Package mlr: Machine Learning in R

Michel Lang^{1,*}, Jakob Richter¹, Bernd Bischl^{1,*}

1. TU Dortmund University *Contact author: {lang,bischl}@statistik.tu-dortmund.de

Keywords: machine learning, model selection, parameter tuning, variable selection, parallelization, survival analysis

Constructing a suitable machine learning model for a given data set or conducting large-scale comparison experiments in this domain can be a tedious task in *R* for various reasons: First, there is no standardized interface for learning algorithms in *R*. The technical ins and outs of each model and operation must be understood and unified for comparison. Many implementations additionally require some sort of special treatment, e.g. transformations of the input data or output results. For benchmark experiments, all this must be embedded in a resampling strategy, where every learner works on the same training sets and is evaluated on the same test sets. More complex statistical learners offer a large set of arguments to allow fine-grained control of the algorithm. For a fair comparison, the arguments must be systematically varied, i.e. using a grid, or, as more efficient approach, tuned by a modern algorithm configurator. Variable reduction and selection is another routine matter. Combining all of these operations with a larger number of machine learning models is not only time-consuming and error-prone to program, but can also easily lead to runtime issues for more comprehensive benchmark studies.

The package **mlr** [3] tries to solve this problem by providing abstractions for learning task, learning machines, resampling strategies, performance measures, tuning algorithms, variable selection methods and other common operations. Its intent is to offer a clean, easy-to-use and flexible domain specific language for machine learning experiments in *R*. It currently offers around 30 classifiers, 20 regression models, most well-known resampling strategies and all popular performance measures. There is also a novel support for survival models and cost-sensitive learning. The package provides several hyperparameter tuning algorithms. These range from very simple random searches to modern approaches based on evolutionary strategies or iterated f-racing. Variable selection is possible through various filter and wrapper approaches.

The clean, abstract interface enables learning algorithms to be enhanced or extended, by chaining the basic method with other useful operations. Examples are generic bagging, adding feature imputation (or other preprocessing) or adding self-tuning.

Parallelization can be triggered on many different levels of a typical **mlr** workflow, e.g. for outer or inner resampling, tuning or variable selection operations. Many popular parallelization back-ends, e.g. **parallel** [4] or **BatchJobs** [2], can easily be selected by the user, as **mlr** internally uses the **parallelMap** [1] package for this.

References

- [1] Bischl, B. and M. Lang (2014). *Unified interface to some popular parallelization back-ends for inter-active usage and package development.*
- [2] Bischl, B., M. Lang, and O. Mersmann (2014). BatchJobs: Batch computing with R.
- [3] Bischl, B., M. Lang, and J. Richter (2014). mlr: Machine Learning in R.
- [4] R Core Team (2014). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.