WEKA 3-5-3 Experimenter 指南

原文版本3. 5 . 3
原文链接
翻译 王娜
校对 C6H5NO2
Pentaho 中文讨论组
QQ 群：12635055
论坛：http://www.bipub.org/bipub/index.asp

David Scuse
Peter Reutemann

June 8, 2006

1 简介.....1
2 标准试验.....2
2.1 简单模式.....2
2.1.1 新试验.....2
2.1.2 结果的目的文件.....2
2.1.3 试验类型.....4
2.1.4 数据集.....5
2.1.5 迭代控制.....6
2.1.6 算法.....6
2.1.7 保存设置.....8
2.1.8 运行试验.....8
2.2 高级模式.....9
2.2.1 定义试验.....9
2.2.2 运行试验.....12
2.2.3 改变试验参数.....13
2.2.4 其他结果的产生.....19
3 远程试验.....23
3.1 准备.....23
3.2 数据库服务器设置.....23
3.3 远程引擎安装.....23
3.4 配置 Experimenter.....24
3.5 疑难问题解答.....24
4 分析结果.....25
4.1 设置.....25
4.2 保存结果.....28
4.3 改变基准算法.....28
4.4 统计显著性.....29
4.5 描述性检验.....29
4.6 排序检验.....29
5 参考文献.....30

1 简介

Weka 试验（Experiment）环境可以让用户创建，运行，修改和分析算法试验，这也许比单独的分析各个算法更加方便。例如，用户可创建一次试验，在一系列数据集上运行多个算法（schemes），然

后分析结果以判断是否某个算法比其他算法（在统计意义下）更好。

可以通过 Simple CLI 在命令行的方式下运行试验环境。例如，在 CLI 上键入以下命令，将通过一个基本的训练和测试步骤在 Iris 数据集上运行 OneR 算法。（注意该命令应放在同一行中输入CLI。）

```
java weka.experiment.Experiment -r -T data/iris.arff
 –D weka.experiment.InstancesResultListener
 –P weka.experiment .RandomSplitResultProducer --
 –W weka. experiment .ClassifierSplitEvaluator --
 –W weka. classifiers. rules. OneR
```

然而直接把命令直接输入 CLI 这种方式并不是很方便，且试验不容易修改。

Experimenter 有两种模式：一种具有较简单的界面，并提供了试验所需要的大部分功能，另一种则提供了一个可以使用 Experimenter 所有功能的界面。你可使用 Experiment Configuration Mode 单选按钮在这两者间进行选择。

- Simple
- Advanced

在两种模式下，你都进行在本地单一机器上的标准试验，或者分布在几台主机上的远程试验。分布式的试验减少了完成试验本身所需的时间，但是另一方面，设置这样的试验需要更多的时间。

以下的章节节将介绍标准试验(包括simple 和 advanced模式)，然后是远程试验，最后部分是结果的分析。

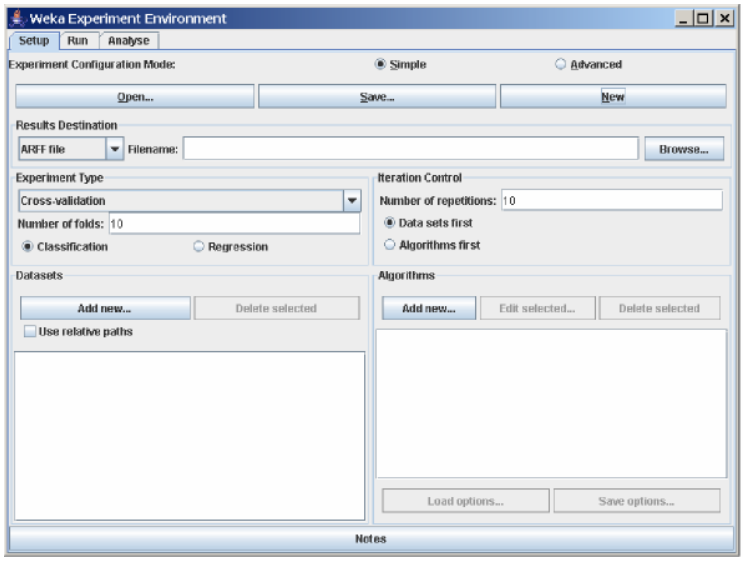
这个手册也可在 WekaDoc Wiki [5] 上找到。

2 标准试验

2.1 Simple（简单）模式

2.1.1 新试验

在点击 New 后，就定义了一次试验的默认参数。



2.1.2 Result Destination （结果的目的文件）

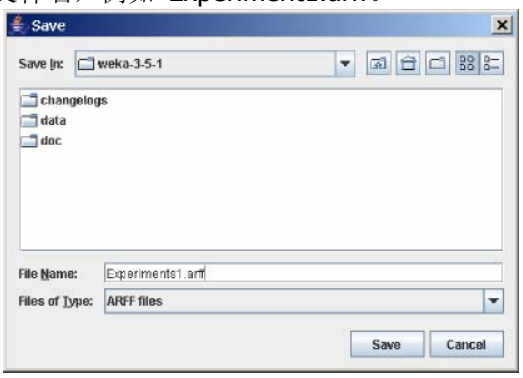
一个 ARFF 文件将默认作为结果输出的目的文件。但你也可选择：

- ARFF file （ARFF 文件）
- CSV file （CSV 文件）
- JDBC database （JDBC 数据库）

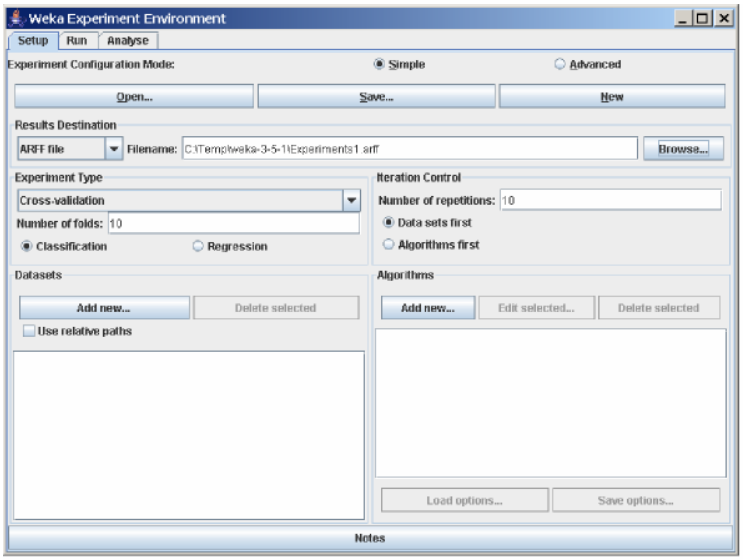
以下章节将详细讨论 ARFF 文件和 JDBC 数据库。CSV 类似于 ARFF，但它可以用其他的电子表格程序加载。

2.1.2.1 ARFF file

如果文件名为空，将在系统的 TEMP 目录下创建一个临时文件。如果你想显式的指定一个结果文件，只需点击 Browse，并选一个文件名，例如 Experiment1.arff。



点击 Save，文件路径将出现在 ARFF file 旁的文本框中。

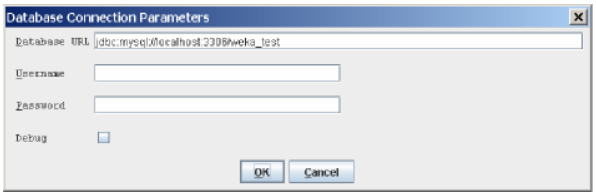


ARFF 或 CSV 文件的优点是它们的创建不需要 Weka 之外的类文件。它们的缺点则是试验一被中断就无法继续进行，所谓中断包括出现错误，添加数据集或添加算法。尤其对于那些相当耗时的试验，这一不足会增加很多麻烦。

2.1.2.2 JDBC database

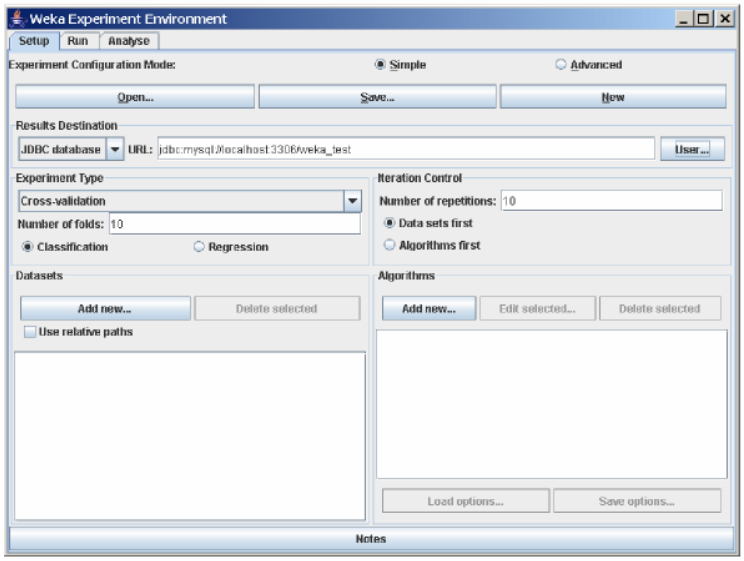
有了 JDBC，就可以很容易的把结果存储在数据库中。要使用某种特定数据库的 JDBC 功能，必须在CLASSPATH 中指定相应的 jar 文件。

