Open-Source Machine Learning: R Meets Weka

Kurt Hornik, Christian Buchta, Michael Schauerhuber, David Meyer, Achim Zeileis

http://statmath.wu-wien.ac.at/~zeileis/

Weka?

Weka is not only a flightless endemic bird of New Zealand (Gallirallus australis, picture from Wekapedia)

but also the

Waikato Environment for Knowledge Analysis.

Weka

- Comprehensive collection of machine-learning algorithms for data mining tasks
- Provides tools for data preprocessing, regression, classification, clustering, association rules, and visualization
- Implemented in Java (initially: C/TclTk) and released under GPL
- 3 GUIs (“Explorer”, “Experimenter”, “KnowledgeFlow”) and a standardized CLI
- Started in 1992, funded by the NZ government since 1993
- Recently “acquired” by Pentaho, the world’s most popular open-source business intelligence platform
- Several other projects are based on Weka

Why bother?

- Complements the book “Data Mining” by Ian Witten and Eibe Frank (heavily used in CS curricula)
- Implements a variety of methods popular in machine learning and useful, but typically not available for statistical learning (e.g., rule, meta, and lazy learners, CobWeb, DBSCAN, Tertius, . . .)
- Provides de-facto reference implementations, including the key decision tree algorithms C4.5 (called J4.8) and M5 (called M5’)

Interfacing strategy

- Weka provides a consistent "functional" methods interface for its learner classes: e.g., all supervised learners (called "classifiers" in Wekaspeak) have `buildClassifier()` and `classifyInstances()` methods.
- Assure generalizability and maintainability by re-using class structure.
- Provide R interface with "the usual look and feel", e.g., a fitting function with formula interface and `print()`, `predict()`, `summary()` methods.
- Our R package **RWeka** accomplishes this by providing *interface generators* for classifiers, clusterers, associators and filters.
- It uses package **rJava** for low-level direct R/Java interfacing.
- Many Weka algorithms/classes are readily in **RWeka**, users can easily register more interfaces on the fly.

```
R> library("RWeka")
R> list_Weka_interfaces()
$Associators
[1] "Apriori" "Tertius"

$Classifiers
 [1] "AdaBoostM1"       "Bagging"          "DecisionStump"    "IBk"
 [5] "J48"              "JRip"             "LBR"              "LMT"
 [9] "LinearRegression" "Logistic"         "LogitBoost"       "M5P"
[13] "M5Rules"          "MultiBoostAB"     "OneR"             "PART"
[17] "SMO"              "Stacking"

$Clusterers
[1] "Cobweb"          "DBScan"          "FarthestFirst" "SimpleKMeans"
[5] "XMeans"

$Filters
[1] "Discretize" "Normalize"
```

```
R> foo <- make_Weka_classifier("weka/classifiers/trees/J48", c("bar",
+     "Weka_tree"))
R> foo
An R interface to Weka class 'weka.classifiers.trees.J48',
which has information

  Class for generating a pruned or unpruned C4.5 decision tree. For
  more information, see

  Ross Quinlan (1993). C4.5: Programs for Machine Learning. Morgan
  Kaufmann Publishers, San Mateo, CA.

Argument list:
  (formula, data, subset, na.action, control = Weka_control())

Returns objects inheriting from classes:
  bar Weka_tree Weka_classifier
```

How does this work?

- `make_Weka_classifier()` creates an interface function `foo()` to the given Weka (classifier) class (JNI notation or Java class)
- `foo()` "knows" to be such an interface, and to return objects inheriting from the given "`bar`" and the general "`Weka_tree`" and "`Weka_classifier`" classes
- All such classifier interfaces have formals

```
formula  data  subset  na.action  control
```

- Printing such interface functions uses Weka's `globalInfo()` method to provide a description of the algorithm being interfaced

- When the interface function is called, a model frame is set up in R which is transferred to a Weka instance object
- The `buildClassifier()` method of the Weka class interfaces is called with these instances
- The model predictions for the training data are obtained by calling the Weka `classifyInstances()` methods for the built classifier and each training instance
- A suitably classed object containing both a reference to the built classifier and the predictions is returned
- Such objects have at least a `print()` method (using Weka's `toString()`) and a `predict()` method for either "classes" (numeric for regression, factor for classification) or class probabilities (using Weka's `distributionForInstance()`)

```
R> m1 <- foo(Species ~ ., data = iris)
R> m1
J48 pruned tree
------------------

Petal.Width <= 0.6: setosa (50.0)
Petal.Width > 0.6
|   Petal.Width <= 1.7
|   |   Petal.Length <= 4.9: versicolor (48.0/1.0)
|   |   Petal.Length > 4.9
|   |   |   Petal.Width <= 1.5: virginica (3.0)
|   |   |   Petal.Width > 1.5: versicolor (3.0/1.0)
|   Petal.Width > 1.7: virginica (46.0/1.0)

Number of Leaves  :           5

Size of the tree :           9
```

Confusion matrix:

```
R> table(true = iris$Species, predicted = predict(m1))
            predicted
true          setosa versicolor virginica
  setosa          50          0         0
  versicolor       0         49         1
  virginica        0          2        48
```

Control arguments

Building the classifiers is controlled by Weka options.

