mlr3shiny: A graphical user interface for easy machine learning in R

31 January 2020

Introduction and Statement of Need

Within recent years, machine learning (ML) applications have entered many industries and open source software is readily available such as the powerful object-orientated and standardized ML framework mlr3 (Lang et al. 2019) an evolution of mlr (Bischl et al. 2016) dating back to 2010 (Szepannek et al. 2010) and based on R (R Core Team 2019). Despite the easy acces to frameworks, their advanced and highly parameterizable functionalities may quickly overwhelm the soaring number of inexperienced newcomers to machine learning.

For this reason, the R package `mlr3shiny` has been developed as a simple accessible and user-friendly web-application, combining the graphical user interface (GUI) offered by Shiny (Chang et al. 2019) with the state-of-the-art ML functionalities provided by mlr3. The resulting application enables users to set up machine learning workflows in a very fast way while familiarizing with the basic steps of a machine learning process. Thus, modern ML functionalites can be referenced and applied in an easy to use point-and-click fashion freeing users from coding in R. Especially the latter makes this new package also a valuable tool to be used for teaching introductory ML courses as it is done e.g. at Stralsund University of Applied Sciences (Germany). Working with `mlr3shiny` facilitates the next step to leverage the entire power of mlr3.

Description of Features

The layout of the application visually guides its users chronologically through the different steps of the machine learning workflow as presented in Figure 1.

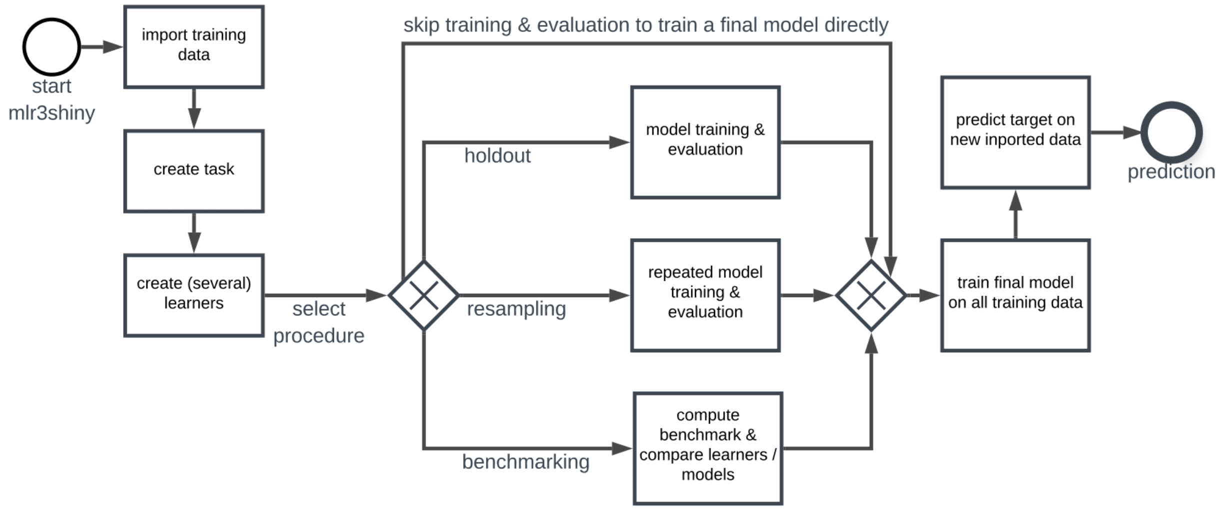
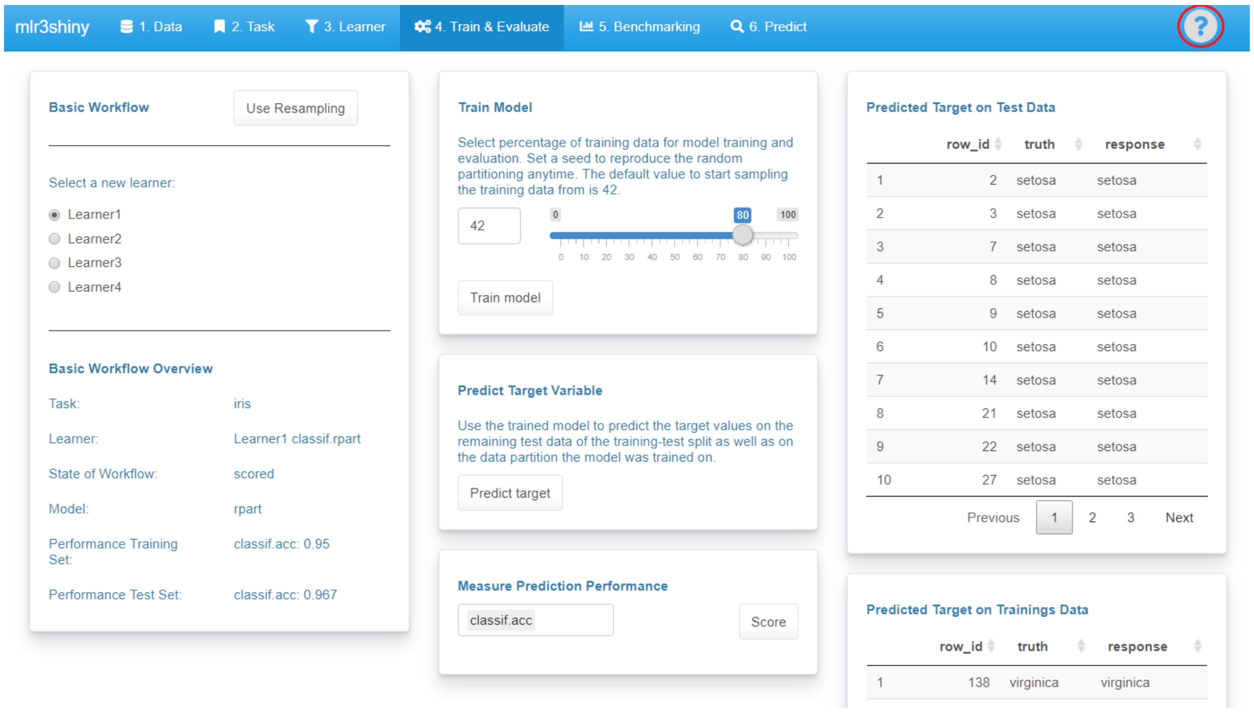