把 ARFF file 改成 JDBC database 后，点击 User... 来指定访问数据库的 JDBC URL 和用户帐号。



在提供了必要的数据并点击 OK 后，主窗口中的 URL将会更新。

注意：这个时候还没有测试数据库连接；启动试验时才会进行连接测试。

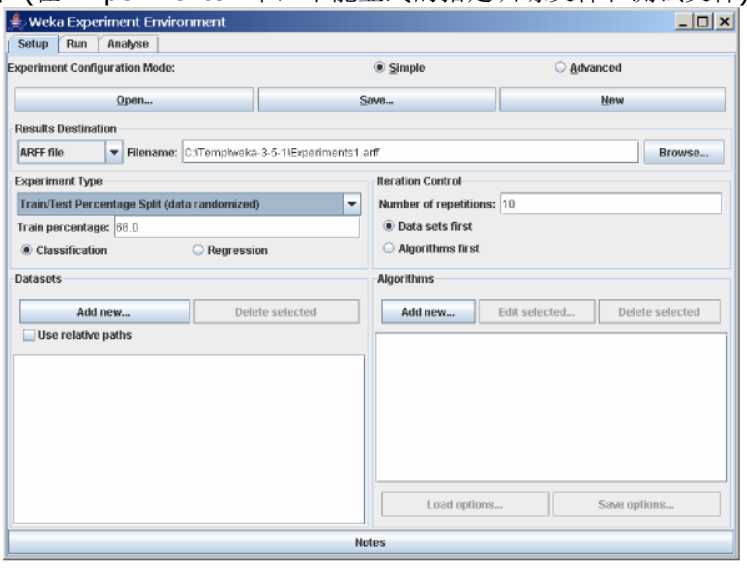


JDBC 数据库的优点是可以继续运行那些被中止的或扩展了的试验。它不用重新运行那些已试验过的算法/数据集组合，而仅计算还没有被试验的那些。

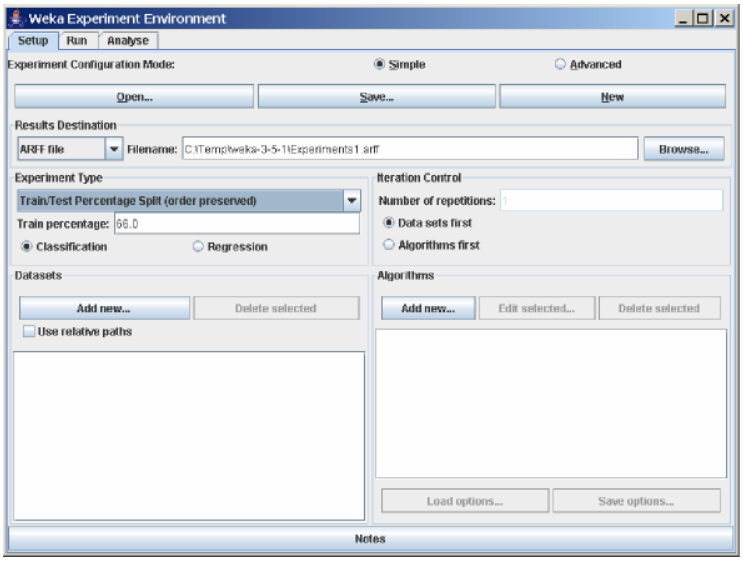
2.1.3 Experiment type（试验类型）

用户可选择以下三种不同的类型：

- Cross-validation （交叉验证） (默认)：
  根据给定的折数执行分层交叉验证
- Train/Test Percentage Split (data randomized) （按比例分割训练/测试集，随机挑选数据）：
  把数据打乱顺序并确定层次后，根据给定的百分比把这个数据集分割成一个训练文件和一个测试文件 (在 Experimenter 中，不能显式的指定训练文件和测试文件)



- Train/Test Percentage Split (order preserved) （按比例分割训练/测试集，按顺序挑选数据）：
  因为不能显式的指定训练/测试文件对， 可以利用这个试验类型把合并过的训练和测试文件还原 (只需找到正确的比例)

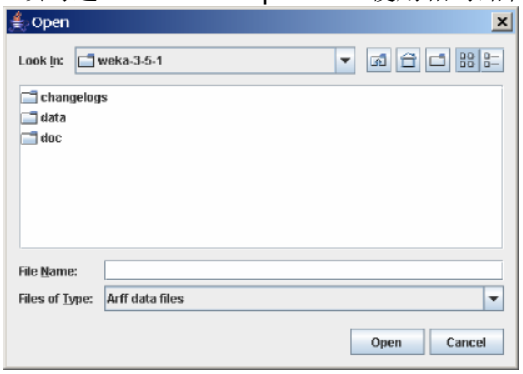


而且，可在 Classification（分类，又称判别）和 Regression（回归）间进行选择，这依赖于所用的数据集和分类器[1]（classifiers）。对于像J48 (即 Quinlan 的 C4.5 算法 [3] 在 Weka 中的实现) 这样的决策树算法和 iris 数据集，Classification 是必需的；另一方面，对于 M5P 这样的数值型分类器，则需要选用Regression。默认选中的是 Classification。

注意：如果使用了按比例分割，必须确保修正过的成对 T 检验在给定的比值下仍能产生有意义的结果 [2]。

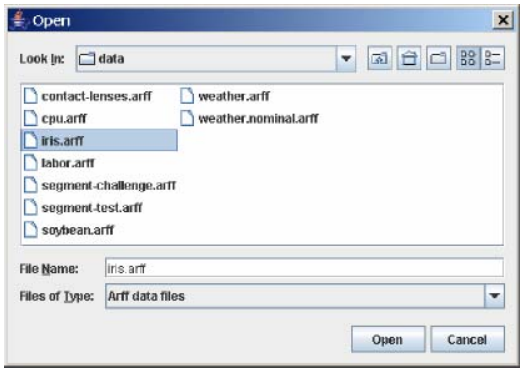
2.1.4 Datasets （数据集）

可以通过绝对路径或相对路径添加数据集文件。后者使得在不同的机器上运行试验更加方便，因此你在点击 Add new....之前，应该勾选 Use relative paths （使用相对路径）。

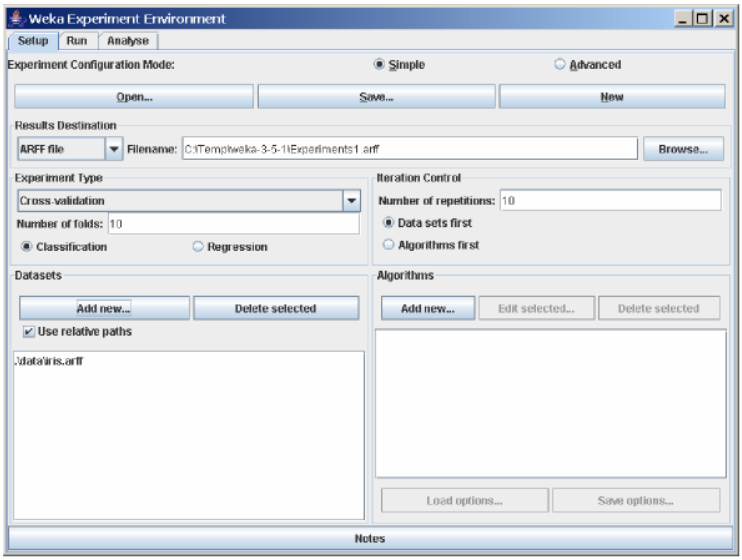


在这个例子中，打开 data 目录，选择 iris.arff 数据集。

[1] WEKA 把用于分类和回归的算法都叫做分类器--译注。



在点击 Open 后，文件将显示在数据集列表中。如果选中一个目录点击 Open，那么将递归的添加所有 ARFF 文件。从列表删除文件时，可选中那些文件，然后点击 Delete selected。



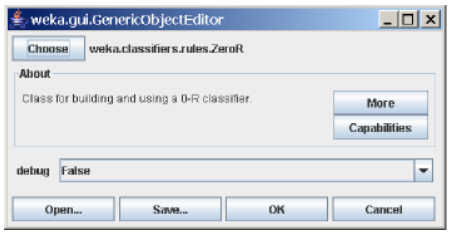
2.1.5 Iteration control （迭代控制）

- Number of repetitions （重复次数）：
  为了获得统计上有意义的结果，默认的迭代数量是 10。在10折交叉验证的情形下，这意味着对一个分类器要进行100次调用——从训练集计算它，并在测试集上测试。
- Data sets first/Algorithms first （数据集优先/算法优先）：

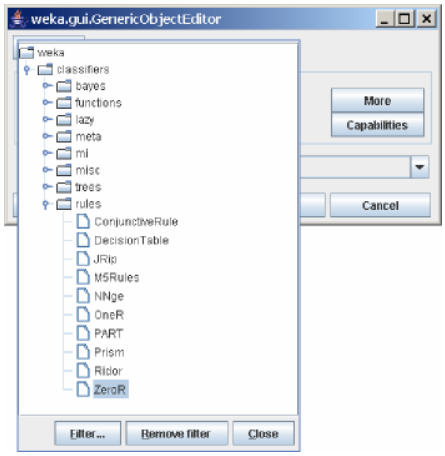
当存在多个数据集和算法的时候，切换成优先迭代数据集的模式可能会有用。举个例子，会有人把结果存储在数据库中，并且想尽早完成某个算法在所有数据集上的结果。

2.1.6 Algorithms （算法）

可以通过 Add new... 按钮添加新算法。如果是第一次打开这个对话框，将出现 ZeroR；否则将出现上次选中的那个。



可以用 Choose 按钮打开 GenericObjectEditor 来选择别的分类器。

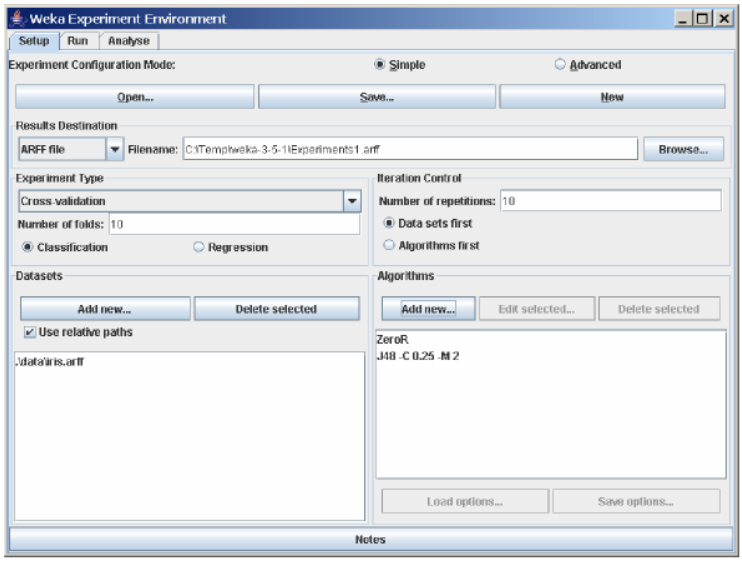


有的分类器仅针对某种特定类型的属性（attribute）和目标属性（class），使用Filter... 按钮能够加亮显示它们。点击Remove filter，加亮显示又会被取消。

可使用 Add new... 按钮继续添加其他的算法，如 J48 决策树。



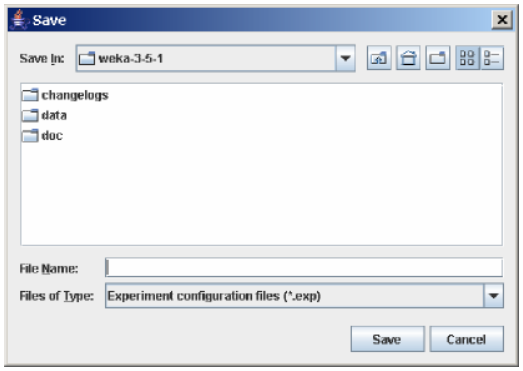
在设置好分类器的参数后，可点击 OK 将之添加进算法列表。



使用 Load options... 和 Save options... 按钮，你可从 XML 加载或保存选中分类器的设置。这对配置相当复杂的分类器(如 nested meta-分类器)尤其有用，因为手动设置它们需要一些时间，却又经常要用到。