These can be queried using `WOW()`, the Weka Option Wizard (works by calling the Weka `listOptions()` method for Weka class interfaced and doing some magic).

These can be set by using `Weka_control()` to set up the `control` argument of the interface function.

These “control lists” essentially produce Weka’s command-line option style (`-R -M 5`) from R’s typical tag-value style (`(R = TRUE, M = 5)`).

Query J4.8 options using the Weka Option Wizard:

```
R> WOW(foo)
-U      Use unpruned tree.
-C      Set confidence threshold for pruning.  (default 0.25)
        Number of arguments: 1.
-M      Set minimum number of instances per leaf.  (default 2)
        Number of arguments: 1.
-R      Use reduced error pruning.
-N      Set number of folds for reduced error pruning. One fold is used
        as pruning set.  (default 3)
        Number of arguments: 1.
-B      Use binary splits only.
-S      Don't perform subtree raising.
-L      Do not clean up after the tree has been built.
-A      Laplace smoothing for predicted probabilities.
-Q      Seed for random data shuffling (default 1).
        Number of arguments: 1.
```

Now build a J4.8 tree with reduced error pruning `R` and the minimal number `M` of instances set to 5:

```
R> m2 <- foo(Species ~ ., data = iris, control = Weka_control(R = TRUE, M = 5))
R> m2
J48 pruned tree
------------------

Petal.Width <= 0.6: setosa (34.0)
Petal.Width > 0.6
|   Petal.Width <= 1.5: versicolor (32.0/1.0)
|   Petal.Width > 1.5: virginica (34.0/2.0)

Number of Leaves  :         3

Size of the tree :          5
```

Performance measures

Function `evaluate_Weka_classifier()` employs Weka's powerful "`Evaluation`" class for computing model performance statistics for fitted classifiers (by default on the training data).

The default is currently used for `summary()` methods.

```
R> evaluate_Weka_classifier(m1)
=== Summary ===

Correctly Classified Instances         147               98      %
Incorrectly Classified Instances         3                2      %
Kappa statistic                          0.97
Mean absolute error                      0.0233
Root mean squared error                  0.108
Relative absolute error                  5.2482 %
Root relative squared error             22.9089 %
Total Number of Instances              150

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = setosa
  0 49  1 |  b = versicolor
  0  2 48 |  c = virginica
```

Plotting

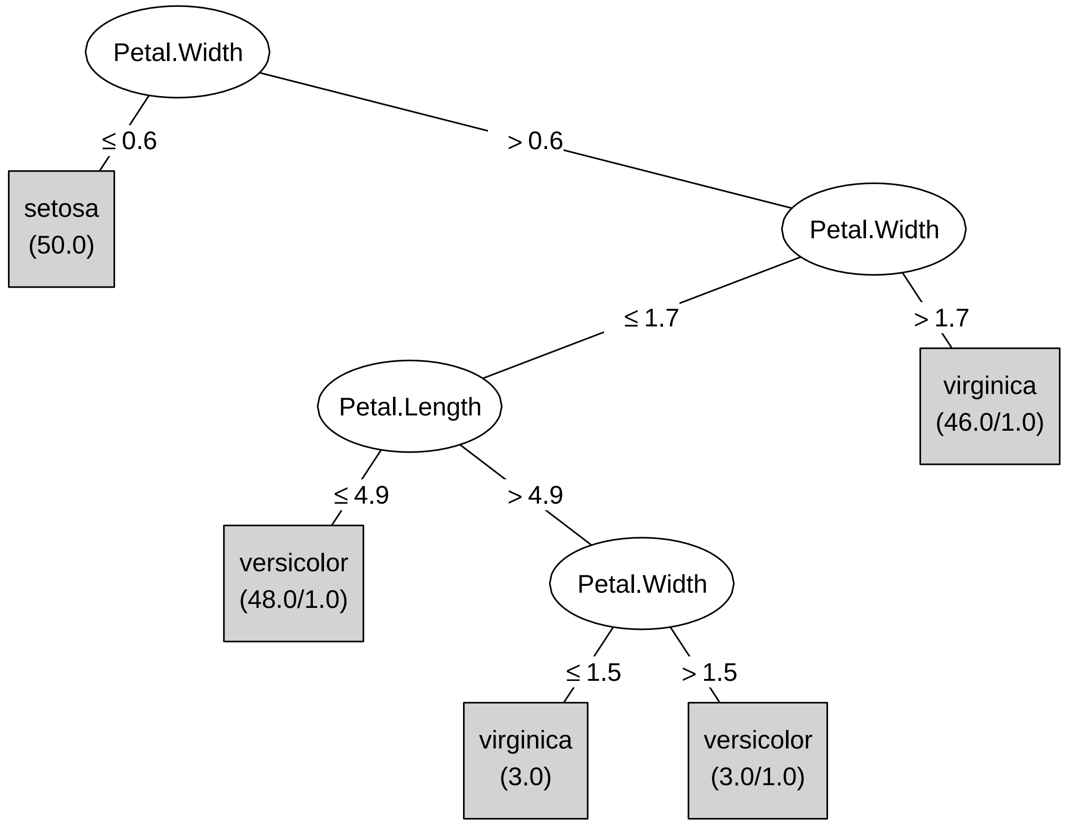
For the Weka tree learners (`J48()`, `M5P()`, `LMT()`) registerd by default, `plot()` methods are based on the routines for "`BinaryTree`" objects in package **party**.