start mlr3shiny
import training data
create task
create (several) learners
select procedure
skip training & evaluation to train a final model directly
holdout
model training & evaluation
resampling
repeated model training & evaluation
benchmarking
compute benchmark & compare learners / models
train final model on all training data
predict target on new inported data
prediction

Figure 1: Workflow with mlr3shiny

Figure 2 demonstrates the GUI of the application which can be started via:

```
mlr3shiny::launchMlr3Shiny()
```



Basic Workflow

Select a new learner:

- Learner1
- Learner2
- Learner3
- Learner4

Basic Workflow Overview

Task:	iris
Learner:	Learner1 classif.rpart
State of Workflow:	scored
Model:	rpart
Performance Training Set:	classif.acc: 0.95
Performance Test Set:	classif.acc: 0.967

Train Model

Select percentage of training data for model training and evaluation. Set a seed to reproduce the random partitioning anytime. The default value to start sampling the training data from is 42.

42   0   80   100

Predict Target Variable

Use the trained model to predict the target values on the remaining test data of the training-test split as well as on the data partition the model was trained on.

Measure Prediction Performance

classif.acc

Predicted Target on Test Data

	row_id	truth	response
1	2	setosa	setosa
2	3	setosa	setosa
3	7	setosa	setosa
4	8	setosa	setosa
5	9	setosa	setosa
6	10	setosa	setosa
7	14	setosa	setosa
8	21	setosa	setosa
9	22	setosa	setosa
10	27	setosa	setosa

Predicted Target on Trainings Data

	row_id	truth	response
1	138	virginica	virginica

Figure 2: The GUI of `mlr3shiny`

The ordered tabs of the application represent the core functionalities of `mlr3shiny` corresponding to the different steps of the workflow as given in Figure 1. A typical ML process starts as follows:

- **Data** import
- Creation of a supervised learning **task** which combines the data with required meta information such as the target variable and the type of task to be performed (either regression or classification).
- Selection of the desired set of **learners** (algorithms) to be used in the ML experiment. Currently supported are linear and logistic regression, decicion trees (Therneau and Atkinson 1997), random forests (Wright and Ziegler 2017) and support vector machines (Meyer et al. 2019). Each learner can be parameterized. In contrast to other GUI based Ml frameworks such as rattle (Williams 2009), several instances of the same learner class can be defined in order to compare their performance.