2.1.7 保存设置

为了将来能重复使用，可将试验的当前设置保存进一个文件，点击窗口顶部的 Save... 即可。

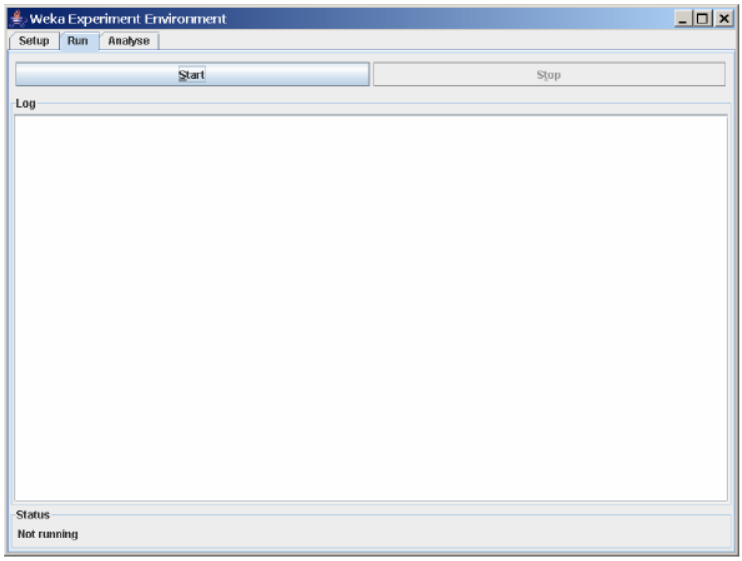


试验文件默认的的格式是 Java 序列化提供的二进制文件。这个格式的缺点是不同版本的 Weka 间可能存在格式的不兼容性。还有一种更加健壮的XML格式可供选择。

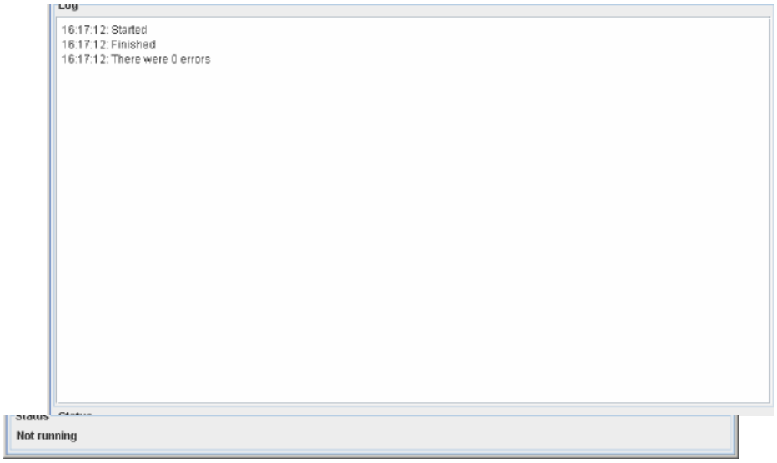
可通过 Open... 按钮重新装载之前保存的 experiments。

2.1.8 运行试验

要运行当前试验，需点击试验环境窗口中的 Run 标签页。当前试验将使用 ZeroR 和 J48 算法在 Iris 数据集上执行十次10折的分层交叉验证。



点击 Start 运行试验。

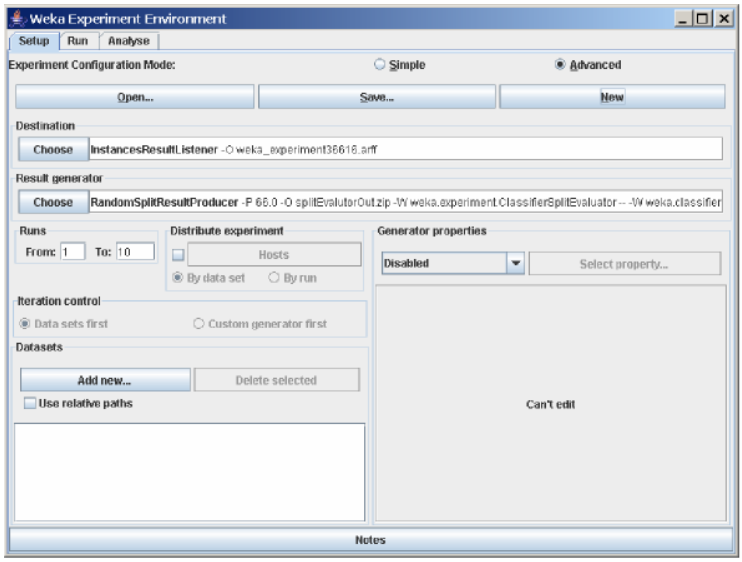


如果试验定义正确，在 Log 面板上将显示如上 3 条信息。试验结果保存在 Experiment1. arff 数据集里。

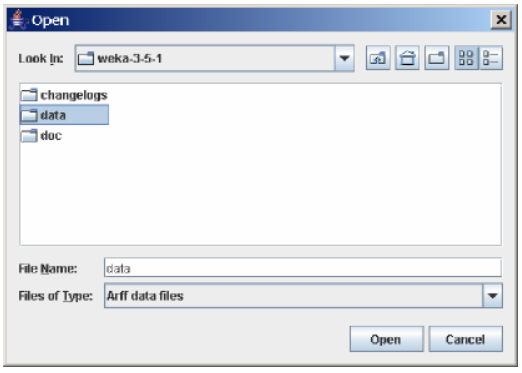
2.2 Advanced （高级）模式

2.2.1 定义试验

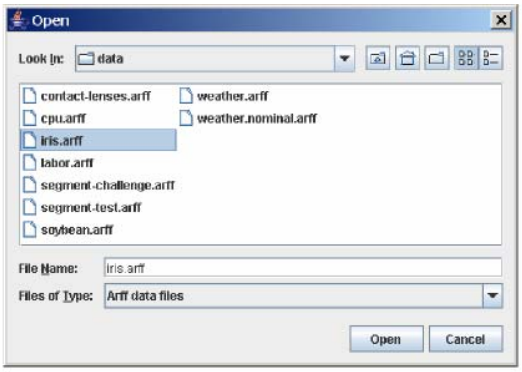
切换到 Setup 标签页，在高级模式下开始试验。点击 New 以初始化一次试验。 这样为试验给定了默认的参数。

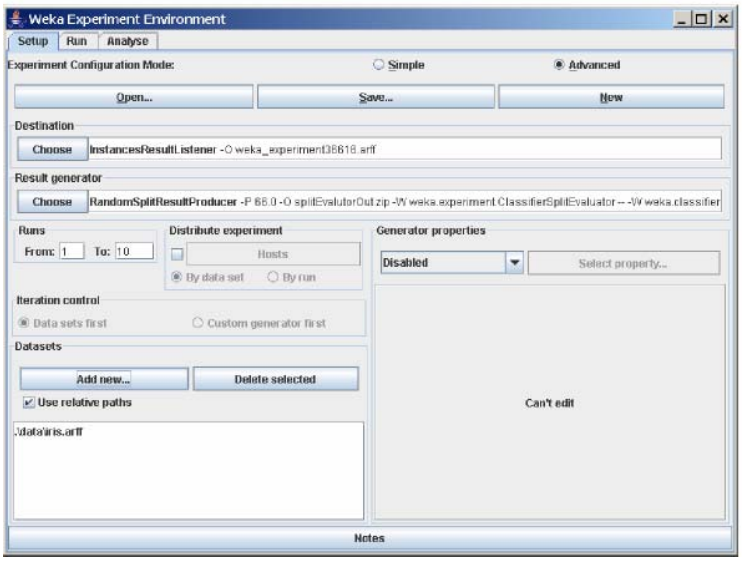


要给定由算法所处理的数据集，先在 Setup 标签页的 Datasets 面板上选择 Use relative paths，然后点击 Add new... 打开一个对话框窗口。



可以双击 data 文件夹查看可用的数据集，也可以浏览到其它的位置。选择 iris.arff，点击 Open 选择 Iris 数据集。

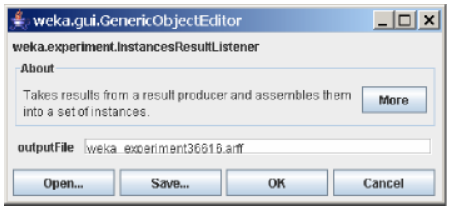


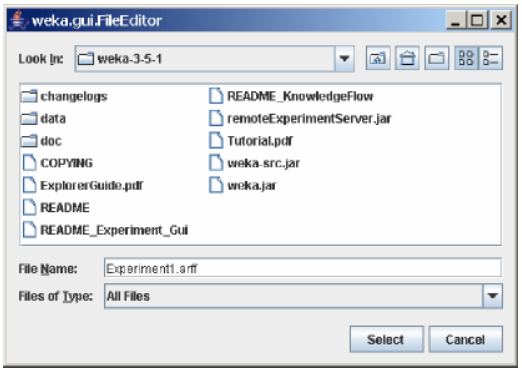


数据集名现在显示在 Setup 标签页的 Datasets 面板。

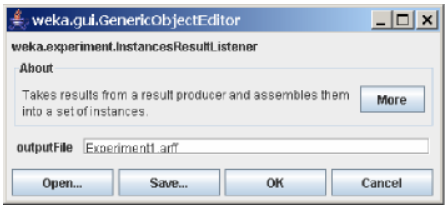
2.2.1.1 保存试验结果

要指定结果保存的数据集，点击 Destination面板上的 InstancesResultListener 条目。输出文件参数靠近窗口底部，在文本 outputFile 的旁边。点击这个参数显示一个文件选择窗口。

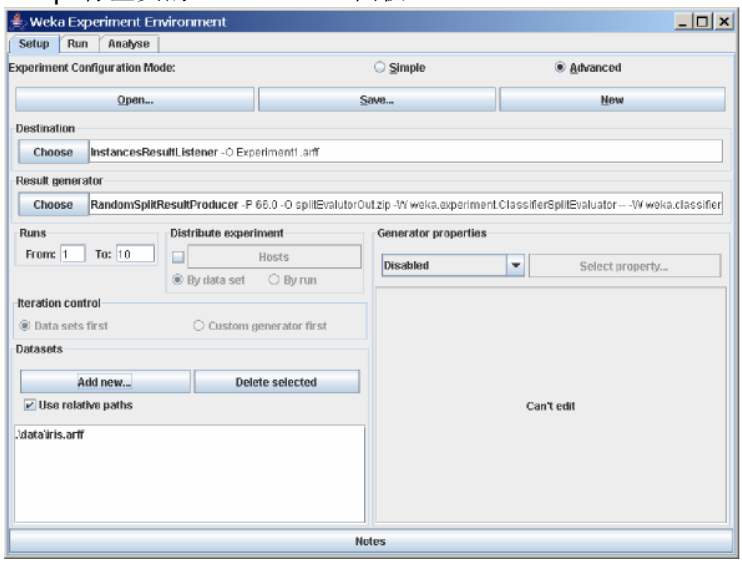




键入输出文件的名字，点击 Select，然后点击 close (x)。文件名显示在 outputFile 面板，点击 OK 关闭窗口。

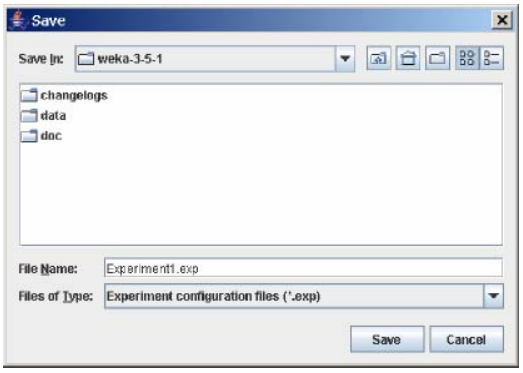


数据集名显示在 Setup 标签页的 Destination 面板。



2.2.1.2 保存试验定义

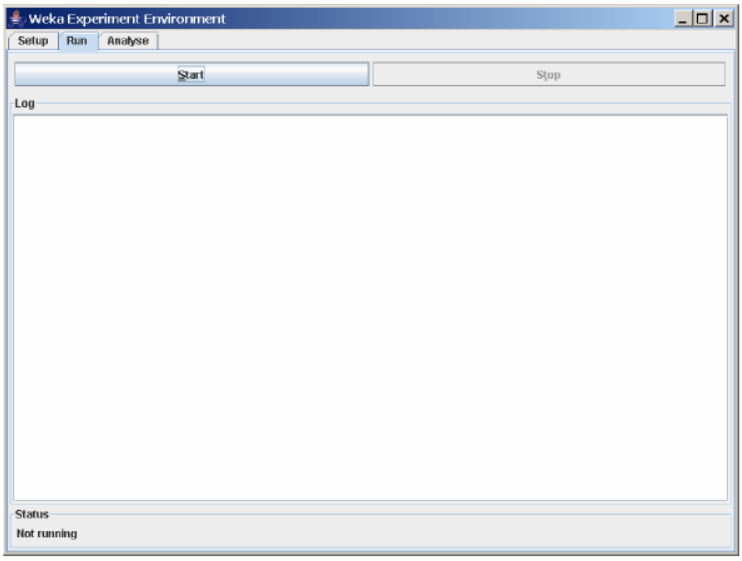
可在任何时候保存 experiment 定义。选择 Setup 标签页顶部的 Save即可。对于二进制文件，输入带有扩展名 exp 的数据集名(如果试验定义的数据集已存在，则选择这个数据集名) 或从 File of Type 复选框中选择试验配置文件(*,xml) (当版本变化时，XML 文件是健壮的)。



