Tutorial: OpenML with R and mlr

Bernd Bischl, Giuseppe Casalicchio et al.

OpenML 2016

Section 1

The mlr Package

Motivation

The good news

- CRAN serves hundreds of packages for machine learning
- Many packages are compliant to the unwritten interface definition:

```
>   model = fit(target ~ ., data = train.data, ...)
>   predictions = predict(model, newdata = test.data, ...)
```

The bad news

- Some packages do not support the formula interface or their API is "just different"
- No meta-information available or buried in docs (sometimes not documented at all)
- Larger experiments lead to lengthy, tedious and error-prone code

Motivation: mlr

https://github.com/mlr-org/mlr

- Unified interface for the basic building blocks: tasks, learners, resampling, hyperparameters, ...
- Reflections: nearly all objects are queryable (i.e. you can ask them for their properties and program on them)
- Possibility to fit, predict, evaluate and resample models
- Different visualizations for e.g. ROC curves and predictions
- Benchmarking of learners for muliple data sets
- Parallelization is built-in
- ...

Task Abstractions

- Regression, classification, survival and cost-sensitive tasks
- Internally: data frame with annotations: target column(s), weights, misclassification costs, ...)

```
> data("Sonar", package = "mlbench")
> task = makeClassifTask(data = Sonar, target = "Class")
> print(task)

## Supervised task: Sonar
## Type: classif
## Target: Class
## Observations: 208
## Features:
## numerics  factors  ordered
##       60        0        0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
## Classes: 2
##   M   R
## 111  97
## Positive class: M
```

Learner Abstractions

- 72 classification, 53 regression, 11 survival, 8 cluster

```
> lrn = makeLearner("classif.rpart")
> print(lrn)


## Learner classif.rpart from package rpart
## Type: classif
## Name: Decision Tree; Short name: rpart
## Class: classif.rpart
## Properties: twoclass,multiclass,missings,numerics,factors,ordered,prob,weights
## Predict-Type: response
## Hyperparameters: xval=0


> lrn = makeLearner("classif.rpart", minsplit = 20, predict.type = "prob")
> getParamSet(lrn)


##                   Type len  Def   Constr Req Tunable Trafo
## minsplit       integer   -   20 1 to Inf   -    TRUE     -
## minbucket      integer   -    - 1 to Inf   -    TRUE     -
## cp             numeric   - 0.01   0 to 1   -    TRUE     -
## maxcompete     integer   -    4 0 to Inf   -    TRUE     -
## maxsurrogate   integer   -    5 0 to Inf   -    TRUE     -
## usesurrogate   discrete   -    2    0,1,2   -    TRUE     -
## surrogatestyle discrete   -    0      0,1   -    TRUE     -
## maxdepth       integer   -   30  1 to 30   -    TRUE     -
## xval           integer   -   10 0 to Inf   -   FALSE     -
## parms          untyped   -    -        -   -    TRUE     -
```

Resampling

- Resampling techniques: CV, Bootstrap, Subsampling, ...

```
> cv3f = makeResampleDesc("CV", iters = 3, stratify = TRUE)
```

- 10-fold CV of rpart on iris

```
> lrn = makeLearner("classif.rpart", predict.type = "prob")
> cv10f = makeResampleDesc("CV", iters = 10)
> measures = list(acc, auc)
>
> resample(lrn, task, cv10f, measures)$aggr

## acc.test.mean auc.test.mean
##     0.7209524     0.7571493
```

