

MACHINE LEARNING IN R: PACKAGE MLR

Bernd Bischl
Computational Statistics, LMU



ABOUT

- Project home page

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

- ▶ **Tutorial** for online viewing / download, including many examples
- ▶ R documentation rendered in HTML
- ▶ If you are interested you can ask questions in the github issue tracker
- 8-10 main developers, quite a few contributors, 4 GSOC projects in 2015/16 and one coming in 2017
- About 20K lines of code, 8K lines of unit tests

MOTIVATION

THE GOOD NEWS

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

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

THE BAD NEWS

- Some packages API is “just different”
- Functionality is always package or model-dependent, even though the procedure might be general
- No meta-information available or buried in docs

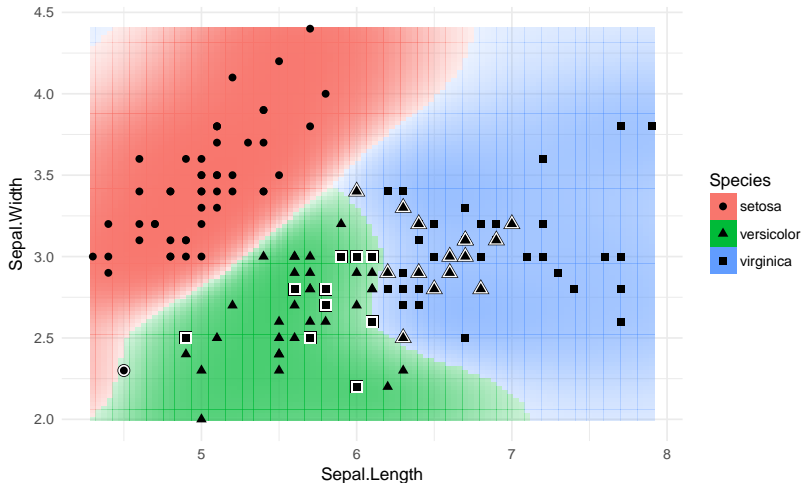
Our goal: A domain-specific language for many machine learning concepts!

MOTIVATION: MLR

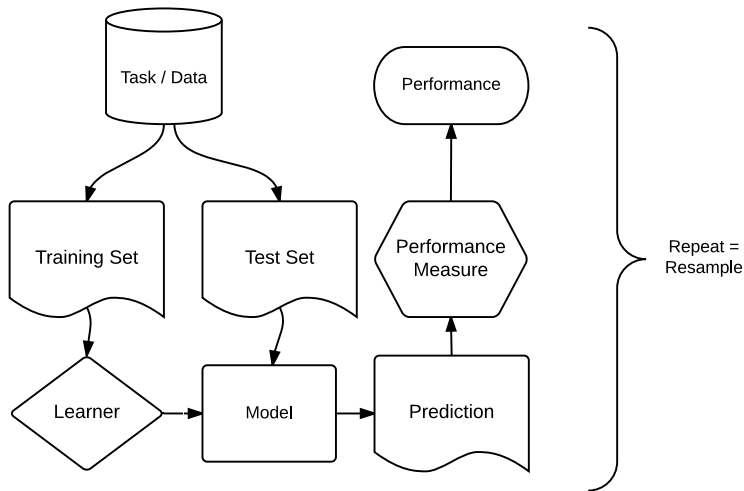
- Unified interface for the basic building blocks: tasks, learners, resampling, hyperparameters, ...
- Reflections: nearly all objects are queryable (i.e. you can ask them for their properties and program on them)
- The OO-structure allows many generic algorithms:
 - ▶ Bagging
 - ▶ Stacking
 - ▶ Feature Selection
 - ▶ ...
- Easily extensible via S3
 - ▶ Extension is not covered here, but explained in detail in the online tutorial
 - ▶ You do not need to understand S3 to use `mlr`
 - ▶ Wondering why we don't use S4? We care about code bloat and speed.

ksvm: fit=FALSE

Train: mmce=0.18; CV: mmce.test.mean=0.22



BUILDING BLOCKS



- `mlr` objects: tasks, learners, measures, resampling instances.

WHAT LEARNERS ARE AVAILABLE? I

CLASSIFICATION (82)

- LDA, QDA, RDA, MDA
- Trees and forests
- Boosting (different variants)
- SVMs (different variants)
- ...

REGRESSION (63)

- Linear, lasso and ridge
- Boosting
- Trees and forests
- Gaussian processes
- ...

CLUSTERING (9)

- K-Means
- EM
- DBscan
- X-Means
- ...

SURVIVAL (15)

- Cox-PH
- Cox-Boost
- Random survival forest
- Penalized regression
- ...