For Weka classifiers with a "Drawable" interface, i.e., providing a `graph()` method, one can create DOT language representations of the built classifiers for processing via **GraphViz**. This can also be used for visualization via **Rgraphviz**.

Could also use Weka's native plotting facilities. Currently only experimental, as not integrated into R's display lists (and hence possibly confusing . . .)

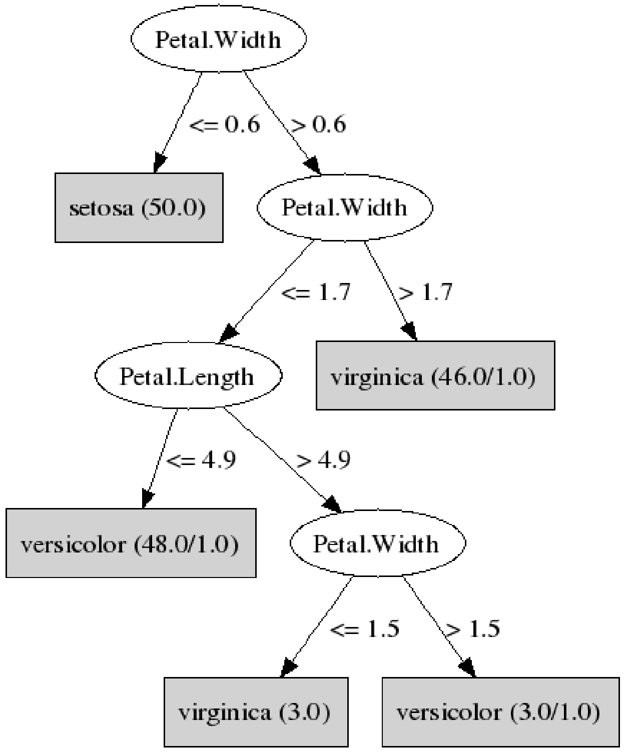
Plotting via **party**:

```
R> plot(m1)
```



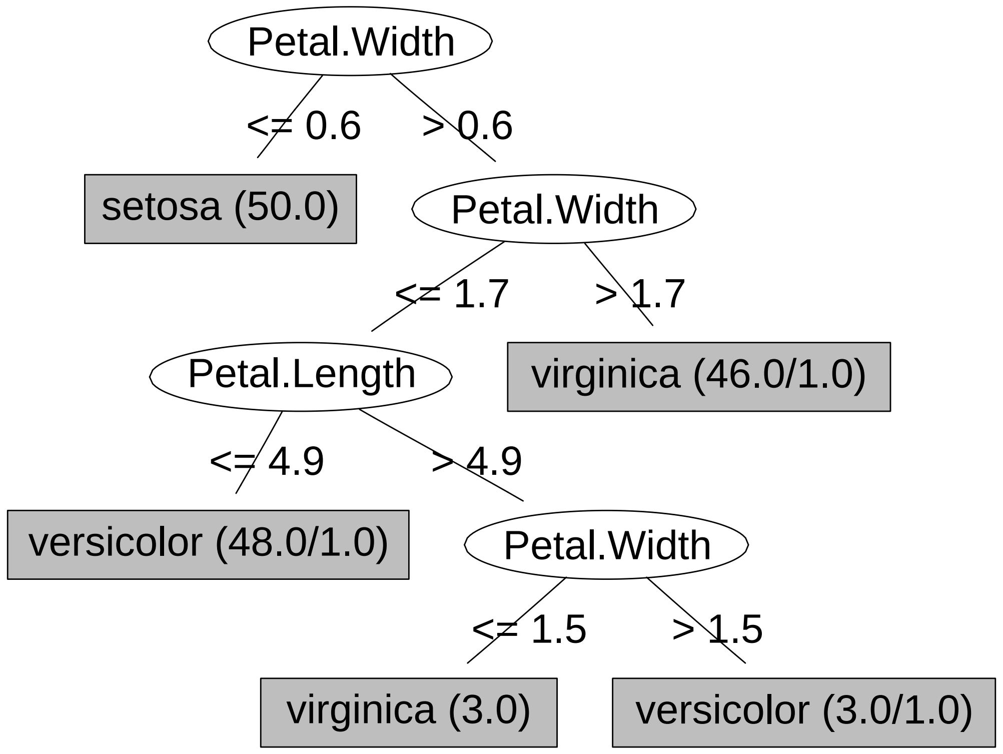
Plotting via **GraphViz**:

```
R> write_to_dot(m1, "m1.dot")
R> system("dot -Tpng m1.dot > m1.png")
```