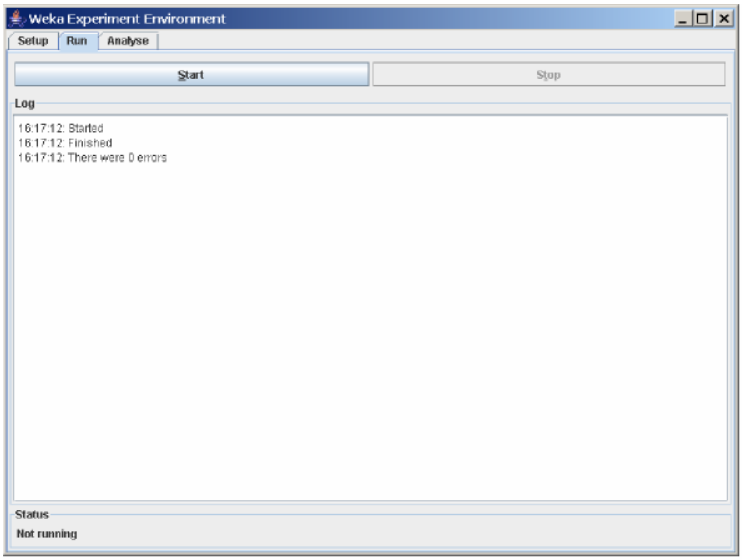
可在 Setup 标签页中选择 Open 恢复试验，然后在对话框窗口中选择 Experiment1.exp。

2.2.2 运行试验

为运行当前试验，点击试验环境窗口顶部的 Run 标签页。当前试验在 Iris 数据集上执行了 10 次随机化的训练和测试，66% 的数据样本用于训练，34% 用于测试，使用 ZeroR 算法。



点击 Start 运行试验。



如果试验定义正确，在 Log 面板顶部显示将 3 条消息。试验结果被保存在 Experiment1.arff 数据

集。这个 dataset 的前面几行显示如下：

```
@relation InstanceResultListener

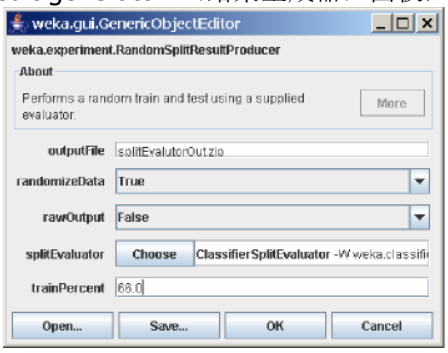
@attribute Key_Dataset {iris}
@attribute Key_Run {1,2,3,4,5,6,7,8,9,10}
@attribute Key_Scheme {weka. classifiers . rules . ZeroR,weka. classifiers. trees . J48} @attribute
Key_Scheme_options {,’-C 0. 25 -M 2’}
@attribute Key_Scheme_version_ID {48055541465867954, -217733168393644444} @attribute
Date_time numeric
@attribute Number_of_training_instances numeric
@attribute Number_of_testing_instances numeric
@attribute Number_correct numeric @attribute
Number_incorrect numeric @attribute
Number_unclassified numeric @attribute
Percent_correct numeric @attribute
Percent_incorrect numeric @attribute
Percent_unclassified numeric @attribute
Kappa_statistic numeric @attribute Mean_absolute_error
numeric @attribute Root_mean_squared_error numeric
@attribute Relative_absolute_error numeric @attribute
Root_relative_squared_error numeric
@attribute SF_prior_entropy numeric @attribute
SF_scheme_entropy numeric @attribute SF_entropy_gain
numeric @attribute SF_mean_prior_entropy numeric
@attribute SF_mean_scheme_entropy numeric @attribute
SF_mean_entropy_gain numeric @attribute
KB_information numeric
@attribute KB_mean_information numeric @attribute
KB_relative_information numeric @attribute True_positive_rate
numeric @attribute Num_true_positives numeric @attribute
False_positive_rate numeric @attribute
Num_false_positives numeric @attribute
True_negative_rate numeric @attribute
Num_true_negatives numeric @attribute
False_negative_rate numeric @attribute
Num_false_negatives numeric @attribute IR_precision
numeric
@attribute IR_recall numeric
@attribute F_measure numeric
@attribute Area_under_ROC numeric @attribute
Time_training numeric @attribute Time_testing
numeric
@attribute Summary {’Number of leaves: 3\nSize of the tree: 5\n’,
   ’Number of leaves: 5\nSize of the tree: 9\n’,
   ’Number of leaves: 4\nSize of the tree: 7\n’}
@attribute measureTreeSize numeric @attribute
measureNumLeaves numeric @attribute measureNumRules
numeric
@data

iris, 1 ,weka.classifiers .rules .ZeroR, ,48055541465867954,20051221 .033,99,51,
17,34,0,33.333333,66.666667,0,0,0.444444,0.471405,100,100,80.833088,80.833088,
0,1.584963,1.584963,0,0,0,0,1,17,1,34,0,0,0,0,0.333333,1,0.5,0.5,0,0,?,?,?,?
```

2.2.3 改变试验参数

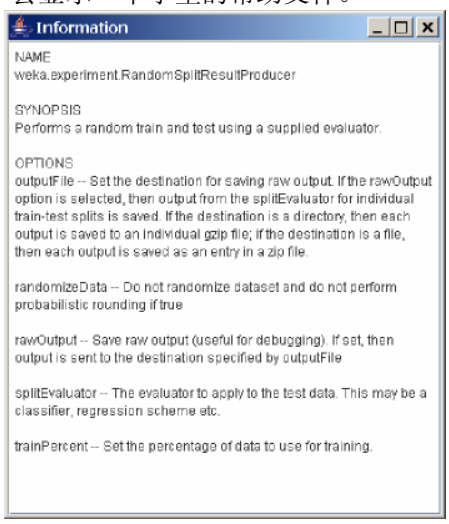
2.2.3.1　改变分类器

可通过点击 Result generator （结果生成器）面板，改变试验的参数。

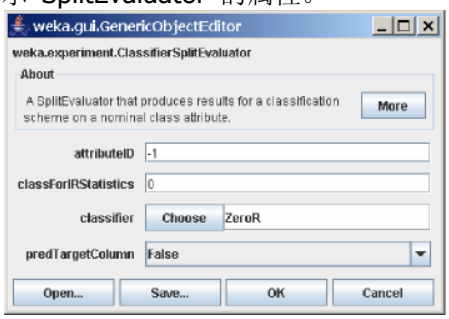


使用RandomSplitResultProducer （随机划分训练集和测试集的结果生成器）将重复的运行训练/测试。在 trainPercent 框中给出了用于训练的样本数目(用百分比表示)。(在 Setup 标签页的 Runs 面板中指定了运行次数)

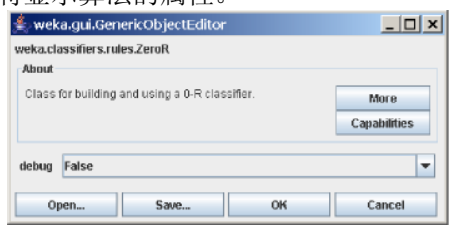
在 About 面板中点击 More，会显示一个小型的帮助文件。



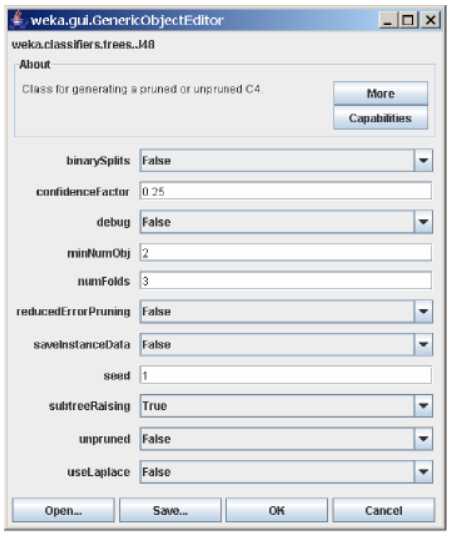
点击 splitEvaluator 条目将显示 SplitEvaluator 的属性。



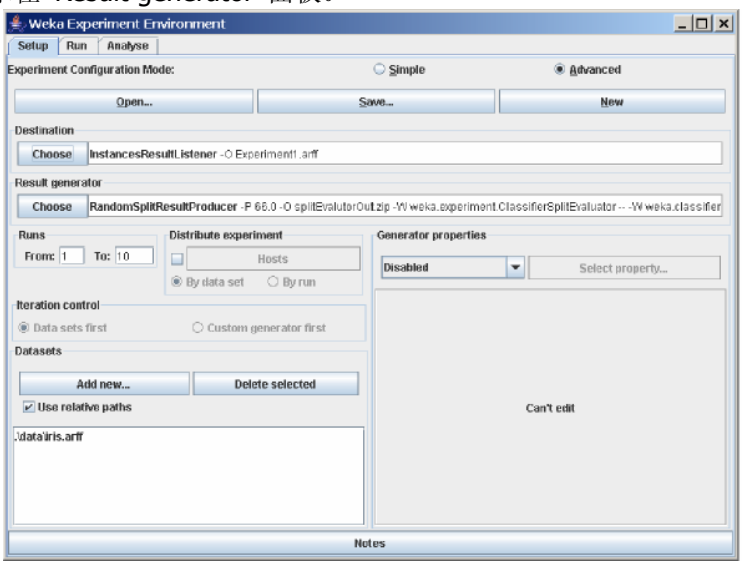
点击 classifier 条目(ZeroR) 将显示算法的属性。



这个算法没有可修改的属性(除了 debug mode on/off)，但大多数其他算法确实有可被用户修改的属性。Capabilities 按钮打开一个小型的对话框，列出了这个分类器可处理的所有属性和目标属性的类型。点击 Choose 按钮可选择一个不同的算法。以下窗口显示了 J48 决策树算法可用的参数。如果需要的话，在这里修改参数，然后点击 OK 关闭窗口。