We can explore them on the webpage – or ask `mlr`

WHAT LEARNERS ARE AVAILABLE? II

```
> # list all classification learners which can predict probabilities
> # and allow multiclass classification
> listLearners("classif",
+   properties = c("prob", "multiclass"))[1:5, c(-2, -5, -16)]
```

##		class	short.name	package	type	installed	numerics
## 1		classif.avNNet	avNNet	nnet	classif	TRUE	TRUE
## 2		classif.bdk	bdk	kohonen	classif	TRUE	TRUE
## 3		classif.boosting	adabag	adabag,rpart	classif	TRUE	TRUE
## 4		classif.C50	C50	C50	classif	TRUE	TRUE
## 5		classif.cforest	cforest	party	classif	TRUE	TRUE
##	factors	ordered	missings	weights	prob	oneclass	twoclass
## 1	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
## 2	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
## 3	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
## 4	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE
## 5	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE
##	class.weights	featimp	se	lcens	rcens	icens	
## 1	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	
## 2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	
## 3	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	
## 4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	
## 5	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	

PARAMETER ABSTRACTION

- Extensive meta-information for hyperparameters available: storage type, constraints, defaults, dependencies
- Automatically checked for feasibility
- You can program on parameters!

```
> getParamSet(lrn)
```

##	Type	len	Def	Constr	Req	Tunable	Trafo
## type	discrete	-	C-classification	C-classification,nu-classification	-	TRUE	-
## cost	numeric	-	1	0 to Inf	Y	TRUE	-
## nu	numeric	-	0.5	-Inf to Inf	Y	TRUE	-
## class.weights	numericvector	<NA>	-	0 to Inf	-	TRUE	-
## kernel	discrete	-	radial	linear,polynomial,radial,sigmoid	-	TRUE	-
## degree	integer	-	3	1 to Inf	Y	TRUE	-
## coef0	numeric	-	0	-Inf to Inf	Y	TRUE	-
## gamma	numeric	-	-	0 to Inf	Y	TRUE	-
## cachesize	numeric	-	40	-Inf to Inf	-	TRUE	-
## tolerance	numeric	-	0.001	0 to Inf	-	TRUE	-
## shrinking	logical	-	TRUE	-	-	TRUE	-
## cross	integer	-	0	0 to Inf	-	FALSE	-
## fitted	logical	-	TRUE	-	-	FALSE	-
## scale	logicalvector	<NA>	TRUE	-	-	TRUE	-

BENCHMARKING AND MODEL COMPARISON I

BENCHMARKING

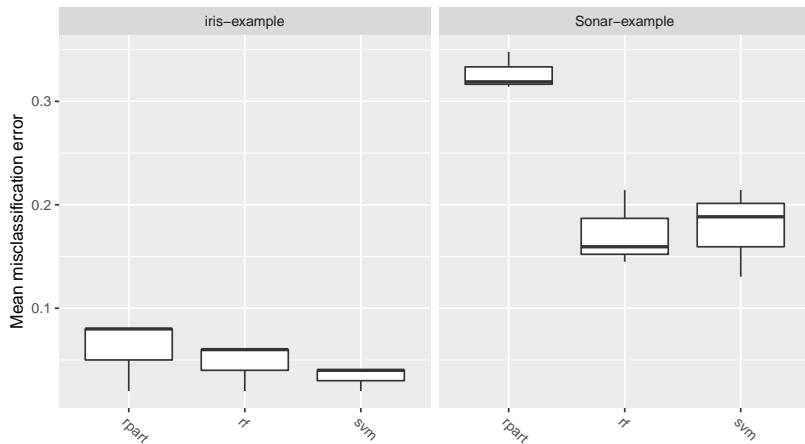
- Comparison of multiple models on multiple data sets
- Aim: Find best learners for a data set or domain, learn about learner characteristics, ...

```
> # these are predefined in mlr for toying around:  
> tasks = list(iris.task, sonar.task)  
> learners = list(  
+   makeLearner("classif.rpart"),  
+   makeLearner("classif.randomForest", ntree = 500),  
+   makeLearner("classif.svm")  
+ )  
>  
> rdesc = makeResampleDesc("CV", iters = 3)  
> br = benchmark(learners, tasks, rdesc)
```

Container object: Results, individual predictions, ...