Plotting via **Rgraphviz**:

```
R> library("Rgraphviz")
R> plot(agread("m1.dot"))
```



Open issues

- Too much privacy in Weka. Some of Weka's internal fitted model structures (linear model terms, tree splits, etc.) are currently not easily accessible. But cooperation with the Weka project team is possible and necessary.
- Too much data manipulation. For applying Weka's algorithms the data is always transferred back and forth. This could be avoided by using smarter proxy objecs which is difficult with R's current semantics.
- Weka to R callbacks. To be able to use R algorithms within Weka (e.g., for meta learners) Weka classes for R algorithms are needed. Can this be done both reasonably generally and efficiently?

Benchmarking study

- tree algorithms from statistics and machine learning communities,
- model trees and constant-fit trees,
- standard benchmarking data sets: 12 regression, 19 classification.
- fit algorithms on 500 bootstrap samples from each data set, evaluate misclassification rate or root mean squared error on out-of-bag sample,
- evaluate also model complexity: number of splits and estimated parameters,
- assess significance of all pairwise differences by multiple testing,
- aggregate all individual results over various data sets by consensus rankings (via optimal linear ordering).

Benchmarking algorithms

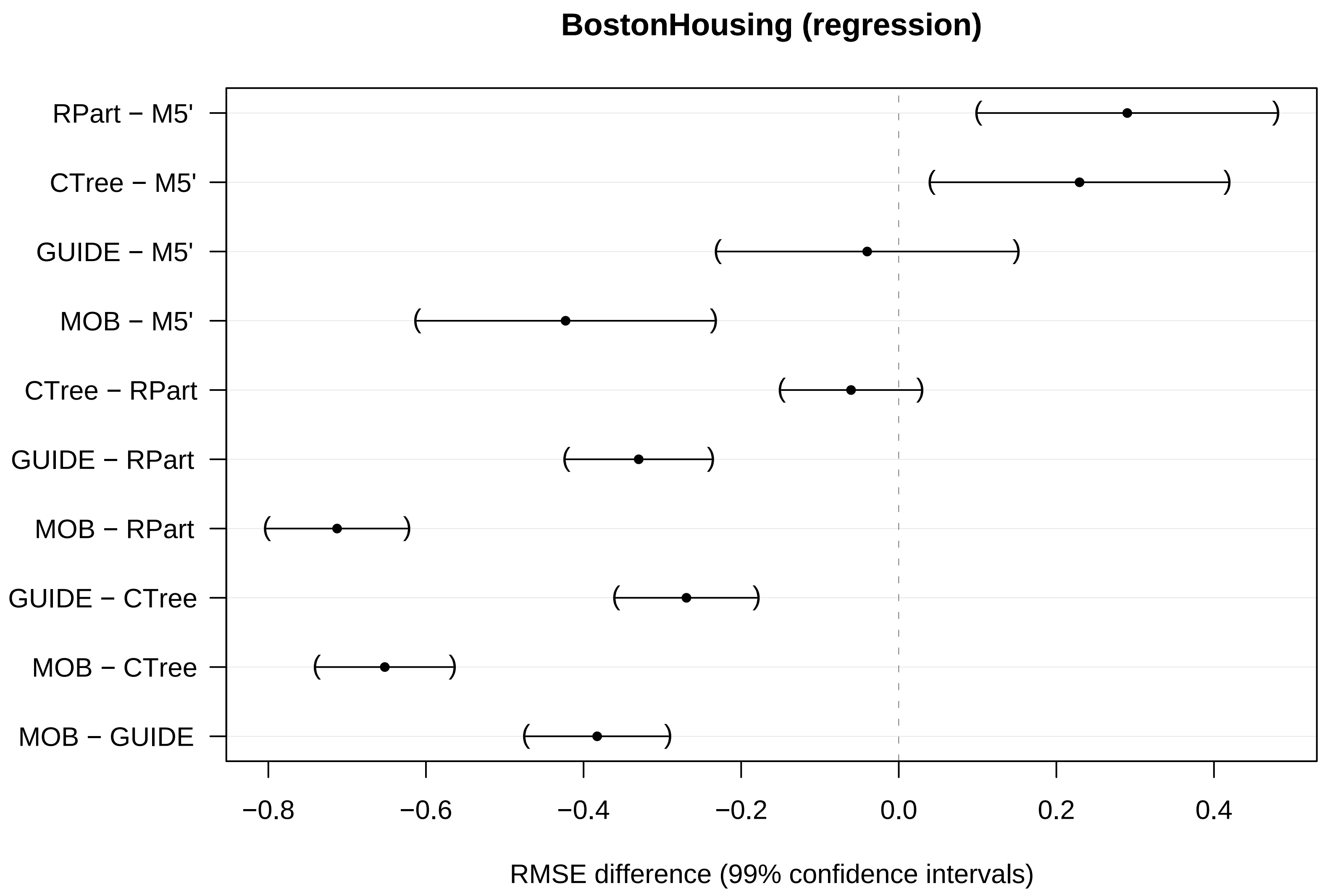
- J4.8: Java implementation of C4.5 revision 8 (**RWeka**),
- M5’: rational reconstruction of M5 (**RWeka**),
- LMT: Logistic Model Trees (**RWeka**),
- RPart: **rpart** implementation of CART (Classification And Regression Trees),
- QUEST: Quick, Unbiased and Efficient Statistical Tree (**LohTools**),
- GUIDE: Generalized, Unbiased, Interaction Detection and Estimation (**LohTools**),
- CRUISE: Classification Rule with Unbiased Interaction Selection and Estimation (**LohTools**),
- CTree: Conditional inference Trees (**party**),
- MOB: MOdel-Based recursive partitioning (here based on linear/logistic regression, **party**),