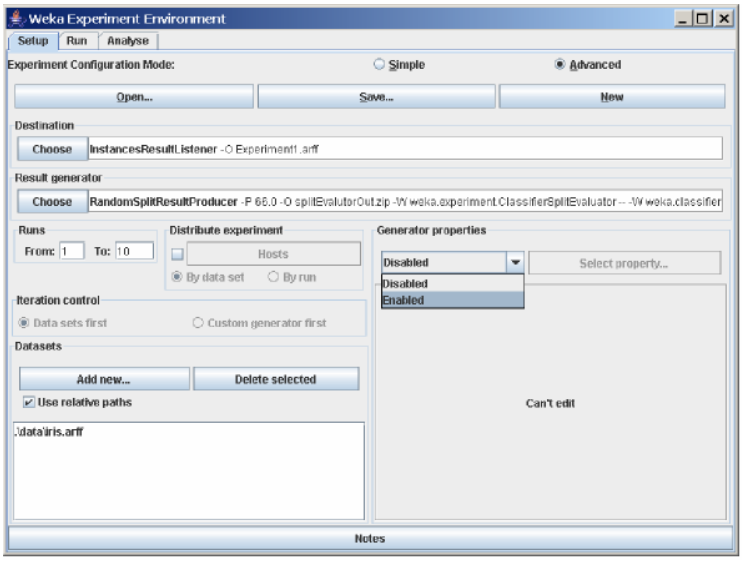


新算法的名称显示在 Result generator 面板。

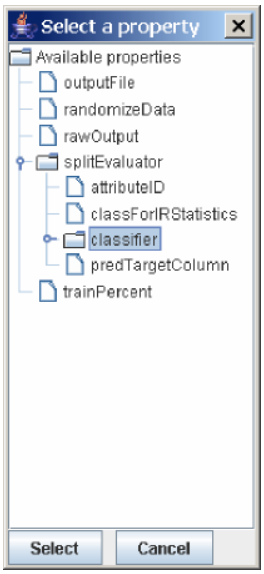


2.2.3.2 添加额外的算法

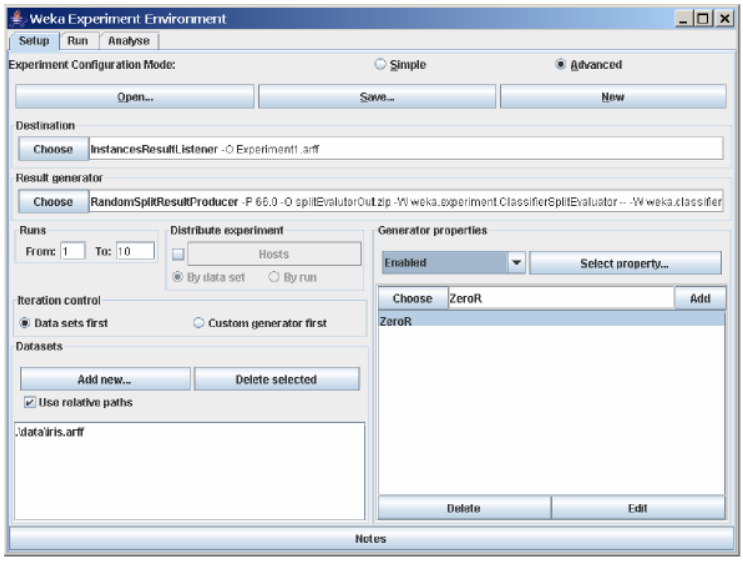
可在 Generator properties 面板中添加额外的算法。开始前，先将 Generator 属性面板中的下拉列表条目从 Disabled 变成 Enabled。



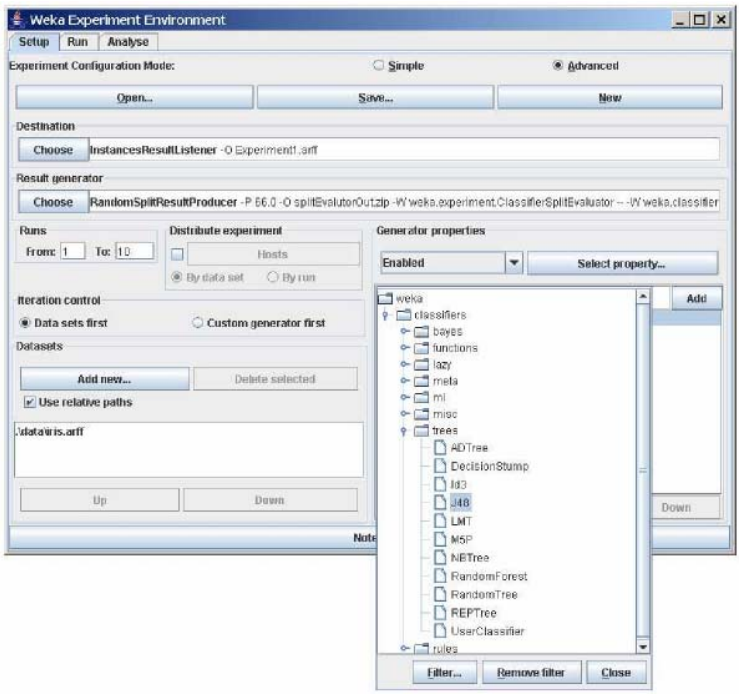
点击 Select property，展开 splitEvaluator，这样 classifier 条目就在属性列表上可见了，选中后点击Select。



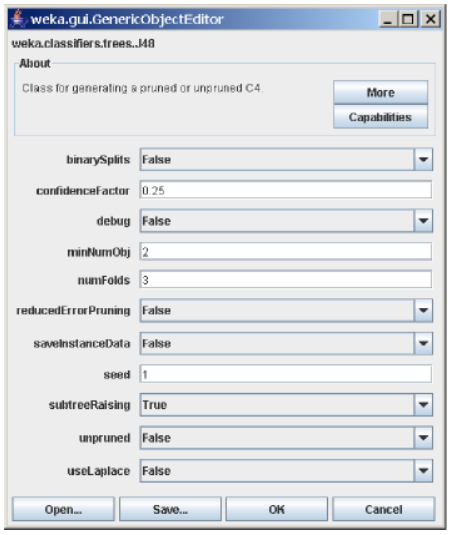
在 Generator properties 面板中显示了算法的名字。



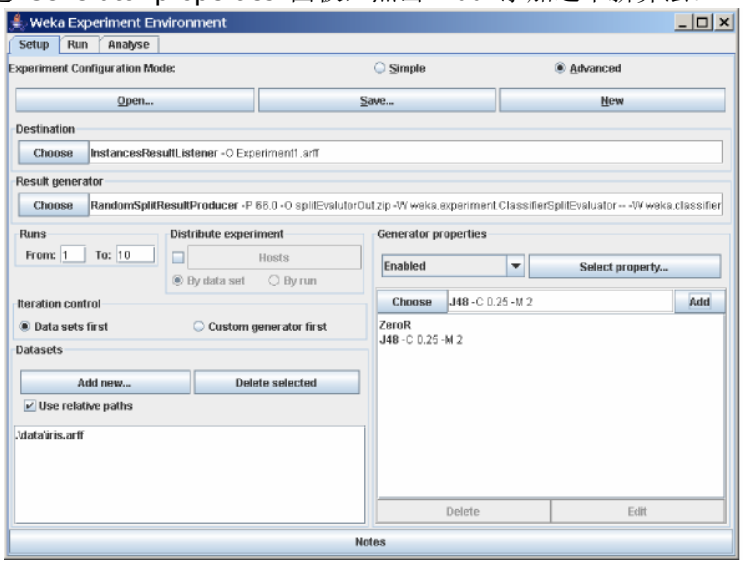
要添加另一个算法，点击 Choose 按钮以显示 GenericObjectEditor 窗口。



选择 J48 (在子树 trees 中) 切换到一个决策树算法。



新的算法被添加进 Generator properties 面板，点击 Add 添加这个新算法。



现在运行试验，两个算法的结果都将被生成。

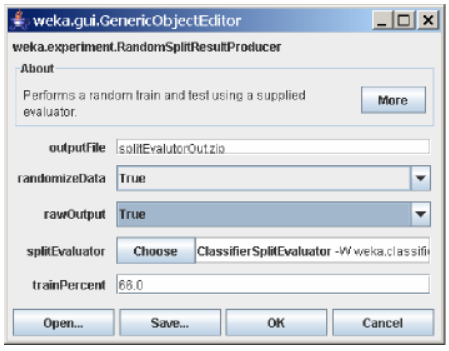
要添加更多的算法，重复上述步骤。要移除一个算法，点击选中它，然后点击 Delete。

2.2.3.3 添加额外的数据集

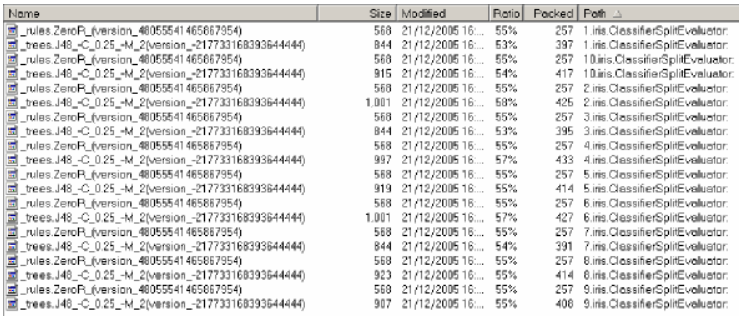
一次试验可以让算法运行于任意数量的数据集上。通过在 Datasets 面板点击 Add new... 添加额外的数据集。要从试验中删除数据集，可在选种那些待删除的数据集后点击 Delete Selected。

2.2.3.4 Raw Output （原始输出）

试验执行过程中生成的原始输出可保存进一个文件，之后进行检查。点击 Setup 标签页的 Result generator 面板打开 ResultProducer 窗口。



点击 rawOutput，从下拉列表中选择 True。输出默认被发送进 zip 文件 splitEvaluatorOut.zip。通过点击该窗口中的 outputFile 面板改变输出文件。现在运行试验时，每次运行的结果都将被记入文件，如下所示。



第一次运行的结果是：

```
ClassifierSplitEvaluator: weka.classifiers.trees.J48 -C 0.25 -M 2(version -217733168393644444)
    Classif ier model:
J48 pruned tree
------------------------

petalwidth <= 0.6: Iris-setosa (33.0)
petalwidth > 0.6
I petalwidth <= 1.5: Iris-versicolor (31.0/1.0) I petalwidth > 1.5:
Iris-virginica (35.0/3.0)

Number of Leaves : 3

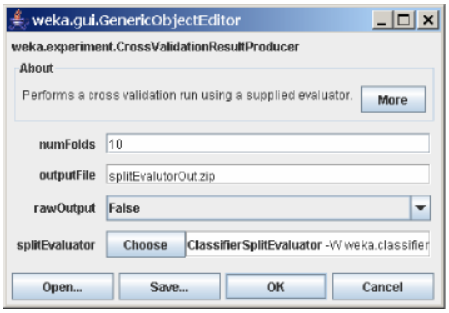
Size of the tree : 5

  Correctly Classified Instances          47               92.1569 °h
  Incorrectly Classified Instances         4                7.8431 °h
  Kappa statistic                          0.8824
  Mean absolute error                      0.0723
  Root mean squared error                  0.2191
  Relative absolute error                 16.2754 °h
  Root relative squared error             46.4676 °h
  Total Number of Instances               51
  measureTreeSize : 5.0
  measureNumLeaves : 3.0
  measureNumRules : 3.0
```

