个给定的类别（回归是某个值）。

分类可分为两步：1、建立模型

2、使用模型对数据对象进行分类

分类是预测离散的值，回归是预测连续的值

所有回归模型均符合同一个通用模式。多个自变量综合在一起可以生成一个结果 — 一个因变量。然后用回归模型根据给定的这些自变量的值预测一个未知的因变量的结果。

一个例子就是给房子定价。房子的价格（因变量）是 很多自变量 — 房子的面积、占地的大小、厨房是否有花岗石以及卫生间是否刚重装过等的结果。所以，不管是购买过一个房子还是销售过一个房子，您都可能会创建一个回归模型 来为房子定价。这个模型建立在邻近地区内的其他有可比性的房子的售价的基础上（模型），然后再把您自己房子的值放入此模型来产生一个预期价格。

对于回归模型，只能是 NUMERIC

在有监督(supervise)的机器学习中，数据集常被分成2~3个，即：训练集(train set) 验证集(validation set) 测试集(test set)。**我们这没有验证集。**

其中训练集用来估计模型，而测试集则检验最终选择最优的模型的性能如何。二部分都是从样本中随机抽取。

样本少的时候，对N个样本采用K折交叉验证法。就是将样本打乱，然后均匀分成K份，轮流选择其中K－1份训练，剩余的一份做测试，计算预测误差平方和，最后把K次的预测误差平方和再做平均作为选择最优模型结构的依据。特别的K取N，就是留一法（leave one out）

## 实例1

在预处理面板中加载cpu.arff 文件

单击 **Classify** 选项卡

单击 Choose 按钮，然后扩展 functions 分支。

选择 LinearRegression 叶。

Test options 测试选项的解释如下

Use training set //用当前指定的数据集中的全部数据进行学习，数据都是训练集，同时也是测试集

Suplied test set //另外指定一个测试集

Cross-validation //交叉验证，在给定的建模样本中，拿出大部分样本进行建模型，留小部分样本用刚建立的模型进行预报，并求这小部分样本的预报误差，记录它们的平方加和。这个过程一直进行，直到所有的样本都被预报了一次而且仅被预报一次。需要输入折数，就是平均分成几份的意思。

**Percentage split //按比例分割 在数据集中，取出特定比例的数据用于训练（训练集）。其余的用来测试（测试集）**

**选择好**因变量class，默认就是

**选择第一项的分析结果**

=== Run information ===

Scheme: weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8

Relation: cpu

Instances: 209

Attributes: 7

MYCT

MMIN

MMAX

CACH

CHMIN

CHMAX

class

Test mode: evaluate on training data 基于训练数据的评估

=== Classifier model (full training set) === 分类器形式 full training set

Linear Regression Model //模型如下，就是一个公式

class =

0.0491 \* MYCT +

0.0152 \* MMIN +

0.0056 \* MMAX +

0.6298 \* CACH +

1.4599 \* CHMAX +

-56.075

Time taken to build model: 0.04 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correlation coefficient 0.93 //相关系数

Mean absolute error 37.9748 //平均绝对误差

Root mean squared error 58.9899 //均方根误差

Relative absolute error 39.592 % //相对绝对误差

Root relative squared error 36.7663 % //相对均方根误差

Total Number of Instances 209 //实例总数

**选择第三项的分析结果**

=== Run information ===

Scheme: weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8

Relation: cpu

Instances: 209

Attributes: 7

MYCT

MMIN

MMAX

CACH

CHMIN

CHMAX

class

Test mode: 10-fold cross-validation

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

Linear Regression Model

class =

0.0491 \* MYCT +

0.0152 \* MMIN +

0.0056 \* MMAX +

0.6298 \* CACH +

1.4599 \* CHMAX +

-56.075

Time taken to build model: 0.01 seconds

=== Cross-validation ===

=== Summary ===

Correlation coefficient 0.9012

Mean absolute error 41.0886

Root mean squared error 69.556

Relative absolute error 42.6943 %

Root relative squared error 43.2421 %

Total Number of Instances 209

**选择第四项的分析结果**

=== Run information ===

Scheme: weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8

Relation: cpu

Instances: 209

Attributes: 7

MYCT

MMIN

MMAX

CACH

CHMIN

CHMAX

class

Test mode: split 66.0% train, remainder test

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

Linear Regression Model

class =

0.0491 \* MYCT +

0.0152 \* MMIN +

0.0056 \* MMAX +

0.6298 \* CACH +

1.4599 \* CHMAX +

-56.075

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on training split: 0 seconds