BENCHMARKING AND MODEL COMPARISON II

```
> plotBMRBoxplots(br)
```



HYPERPARAMETER TUNING

TUNING

- Used to find “best” hyperparameters for a method in a data-dependent way
- General procedure: Tuner proposes param point, eval by resampling, feedback value to tuner

GRID SEARCH

- Basic method: Exhaustively try all combinations of finite grid
 \leadsto Inefficient, combinatorial explosion, searches irrelevant areas

RANDOM SEARCH

- Randomly draw parameters
 \leadsto Scales better than grid search, easily extensible

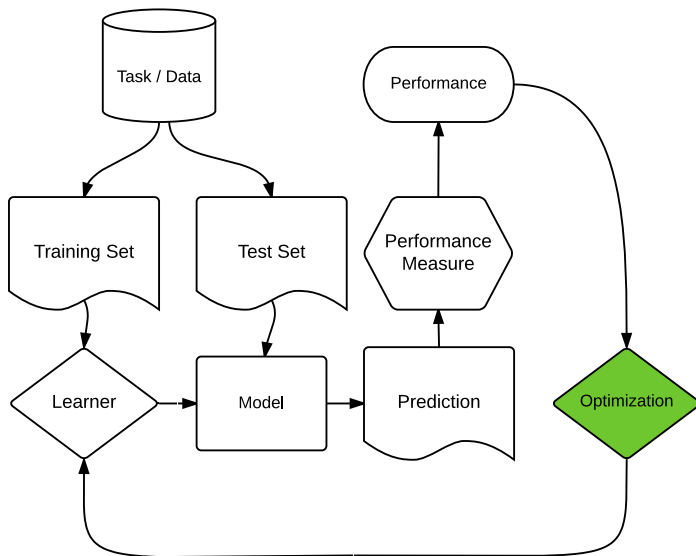
AUTOMATIC MODEL SELECTION

PRIOR APPROACHES:

- Looking for the silver bullet model
 - ~ Failure
- Exhaustive benchmarking / search
 - ~ Per data set: too expensive
 - ~ Over many: contradicting results
- Meta-Learning:
 - ~ Failure
 - ~ Usually not for preprocessing / hyperparameters

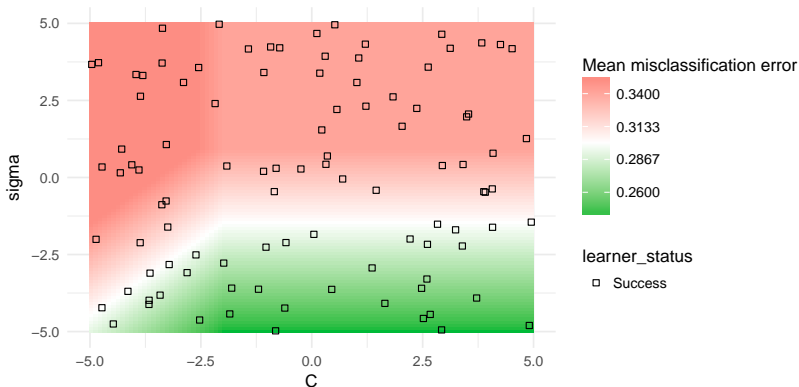
GOAL: Data dependent + Automatic + Efficient

ADAPTIVE TUNING



TUNING EXAMPLE

```
> ps = makeParamSet(  
+   makeNumericParam("C", lower = -5, upper = 5, trafo = function(x) 2^x),  
+   makeNumericParam("sigma", lower = -5, upper = 5, trafo = function(x) 2^x)  
+ )  
> ctrl = makeTuneControlRandom(maxit = 100L)  
> rdesc = makeResampleDesc("CV", iters = 2L)  
> res = tuneParams("classif.ksvm", task = pid.task, control = ctrl,  
+   resampling = rdesc, par.set = ps, show.info = FALSE)
```



PARALLELIZATION

- We use our own package: `parallelMap`
- Setup:

```
> parallelStart("multicore")  
> benchmark(...)  
> parallelStop()
```

- Backends: `local`, `multicore`, `socket`, `mpi` and `batchtools`
- The latter means support for: makeshift SSH-clusters, Docker swarm and HPC schedulers like SLURM, Torque/PBS, SGE or LSF
- Levels allow fine grained control over the parallelization
 - ▶ `mlr.resample`: Job = “train / test step”
 - ▶ `mlr.tuneParams`: Job = “resample with these parameter settings”
 - ▶ `mlr.selectFeatures`: Job = “resample with this feature subset”
 - ▶ `mlr.benchmark`: Job = “evaluate this learner on this data set”