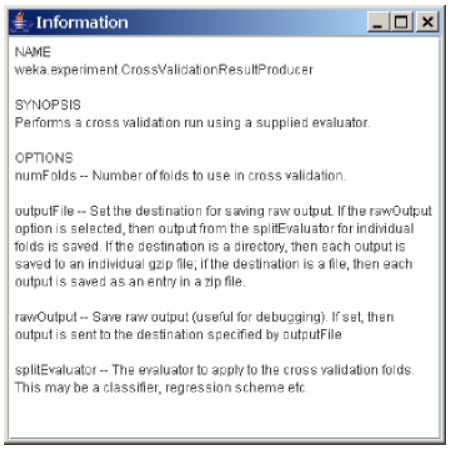
2.2.4 其他结果的产生

2.2.4.1 Cross-Validation Result Producer （交叉验证的结果生成器）

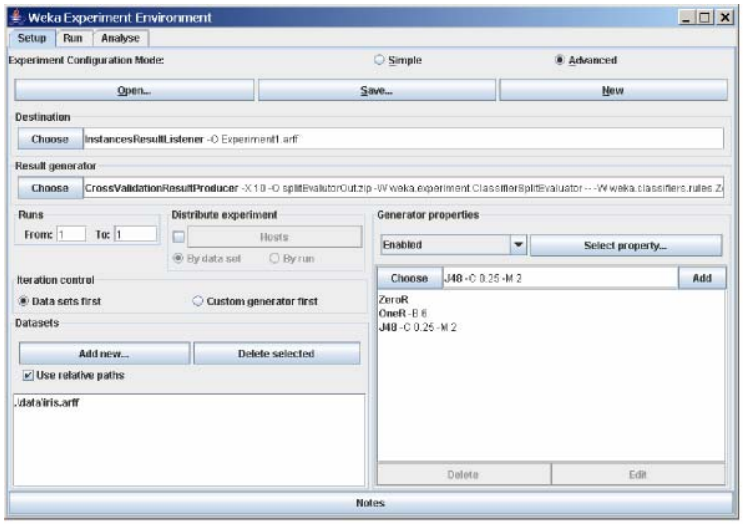
为从随机划分训练集和测试集的试验切换到交叉验证试验，点击 Result generator 条目。在窗口顶部，点击下拉列表，选择 Cross-ValidationResultProducer。窗口现在包含交叉验证专有的参数，例如它的折数。试验将执行10折交叉验证，而不是像上面例子中那样分开训练集和测试集。



Result generator 面板现在告诉我们将执行交叉验证。点击 More 生成关于 Cross-ValidationResultProducer 的简短描述。

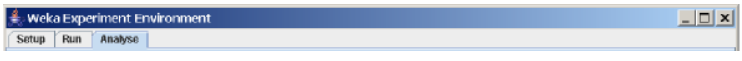


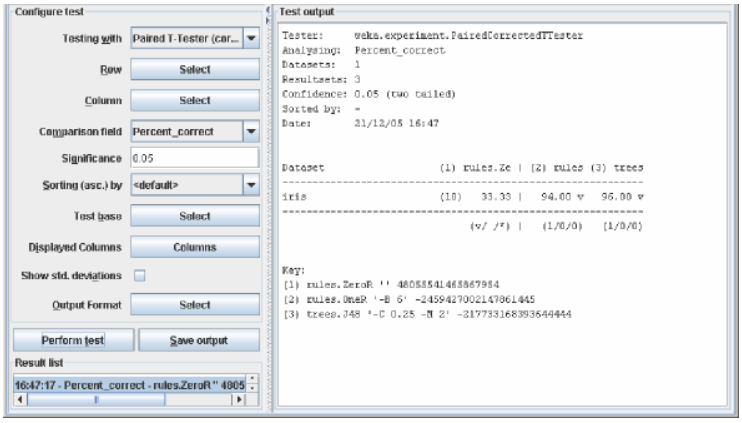
和 RandomSplitResultProducer 一样，在交叉验证期间可以运行多个算法，只需将它们添加进 Generator properties 面板即可。



在这个例子中，运行的次数在 Setup 标签页被设置成 1，因此对于每个算法和数据集，仅执行一次交叉验证。

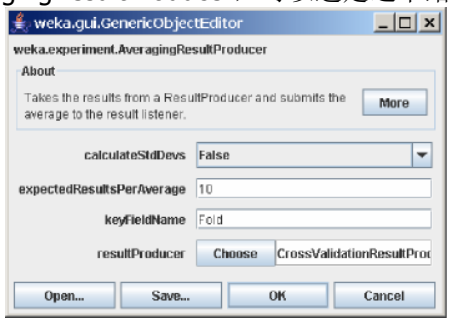
当分析这个例子时，将有下图所示的结果。注意这里处理了 30 个(1 次运行×10 折交叉验证× 3 个算法) 结果行。



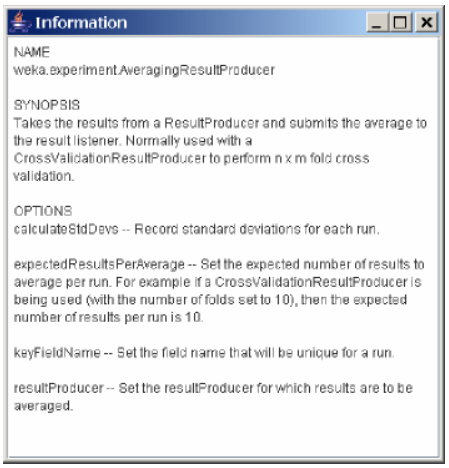


2.2.4.2 Averaging Result Producer （平均化的结果生成器）

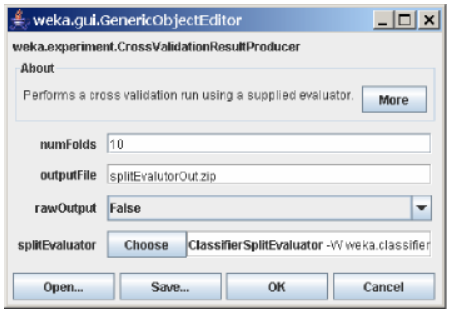
Cross-ValidationResultProducer 的一个替换物是 AveragingResultProducer。这个结果生成器取了一组试验(通常是 cross-validation 试验)运行的平均值。点击 Result generator面板，然后从 GenericObjectEditor 选择 AveragingResultProducer，可以选定这个结果生成器。



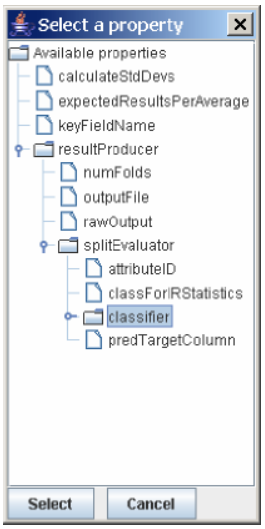
下图显示相关的帮助文件。

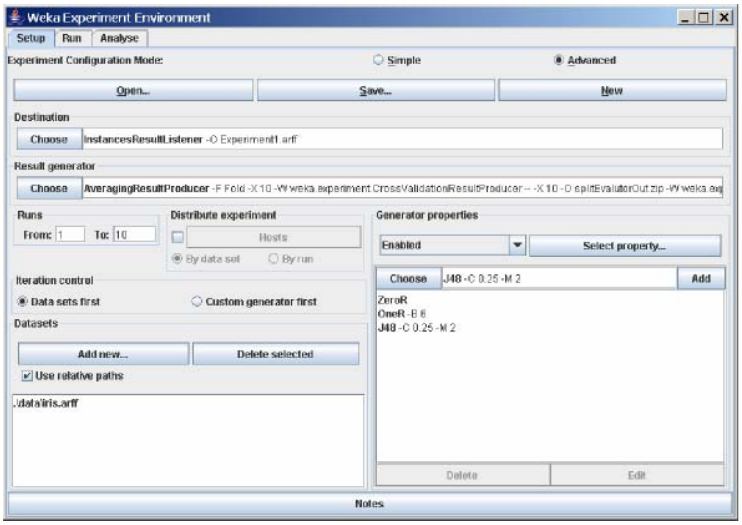


点击 resultProducer 面板，会出现以下窗口。

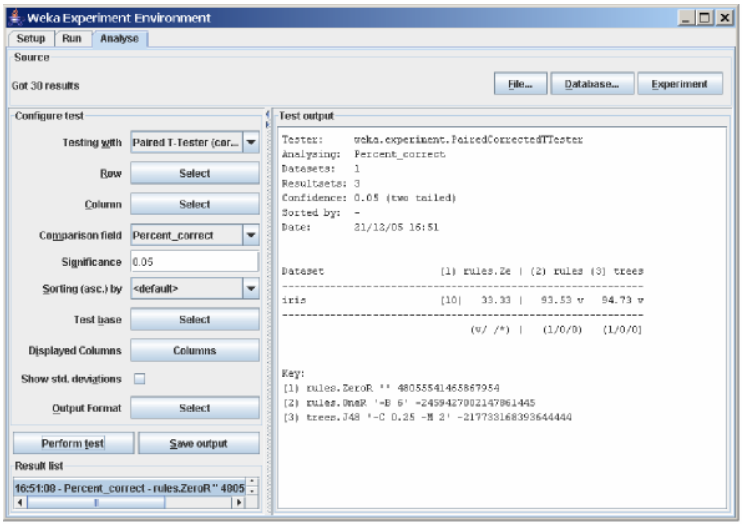


和其他的结果生成器一样，可给定多个的算法。当使用 AveragingResultProducer 时，分类器属性位于 Generator 属性设置的较深层次中。





在这个试验中，ZeroR，OneR，和 J48算法将运行 10 次，进行10折交叉验证。 每组10折交叉验证的结果被取平均值，来产生一个结果行(而不是之前使用 Cross-ValidationResultProducer 的例子中的每一折产生一个结果行)，总共有 30 个结果行。如果保存了原始输出，所有的 300 个结果被发送到文档中。



3 远程试验

远程试验使得你能够将计算负载分布于多台计算机上。

3.1 准备

为运行一个远程试验，你需要：

- 一个数据库服务器
- 若干可运行远程试验引擎（remote engines）的计算机
- 编辑 Weka 发行包中的远程引擎策略（remote engine policy）文件允许从你的主目录装载类文件
- 在某台机器上运行 Experimenter (任何哪台都可以)

在下面的例子中，我们假设名为 johndoe 的用户使用了这个设置：

- 能够访问一组运行某一版本Unix的机器(对于 Windows，需要改变路径名)
- 主目录位于 /home/johndoe
- 这个远程试验使用 HSQLDB [4] 数据库来描述，这需要编辑 DatabaseUtils.props 文件。
- HSQLDB需要设置参数，把 DatabaseUtils.props.hsql 的内容放到 DatabaseUtils.props即可。
- HSQLDB 所需的 jar 文件位于：/home/johndoe/hsqldb. jar