=== Summary ===

Correlation coefficient 0.9158

Mean absolute error 38.1617

Root mean squared error 48.9672

Relative absolute error 45.5102 %

Root relative squared error 46.332 %

Total Number of Instances 71 （参与测试的数据数量 209\*34%）

## 实例2

**M5P 算法**

**一种很实用的回归算法。这个算法是决策树和线性回归算法的一个结合体。**

举一个例子向大家说明。

预测 处理器芯片的价格。 处理器芯片分为电脑使用的通用芯片（代表是Intel 和 AMD）和手机、嵌入式设备使用的芯片（ARM芯片）。即使两种芯片的其他参数大体一致，通用芯片和嵌入式芯片的价格还是会有很大不同。如果在生成回归模型的时候把这两类芯片的数据混为一谈，那么生成出来的线性模型的误差必定会变得很大。如果能用决策树算法把样本预先分成两类（通用和嵌入式），然后对两个分类分别进行回归算法，那么回归模型就会好很多。

在trees条目下找到M5P算法。

其它都一样

结果如下：

=== Run information ===

Scheme: weka.classifiers.trees.M5P -M 4.0

Relation: cpu

Instances: 209

Attributes: 7

MYCT

MMIN

MMAX

CACH

CHMIN

CHMAX

class

Test mode: evaluate on training data

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

M5 pruned model tree:

(using smoothed linear models)

CHMIN <= 7.5 : LM1 (165/12.903%)

CHMIN > 7.5 :

| MMAX <= 28000 :

| | MMAX <= 13240 :

| | | CACH <= 81.5 : LM2 (6/18.551%)

| | | CACH > 81.5 : LM3 (4/30.824%)

| | MMAX > 13240 : LM4 (11/24.185%)

| MMAX > 28000 : LM5 (23/48.302%)

LM num: 1 CHMIN <= 7.5 的前提下使用该算法，下面的类似

class =

-0.0055 \* MYCT

+ 0.0013 \* MMIN

+ 0.0029 \* MMAX

+ 0.8007 \* CACH

+ 0.4015 \* CHMAX

+ 11.0971

LM num: 2

class =

-1.0307 \* MYCT

+ 0.0086 \* MMIN

+ 0.0031 \* MMAX

+ 0.7866 \* CACH

- 2.4503 \* CHMIN

+ 1.1597 \* CHMAX

+ 70.8672

LM num: 3

class =

-1.1057 \* MYCT

+ 0.0086 \* MMIN

+ 0.0031 \* MMAX

+ 0.7995 \* CACH

- 2.4503 \* CHMIN

+ 1.1597 \* CHMAX

+ 83.0016

LM num: 4

class =

-0.8813 \* MYCT

+ 0.0086 \* MMIN

+ 0.0031 \* MMAX

+ 0.6547 \* CACH

- 2.3561 \* CHMIN

+ 1.1597 \* CHMAX

+ 82.5725

LM num: 5

class =

-0.4882 \* MYCT

+ 0.0218 \* MMIN

+ 0.003 \* MMAX

+ 0.3865 \* CACH

- 1.3252 \* CHMIN

+ 3.3671 \* CHMAX

- 51.8474

Number of Rules : 5

Time taken to build model: 0.11 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correlation coefficient 0.9728