- For the lazy:

```
> r = crossval(lrn, task, iters = 3L)
```

Benchmarking

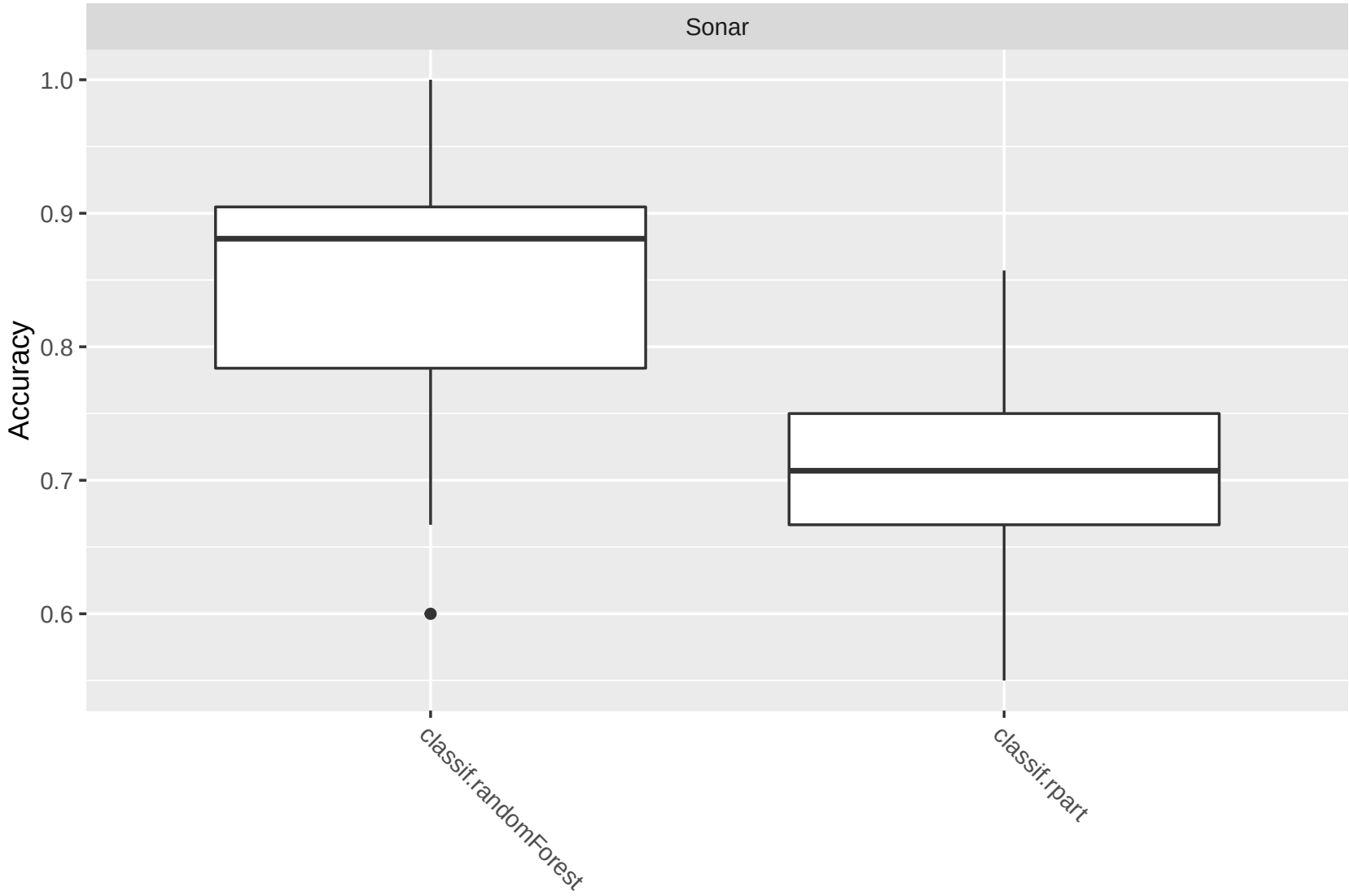
- Compare multiple learners on multiple tasks
- Fair comparisons: same training and test sets for each learner

```
> sonar.task = makeClassifTask(data = Sonar, target = "Class")
> cv10f = makeResampleDesc("CV", iters = 10)
> measures = list(acc, auc)
> learners = list(
+   makeLearner("classif.randomForest", predict.type = "prob"),
+   makeLearner("classif.rpart", predict.type = "prob")
+ )
>
> (res = benchmark(learners, sonar.task, cv10f, measures))

##   task.id           learner.id acc.test.mean auc.test.mean
## 1   Sonar classif.randomForest     0.8402381     0.9344984
## 2   Sonar        classif.rpart     0.7059524     0.7305445
```

Benchmarking

```
> plotBMRBoxplots(res)
```



Further Features

- Easy extension mechanism through S3 inheritance
- Many convenience methods and generic building blocks for your machine learning experiments
- Easy hyperparameter tuning using different optimization strategies, including potent configurators like iterated F-racing (irace) or sequential model-based optimization
- Variable selection with filters and wrappers
- Nested resampling of models with tuning and feature selection
- Cost-sensitive learning, threshold tuning and imbalance correction
- Wrapper mechanism to extend learner functionality and complex and custom ways
- Combine different processing steps to a complex data mining chain that can be jointly optimized
- ...

Section 2

OpenML R-Package

OpenML R-Package

`https://github.com/openml/r`

Current API in R

- Explore data and tasks
- Download data and tasks
- Register learners
- Upload runs
- Explore your own and other people's results

Already nicely connected to `mlr`!

OpenML: Configuration I

```
> library(OpenML)
> getOMLConfig()

## OpenML configuration:
##   server            : http://api_new.openml.org/v1
##   cachedir          : C:\Users\Giuseppe\AppData\Local\Temp\RtmpCmCA7e/cache
##   verbosity         : 1
##   arff.reader       : RWeka
##   apikey            : PLEASE CHANGE ME
```

You can set your apikey using

- `setOMLConfig` for the current R session only
- `saveOMLConfig` to set it globally (it writes a `config` file in your home directory)

OpenML: Configuration II

```
> setOMLConfig(apikey = "**************************31fea")
> saveOMLConfig(apikey = "**************************31fea")
```

```
## OpenML configuration:
##   server             : http://api_new.openml.org/v1
##   cachedir           : C:\Users\Giuseppe\AppData\Local\Temp\RtmpCmCA7e/cache
##   verbosity          : 0
##   arff.reader        : RWeka
##   apikey             : **************************31fea
```