3.2 数据库服务器设置

要设置数据库服务器，先选定或创建一个目录来运行数据库服务器，然后使用以下命令启动服务器：

```
java -classpath /home/johndoe/hsqldb.jar org.hsqldb.Server -database.0 -dbname xdb
```

3.3 远程引擎安装

- 首先，为脚本和策略文件设置一个目录：/home/j ohndoe/remote_engine
- 接着，复制 remoteEngine.jar (Weka 发行包中的；或从源码编译，使用命令 ant remotejar) 到 /home/johndoe/remote_engine 目录。
- 创建一个脚本，名为 /home/johndoe/remote engine/startRemoteEngine，内容如下(如果你使用某种类型的Unix话，不要忘记使用chmod让它可执行)：

```
/path/to/your/jdk/bin/java -Xmx256m\
-classpath /home/johndoe/hsqldb.jar:remoteEngine.jar \
-Djava.security.policy=remote_engine.policyweka.experiment.RemoteEngine &
```

- 现在我们将启动远程引擎(注意必须让 Experimenter 和远程引擎使用相同版本的Java)：
  - 编辑/home/johndoe/remote_engine 中的远程 engine.policy 文件，添加以下行：

```
permissionjava.io.FilePermission
"/home/johndoe/-","read";
```

注意：如果数据集文件不在用户的主目录下，那么在策略文件中就得给出另一个有关文件权

限的条目。

- 对于每台你想在其上运行引擎的机器：
  - Ssh登录到这台机器
  - cd 到 /home/johndoe/remote_engine
  - 运行/home/johndoe/startRemoteEngine (要使远程引擎能使用更多内存，可修改 startRemoteEngine 脚本中的 -Xmx 选项)

3.4 配置 Experimenter

现在我们将运行 Experimenter：

- 为试验创建一个目录
- 复制 DatabaseUtils.props.hsql 文件到这个目录，将之重命名为 DatabaseUtils.props —— Weka 发行包的 weak/experiment 中有它的一个副本。
- 编辑这个文件，修改"jdbcURL=jdbc :hsqldb :hsql ://servername" 这一项，使之以包含运行你的数据库服务器的机器的名字(例如，jdbcURL=jdbc :hsqldb :hsql ://dodo.company.com)
- 现在启动 experimenter (在这个目录中)：

```
Java –classpath /home/johndoe/hsqldb.jar \
-Djava.rmi.server.codebase=file:/home/<path to your weka classes>/ \
weka.gui.experiment.Experimenter
```

现在我们将配置试验：

- 首先在 Setup 标签页中选择 Advanced 模式。
- 现在在 Destination 面板选择 DatabaseResultListener。配置这个结果生成器，并提供sa值作为用户名。
- 从 Result generator 面板，选择 CrossValidationResultProducer 或 RandomSplitResultProducer (这是最常使用的)，然后配置剩下的试验详细信息(例如数据集和分类器)
- 现在勾选激活 Distribute Experiment 面板。
- 点击 Hosts 按钮，然后输入你想启动远程引擎的机器的名字。
- 你可选择by run（按运行分布）或者by dataset（按数据集分布） (试着取某种平衡)
- 保存你的试验配置。
- 现在像通常一样启动你的试验。
- 在 Analyse 标签页中，点击 Database 或 Experiment 按钮，检查你的结果。

3.5 疑难问题解答

- 不要忘记 TRAILING SLASH – 当你为 Experimenter 提供 Java 启动选项时：

```
-Djava.rmi.server.codebase=file:/home/<path to your weka classes>/
```

没有它，你将会碰到安全访问异常。

- 如果你在启动试验时碰到一个错误，像下面这样 ：

```
01:13:19: RemoteExperiment (//blabla.company.com/RemoteEngine)
(sub)experiment (datataset vineyard.arff) failed : java.sql.SQLException:
Table already exists: EXPERIMENT_INDEX in statement [CREATE TABLE
Experiment_index ( Experiment_type LONGVARCHAR, Experiment_setup
LONGVARCHAR, Result_table INT )]

01:13:19: dataset :vineyard.arff RemoteExperiment
(//blabla.company.com/RemoteEngine) (sub)experiment (datataset
vineyard.arff) failed : java.sql.SQLException: Table already
exists: EXPERIMENT_INDEX in statement [CREATE TABLE Experiment_index
( Experiment_type LONGVARCHAR, Experiment_setup LONGVARCHAR, Result_table
INT )]. Scheduling for execution on another host.
```

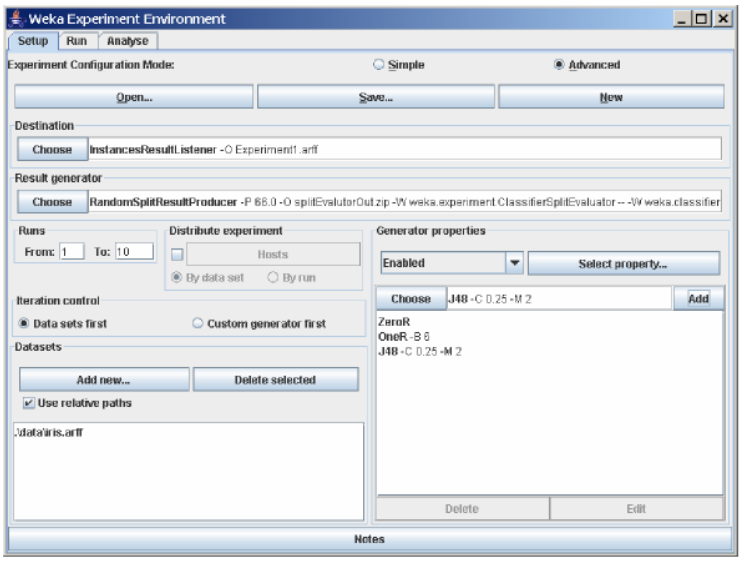
不要紧张。出现这个是因为多台远程机器创建了相同的表，临时死锁了——这个问题将会自动解决，因此不用管它。实际上，它是试验正在运行的一个标志！

- 如果你在序列化（serialize）一个试验之后修改你的 DatabaseUtils.props 文件 (例如一个丢失类型映射（type-mapping）)，将导致一个错误，Experimenter 将使用你序列化试验时所拥有的 DatabaseUtils.props。记住序列化过程同时也序列化了DatabaseUtils 类，并存储了配置文件！这是为什么要把试验保存为 XML，而不是 Java 序列化产生的二进制格式文件的另一个原因。
- 使用一个损坏的或不完整的 DatabaseUtils.props 文件可导致某些罕见的接口错误，例如不能使用数据库 URL 旁边的"User" 按钮。如果怀疑是这个原因，从 CVS 复制一个完好的 DatabaseUtils.props。
- 如果你的远程引擎在 java.util.Hashtable.get() 处碰到NullPointerException，不要害怕。这对你的试验结果没有任何影响。

4 分析结果

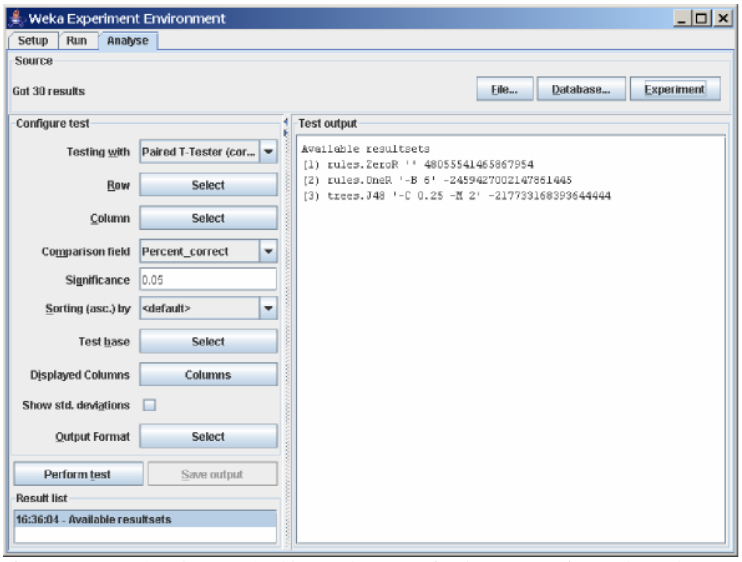
4.1 设置

Weka 包含一个试验结果分析器，用来分析发送InstancesResultListener 的 试验结果。以下显示的试验使用了 3 个算法，ZeroR，OneR 和 J48，运行10次训练和测试来为试验中的 Iris 数据作分类，66% 的数据用于训练，34% 用于测试。



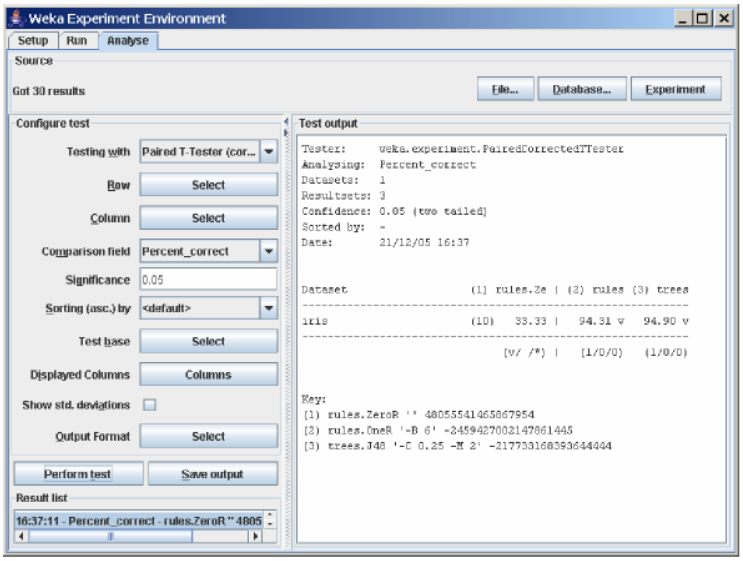
在完成试验设置后，运行该试验。然后选择 Experiment 环境窗口顶部的Analyse 标签页来分析结果。

点击 Experiment 以分析当前试验结果。



在 Source 面板中显示可用的结果行的数量(有 30 个结果)。这个试验包含运行了10次，用了 3 个算法，1 个数据集，总共 30 个结果行。先前的试验文件可以通过点击 File，选择适当的 .arff 结果文件来载入。同样，可从数据库载入发送到数据库的结果(使用 DatabaseResultListener)。