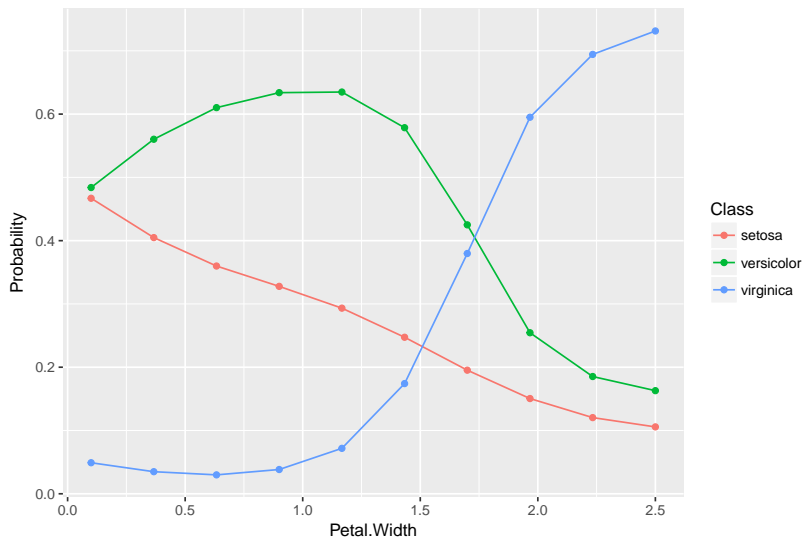
PARTIAL PREDICTIONS PLOTS I

PARTIAL PREDICTIONS

- Estimate how the learned prediction function is affected by one or more features.
- Displays marginalized version of the predictions of one or multiple effects.
- Reduce high dimensional function estimated by the learner.

```
> library(kernlab)
> lrn.classif = makeLearner("classif.svm", predict.type = "prob")
> fit.classif = train(lrn.classif, iris.task)
> pd = generatePartialDependenceData(fit.classif, iris.task, "Petal.Width")
>
> plotPartialDependence(pd)
```

PARTIAL PREDICTIONS PLOTS II



MLR LEARNER WRAPPERS I

WHAT?

- Extend the functionality of learners by adding an `mlr` wrapper to them
- The wrapper hooks into the `train` and `predict` of the base learner and extends it
- This way, you can create a new `mlr` learner with extended functionality
- Hyperparameter definition spaces get joined!

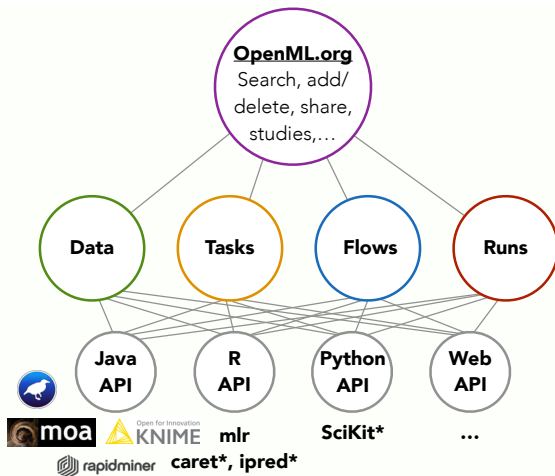
MLR LEARNER WRAPPERS II

AVAILABLE WRAPPERS

- PREPROCESSING: PCA, normalization (z-transformation)
- PARAMETER TUNING: grid, optim, random search, genetic algorithms, CMAES, iRace, MBO
- FILTER: correlation- and entropy-based, χ^2 -test, mRMR, ...
- FEATURE SELECTION: (floating) sequential forward/backward, exhaustive search, genetic algorithms, ...
- IMPUTE: dummy variables, imputations with mean, median, min, max, empirical distribution or other learners
- BAGGING to fuse learners on bootstrapped samples
- STACKING to combine models in heterogenous ensembles
- OVER- AND UNDERSAMPLING for unbalanced classification

OPENML

Main idea: Make ML experiments reproducible, computer-readable and allow collaboration with others.



OPENML R-PACKAGE I

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

TUTORIAL

- `http://openml.github.io/openml-r`
- Caution: Work in progress

CURRENT API IN R

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

THERE IS MORE ...

- Clustering and Survival analysis
- Regular cost-sensitive learning (class-specific costs)
- Cost-sensitive learning (example-dependent costs)
- ROC and learning curves
- Imbalancy correction
- Multi-Label learning
- Bayesian optimization
- Multi-criteria optimization
- Ensembles, generic bagging and stacking
- Some interactive plots with ggvis
- ...

OUTLOOK

WE ARE WORKING ON

- Even better tuning system
- More interactive and 3D plots
- Large-Scale learning on databases
- Keeping the data on hard disk & distributed storage
- Time-Series tasks
- Large-Scale usage of OpenML
- auto-mlr
- ...

Thanks!