Mean absolute error 21.9246

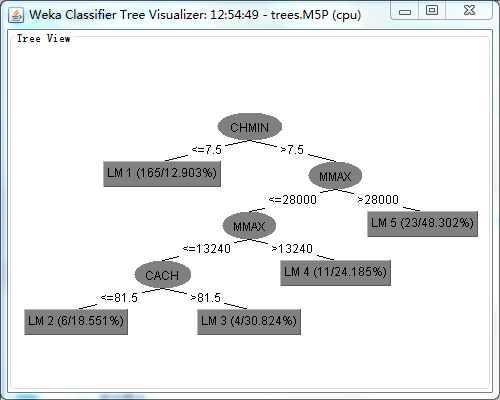
Root mean squared error 37.5095

Relative absolute error 22.8582 %

Root relative squared error 23.3784 %

Total Number of Instances 209

貌似m5p算法性能更优秀。而且它可以显示算法树



右键 result list 的条目，在菜单上选择 visualize tree。

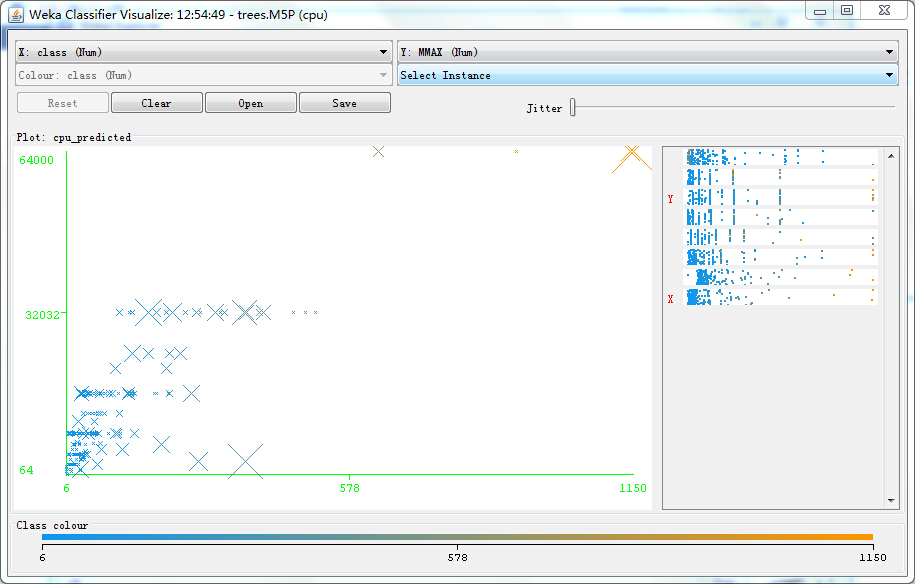
括号里的数字表示： 前一个是达到该节点的实例数量

后一个是标准偏差

## 比较算法优劣

右键 result list 的条目，在菜单上选择visualize classifier errors

弹出下面界面



叉叉代表一个数据点，越大代表误差越大，反之误差越小。所以评价算法的优劣一个直观的办法就是看谁小叉叉多，大叉叉少。 M5P算法大叉叉要少点。

# 七、代码分析

Weka 中待分析的数据是放在一个数据集中的：*Instances*

它包含着多个数据对象：*Instance*  ， 每个 *Instance* 就相当于是我们的一行记录。

Instances 可以从数据文件（\*.arff）中整个读取进来。也可以从数据库中整个读取（通过sql查询语句,但一般情况下是不可能的，因为数据格式不兼容），通常数据源是数据库的话，我们还是得建立 instance（需要格式转换等一系列操作），然后一个一个插入instances

Instances 需要指定类字段（后面的分析需要）

Instances 有很多*Attribute*，就是属性，相当于是字段属性

各种算法类（关联算法、回归算法）都实现了 OptionHandler 接口，我们可以通过这个接口来获取、设置 算法参数。

每个算法都有一个父类（Apriori 算法的父类：Associator

LinearRegression、m5p 算法的父类 Classifier）

可以通过Classifier 来执行他们公共的方法