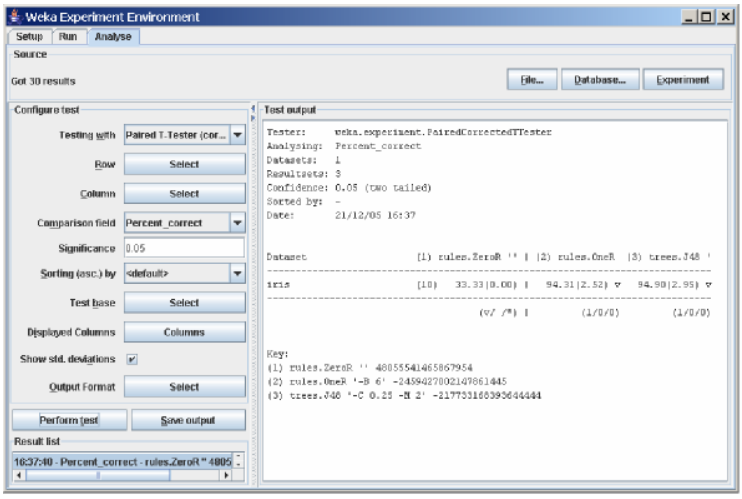
从 Comparison field （比较范围）选择 Percent_correct 属性，点击 Perform test，对3个算法进行比较。



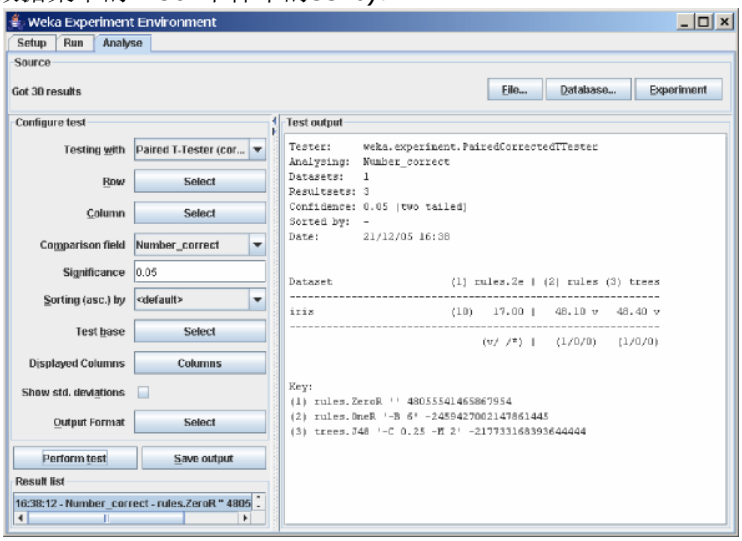
试验中使用的算法显示在列，使用的数据集显示在行。

对于各个算法，用百分比表示的正确率显示在各数据集所在的行：ZeroR 是33.33%， OneR 是94.31% ，J48 是94.90%。符号 v 或 * 表示一个特定结果在指定的显著水平（significance level，目前是0.05）下优于(v) 或不如(*)基准算法 (这里是ZeroR)。OneR 和 J48 的结果统计上优于 ZeroR 所建立的基准。除了第一列，每列的底部有一个总计(xx/ yy/ zz)，它表示在试验所用的数据机上，算法比的基准算法优越 (xx)，等同 (yy)，不如(zz) 的次数。在这个例子中，仅有一个数据集，OneR 比 ZeroR 好一次，从来没有等同或差于 ZeroR (1/0/0)， J48 也比该数据集上的 ZeroR 好。

待估属性的标准差可通过选择 Show std deviations 复选框，并再次点击 Perform test生成。iris 行开始处的值(10) 表示需要计算标准差的待估属性的个数(此时就是运行的次数)。



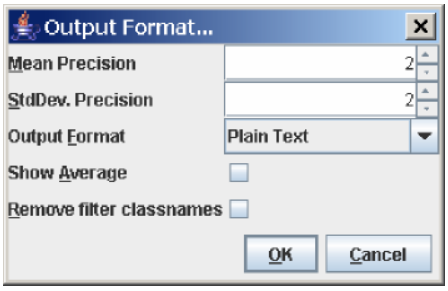
选择 Number correct 作为比较范围，点击 Perform test 得到正确分类个数的平均值。 (来自 50 个 测试样本 - Iris 数据集中的 150 个样本的33%)。



点击输出格式的按钮，会出现一个对话框，让你选择均值和标准差的精度，以及输出的格式。

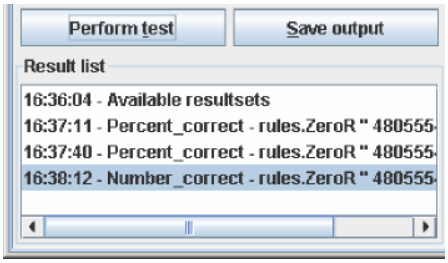
支持以下格式：

- CSV
- GNUPlot
- HTML
- LaTeX
- Plain text （文本文件） (默认)
- Significance only （仅显著水平）



4.2 保存结果

显示在 Test output 面板中的信息由 Result list 面板中当前选中的条目控制。点击一个条目，会显示相应于那个条目的结果。

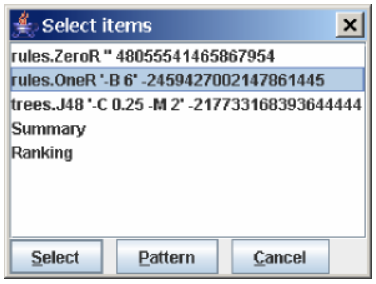


通过点击 Save output，显示在 Test output 面板中的结果可保存进一个文件，一次仅能保存一组结果。但 Weka 允许用户将所有结果保存进相同的数据集：挨个保存，并在第二次和随后的保存中使用 Append 选项，而不是 Overwrite 选项。

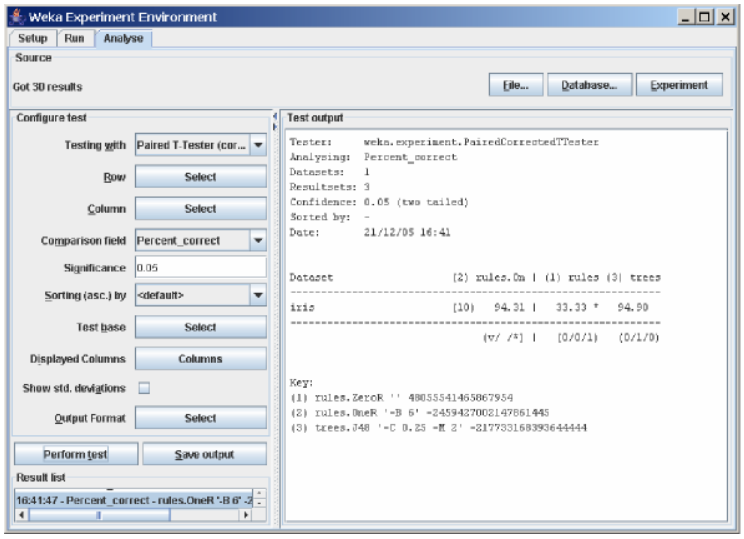


4.3 改变基准算法

改变基准算法可通过点击 Select base... ，然后选择想要的算法。选择 OneR 算法，会导致其它算法和 OneR 算法进行比较。



如果以 OneR 作为基准算法，在 Percent_correct上进行了测试，对于 OneR 和J48的结果，系统显示其没有统计意义下的区别。然而，在 OneR 和 ZeroR间有统计意义下有显著的区别。



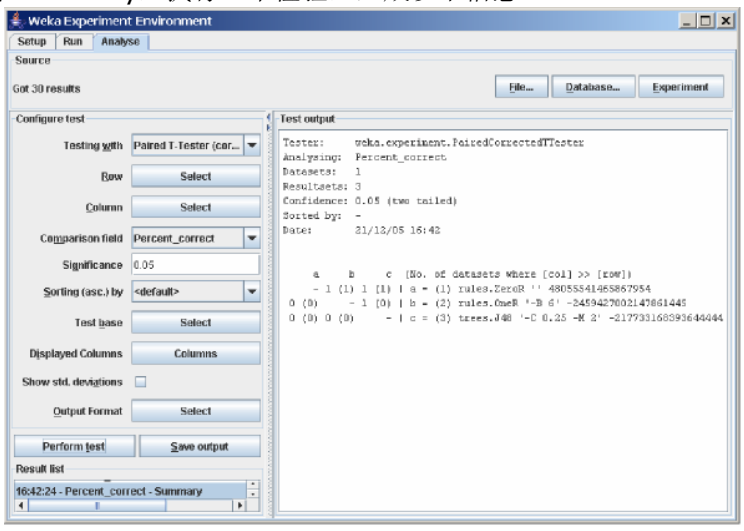
4.4 统计显著性

上一节中使用的术语“统计意义下显著”指的是使用标准 T 检验 或 效正过的重新抽样 T 检验 [2] 对算法结果进行成对比较的结果。后者是默认的，这是因为误差估计之间存在相依性，标准 T- Test 可能会报告太多的显著区别， (尤其是当运行一次 x 折交叉验证时)。T 检验的更多信息，请参考 Weka book [1] 或一个介绍性的统计课本。随着显著水平降低，结论的置信度将提高。

在当前试验中，OneR 和 J48 算法在统计意义下没有显著区别。

4.5 Summary Test （描述性检验）

从 Test base 中选择 Summary，执行一个检验，生成以下信息。

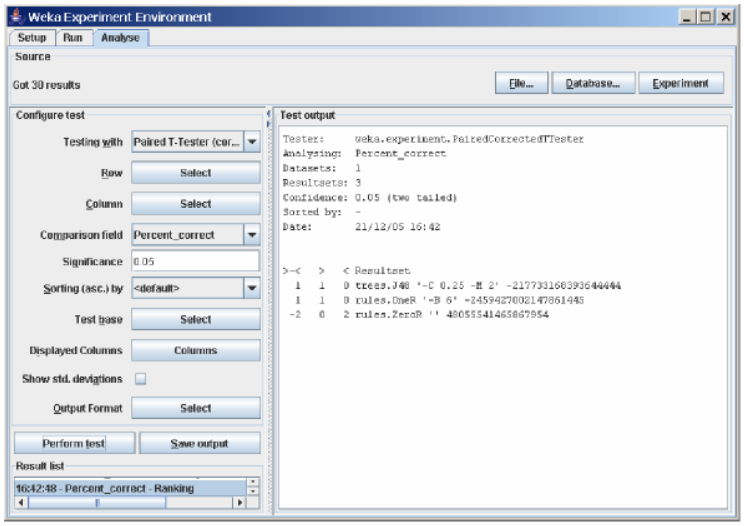


在这个试验中，第一行(- 1 1) 表示列 b (OneR) 优于行 a (ZeroR)，列 c (J48) 也优于行 a。 括号中的数字表示对应列显著的优于对应行的次数。一个 0 表示对应列的算法从没有（显著的）优于对应行的算法。

4.6 Ranking Test[2] （排序检验）

从 Test base 中选择 Ranking，会生成以下信息。

[2]这里不是统计中通常所说的秩检验—译注。



排序检验根据某算法显著的优于(>)和劣于(<)其它算法的总次数来给算法排序。第一列 (>– <) 是优的次数与劣的次数之差。这个差就是排序的依据。

5 参考文献

1) Witten, I.H. and Frank, E. (2005) Data Mining: Practical machine learning tools and techniques. 2nd edition Morgan Kaufmann, San Francisco.
2) Bengio, Y. and Nadeau, C. (1999) Inference for the Generalization Error.
3) Ross Quinlan (1993). C4. 5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA.
4) HSQLDB – http://hsqldb.sourceforge.net/
5) WekaDoc – http://weka.sourceforge.net/wekadoc/