OpenML: Explore and Select Data I

```
> ds = listOMLDataSets()
> str(ds, vec.len = 2)


## 'data.frame': 2417 obs. of  14 variables:
##  $ did                                : int  1 2 3 4 5 ...
##  $ status                             : Factor w/ 1 level "active": 1 1 1 1 1 ...
##  $ name                               : chr  "anneal" "anneal" ...
##  $ MajorityClassSize                  : int  684 684 1669 37 245 ...
##  $ MaxNominalAttDistinctValues        : int  10 9 3 3 2 ...
##  $ MinorityClassSize                  : int  0 0 1527 20 0 ...
##  $ NumBinaryAtts                      : int  14 7 34 3 73 ...
##  $ NumberOfClasses                    : int  6 6 2 2 16 ...
##  $ NumberOfFeatures                   : int  39 39 37 17 280 ...
##  $ NumberOfInstances                  : int  898 898 3196 57 452 ...
##  $ NumberOfInstancesWithMissingValues: int  0 898 0 56 384 ...
##  $ NumberOfMissingValues              : int  0 22175 0 326 408 ...
##  $ NumberOfNumericFeatures            : int  6 6 0 8 206 ...
##  $ NumberOfSymbolicFeatures           : int  32 32 36 8 73 ...
```

OpenML: Explore and Select Data II

```
> tasks = listOMLTasks()
> str(tasks, vec.len = 2)


## 'data.frame': 7386 obs. of  20 variables:
##  $ task.id                          : int  1 2 3 4 5 ...
##  $ task.type                        : chr  "Supervised Classification" "Supervised Classification" ..
##  $ did                              : int  1 2 3 4 5 ...
##  $ status                           : Factor w/ 1 level "active": 1 1 1 1 1 ...
##  $ name                             : chr  "anneal" "anneal" ...
##  $ target.feature                   : chr  "class" "class" ...
##  $ tags                             : chr  "basic, study_1, study_7, under100k, under1m" "basic, study
##  $ estimation.procedure             : Factor w/ 12 levels "10-fold Crossvalidation",..: 1 1 1 1 1 ...
##  $ evaluation.measures              : chr  "predictive_accuracy" "predictive_accuracy" ...
##  $ MajorityClassSize                : int  684 684 1669 37 245 ...
##  $ MaxNominalAttDistinctValues      : int  10 9 3 3 2 ...
##  $ MinorityClassSize                : int  0 0 1527 20 0 ...
##  $ NumBinaryAtts                    : int  14 7 34 3 73 ...
##  $ NumberOfClasses                  : int  6 6 2 2 16 ...
##  $ NumberOfFeatures                 : int  39 39 37 17 280 ...
##  $ NumberOfInstances                : int  898 898 3196 57 452 ...
##  $ NumberOfInstancesWithMissingValues: int  0 898 0 56 384 ...
##  $ NumberOfMissingValues            : int  0 22175 0 326 408 ...
##  $ NumberOfNumericFeatures          : int  6 6 0 8 206 ...
##  $ NumberOfSymbolicFeatures         : int  32 32 36 8 73 ...
```

OpenML: Download a Data Set

```
> # uses built in caching from disk
> d = getOMLDataSet(1)
```

Is a list with of class `OMLDataSet`. Important slots:

- `desc` contains all infos from the `.xml`
- `data` contains the `data.frame` with the data from the `.arff` file.

OpenML: Download a Task I

```
> # uses built in caching from disk
> oml.task = getOMLTask(task.id = 1)
> oml.task

##
## OpenML Task 1 :: (Data ID = 1)
##   Task Type             : Supervised Classification
##   Data Set              : anneal :: (Version = 2, OpenML ID = 1)
##   Target Feature(s)     : class
##   Tags                  : basic, study_1, study_7, under100k, under1m
##   Estimation Procedure  : Stratified crossvalidation (1 x 10 folds)
```

OpenML: Download a Task II

```
> oml.task$input$data.set

##
## Data Set "anneal" :: (Version = 2, OpenML ID = 1)
##   Default Target Attribute: class

> oml.task$input$estimation.procedure

##
## Estimation Method :: crossvalidation
##  Parameters:
##  number_repeats = 1
##  number_folds = 10
##  stratified_sampling = true

> oml.task$input$evaluation.measures

## [1] "predictive_accuracy"

> oml.task$input$target.features

## [1] "class"
```

OpenML: Run a Task

```
> res1 = runTaskMlr(oml.task, makeLearner("classif.rpart"))
> res2 = runTaskMlr(oml.task, makeLearner("classif.randomForest"))
> bench = mergeBenchmarkResultLearner(
+   res1$mlr.benchmark.result,
+   res2$mlr.benchmark.result
+ )
```

OpenML: Run a Task

```
> plotBMRBoxplots(bench)
```