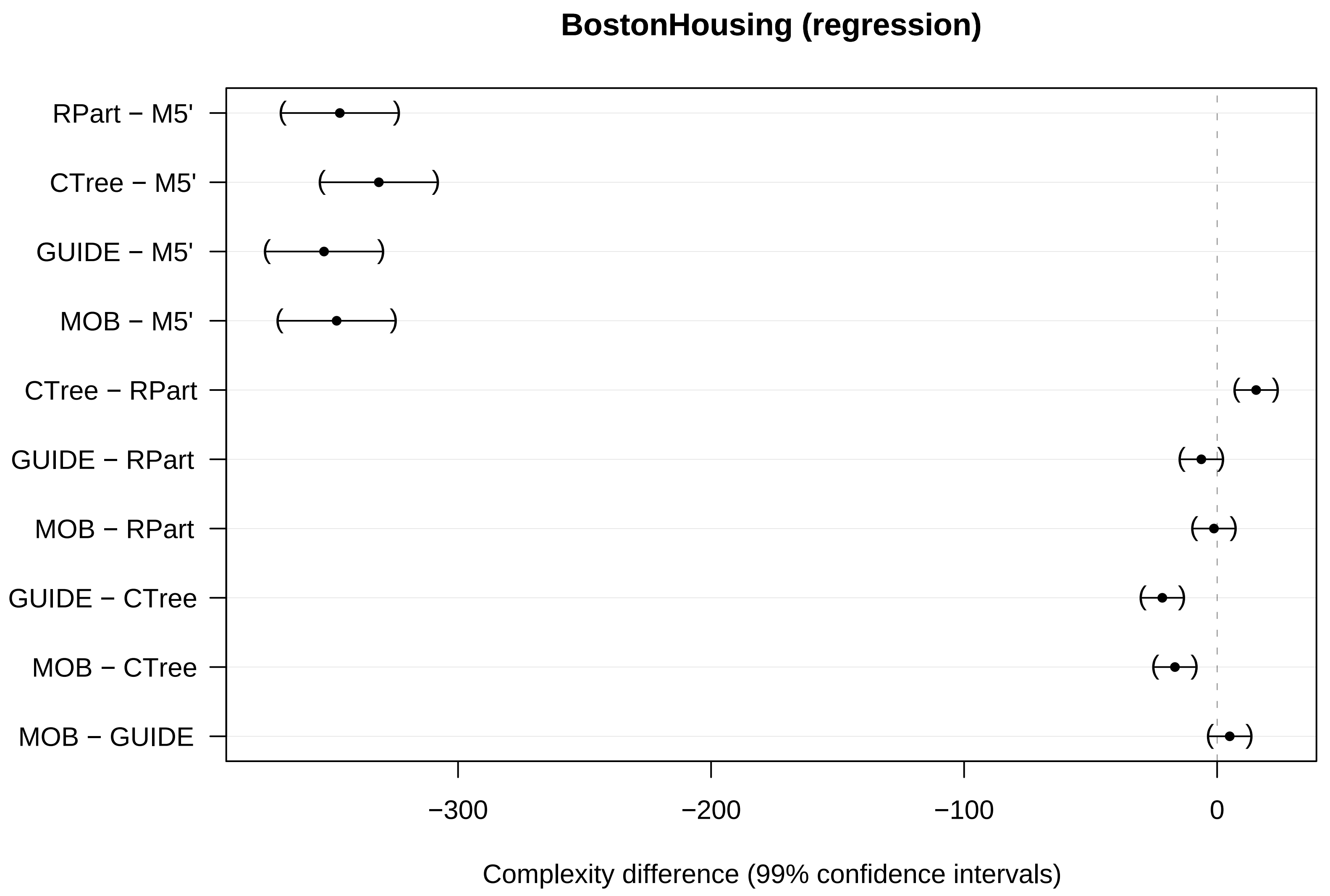
Regression data sets

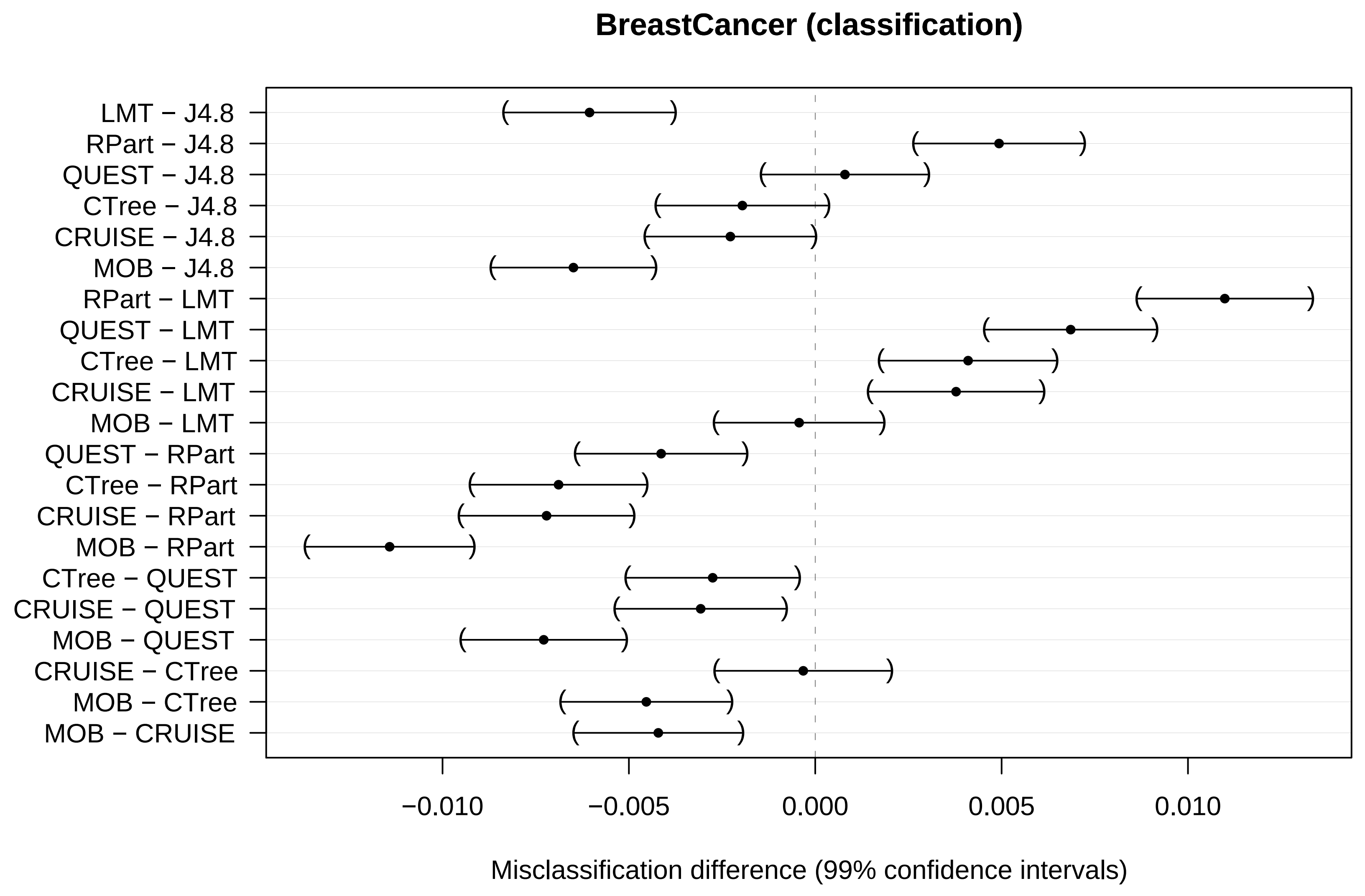
Name	**Size**	**Cat. Var.**	**Num. Var.**
abalone	4177	1	7
autompg	398	3	4
autos	205	10	15
BostonHousing	506	1	12
comp-active	8192	-	12
Friedman1	1000	-	5
Friedman2	1000	-	4
Friedman3	1000	-	4
journals	180	1	6
Ozone	366	3	9
Servo	167	2	3
SLID	7425	2	2