The next three tabs correspond to the different branches in the workflow diagram:

- Typically, a single learner out of the specified ones is **trained and evaluated** on the holdout sample with the holdout sample being a randomly selected portion of the data. Figure 2 depicts this step of the modelling process for a classification task. Via *use resampling* resampling techniques such as cross-validation or bootstrap can be used. A trained model can be further applied to both training and holdout data (predict) and further evaluated (score) on both data sets w.r.t some performance metric. At the moment of writing the metrics classification error (ce) and accuracy (ac) are supported for classification tasks and mean squared error (mse) as well as mean absolute error (mae) for regression tasks.
- Several learners can be **benchmarked** and compared against each other w.r.t. a selected performance measure and one of the previuosly mentioned resampling strategies.
- Finally, a model can be applied to **predict** new data. For an optimal performance, it is meaningful to first retrain the learner on the entire data that have been used so far while the former two steps rather serve to evaluate a model with its parameters and assess its predictive performance.

The whole application is designed in such a way that also users new to this field can successfully conduct machine learning experiments. All steps are accompagnied by a help page (red circled question mark in the top right corner) which provides a brief general *description* of the methodology as well as the *functionality* for each step. As an important feature, results can be saved for further use: either benchmark results, a final retrained model on the entire data or the predictions on new data. To guard new users from committing easy mistakes and to provide orientation, default input values and examples are given in accordance with current practices. On top, it is ensured that conducted experiments are replicable.

Summary

In summary, the R package `mlr3shiny` provides a user-friendly web-application that implements the basic steps of a machine learning workflow. As such the access to ML for users unfamiliar with R or coding is facilitated and an easy entry into the universe of machine learning is provided based on the workflow of one of the state-of-the-art ML frameworks.

Acknowledgements

We greatly acknowledge Michel Lang and the mlr3 developer team for helpful discussions and feedback on the design and the architecure of the application. We would further like to acknowledge Rabea Aschenbruck for her constructivism and Stralsund University of Applied Sciences for providing the creative environment that led to this development.

References

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "Mlr: Machine Learning in R." *Journal of Machine Learning Research* 17 (170): 1–5. http://jmlr.org/papers/v17/15-066.html.

Chang, Winston, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. 2019. "Shiny: Web Application Framework for R." https://CRAN.R-project.org/package=shiny.

Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. "mlr3: A Modern Object-Oriented Machine Learning Framework in R." *Journal of Open Source Software*, December. https://doi.org/10.21105/joss.01903.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang, and Chih-Chen Lin. 2019. "E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien." https://CRAN.R-project.org/package=e1071.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Szepannek, Gero, Matthias Gruhne, Bernd Bischl, Sebastian Krey, Tamas Harczos, Frank Klefenz, Christian Dittmar, and Claus Weihs. 2010. "Perceptually Based Phoneme Recognition in Popular Music." In *Classification as a Tool for Research. Studies in Classification, Data Analysis, and Knowledge Organization*, edited by Locarek-Junge H. and Weihs C., 367–77. Heidelberg: Springer. https://doi.org/10.1007/978-3-642-10745-0_83.

Therneau, Terry M., and Elizabeth J. Atkinson. 1997. "An Introduction to Recursive Partitioning Using the Rpart Routines. Divsion of Biostatistics 61."

Williams, Graham J. 2009. "Rattle: A Data Mining GUI for R." *The R Journal* 1 (2): 45–55. https://doi.org/10.32614/RJ-2009-016.

Wright, Marvin N., and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.