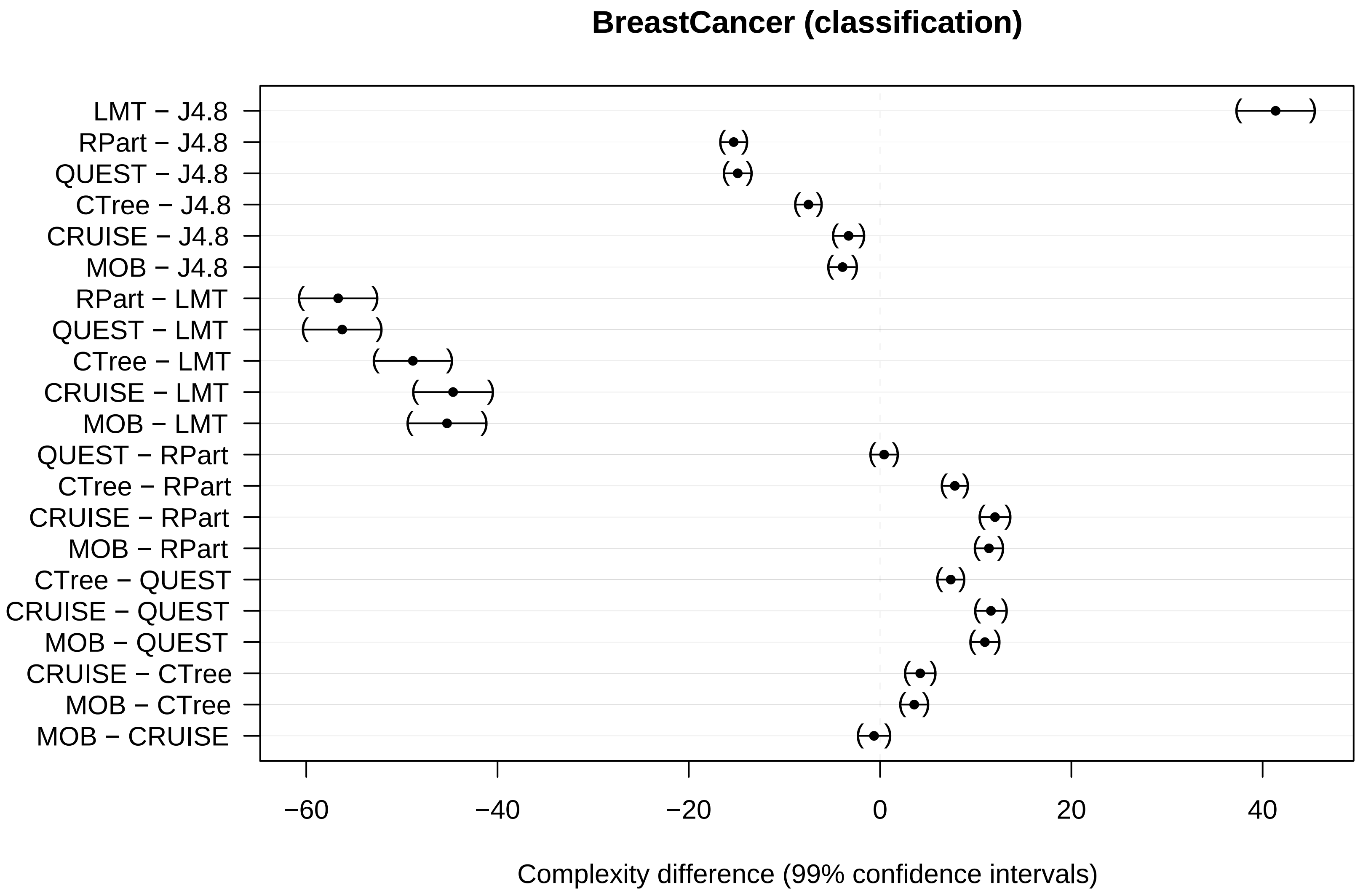
Classification data sets

Name	Size	Cat. Var.	Num. Var.
BreastCancer	699	9	-
chess	3196	36	-
circle	1000	-	2
credit	690	-	24
heart	303	8	5
hepatitis	155	13	6
HouseVotes84	435	16	-
Ionosphere	351	1	32
liver	345	-	6
musk	476	-	166
PimaIndiansDiabetes	768	-	8
promotergene	106	57	-
ringnorm	1000	-	20
sonar	208	-	60
spirals	1000	-	2
threenorm	1000	-	20
tictactoe	958	9	-
titanic	2201	3	-
twonorm	1000	-	20


BostonHousing (regression)
RPart − M5'
CTree − M5'
GUIDE − M5'
MOB − M5'
CTree − RPart
GUIDE − RPart
MOB − RPart
GUIDE − CTree
MOB − CTree
MOB − GUIDE
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
RMSE difference (99% confidence intervals)


BostonHousing (regression)
RPart − M5'
CTree − M5'
GUIDE − M5'
MOB − M5'
CTree − RPart
GUIDE − RPart
MOB − RPart
GUIDE − CTree
MOB − CTree
MOB − GUIDE
−300
−200
−100
0
Complexity difference (99% confidence intervals)


BreastCancer (classification)
LMT − J4.8
RPart − J4.8
QUEST − J4.8
CTree − J4.8
CRUISE − J4.8
MOB − J4.8
RPart − LMT
QUEST − LMT
CTree − LMT
CRUISE − LMT
MOB − LMT
QUEST − RPart
CTree − RPart
CRUISE − RPart
MOB − RPart
CTree − QUEST
CRUISE − QUEST
MOB − QUEST
CRUISE − CTree
MOB − CTree
MOB − CRUISE
−0.010
−0.005
0.000
0.005
0.010
Misclassification difference (99% confidence intervals)


BreastCancer (classification)
LMT − J4.8
RPart − J4.8
QUEST − J4.8
CTree − J4.8
CRUISE − J4.8
MOB − J4.8
RPart − LMT
QUEST − LMT
CTree − LMT
CRUISE − LMT
MOB − LMT
QUEST − RPart
CTree − RPart
CRUISE − RPart
MOB − RPart
CTree − QUEST
CRUISE − QUEST
MOB − QUEST
CRUISE − CTree
MOB − CTree
MOB − CRUISE
−60
−40
−20
0
20
40
Complexity difference (99% confidence intervals)

Consensus rankings for regression data sets:

	RMSE	**Complexity**
1	GUIDE	GUIDE
2	M5'	CTree
3	CTree	RPart
4	RPart	M5'

Consensus rankings for classification data sets:

	Misclassification	**Complexity**
1	LMT	RPart
2	J4.8	QUEST
3	CRUISE	CTree
4	RPart	J4.8
5	CTree	CRUISE
6	QUEST	LMT

References

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and Regression Trees*. Wadsworth: California.

Hornik K, Zeileis A, Hothorn T, Buchta C (2007). "**RWeka**: An R Interface to Weka." R package version 0.2-14. URL `http://CRAN.R-project.org/`.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Hothorn T, Zeileis A, Hornik K (2007). "**party**: A Laboratory for Recursive Part(y)itioning." R package version 0.9-9. URL `http://CRAN.R-project.org/`.

Landwehr N, Hall M, Frank E (2005). "Logistic Model Trees." *Machine Learning*, **59**, 161–205.

Loh W-Y (2002). "Regression Trees With Unbiased Variable Selection and Interaction Detection." *Statistica Sinica*, **12**, 361–386.

Quinlan JR (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Therneau TM, Atkinson EJ (1997) "An Introduction to Recursive Partitioning Using the **rpart** Routine." *Technical Report 61*, Section of Biostatistics, Mayo Clinic, Rochester. URL `http://www.mayo.edu/hsr/techrpt/61.pdf`.

Witten IH, Frank E (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Zeileis A, Hothorn T, Hornik K (2005). "Model-based Recursive Partitioning." *Report 19*, Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Research Report Series. URL `http://epub.wu-wien.ac.at/`.

Zeileis A, Hothorn T (2007). "**LohTools**: R Interface to GUIDE, QUEST, CRUISE." R package version 0.0-1.