Teaching Machine Learning with mlr3 using Shiny

Gero Szepannek [1] **Laurens Tetzlaff** [2] **Alexander Frahm** [3] **Karsten Lübke** [4]

Abstract

The command-line use of R and Python can be a barrier for novice learners in the field of machine learning. Lowering this bar for nontechnical students may help them to understand important core principles of machine learning like training and evaluation of models. Shiny applications can provide an user friendly graphical interface to the machine learning workflow. mlr3shiny uses the functionalities of the R-package mlr3 and provides teachers and learners of machine learning the opportunity to explore the machine learning workflow without the need to learn R programming first.

1. Motivation

The argument of Wild et al. (2017) for teaching the bootstrap method ”With the rapid, ongoing expansions in the world of data, we need to devise ways of getting more students much further, much faster” can also be applied to teaching machine learning. So in teaching machine learning we should think about ways ”to minimize prerequisites to research” (Cobb, 2015).

In the recent years numerous frameworks for automated machine learning (autoML Hutter et al., 2018) have been developed. These tools make it easy for users to create machine learning models which might be dangerous in case of a lack of knowledge on the underlying methodology. Moreover, software for machine learning is quite often implemented in R or Python. But as argued by e.g. (Gerbing, 2021) using command-line environments requires some skills and concepts e.g. business students are not familiar with. Whereas some courses are aiming at teaching programming it is not a learning outcome for all courses. So for some courses the increased cognitive load by programming is not needed and can be reduced by graphical user interfaces. This allows to focus on teaching the conceptual ideas of machine learning by an example.

[*]Equal contribution [1]Stralsund University of Applied Sciences, Germany [2]Jheronimus Academy of Data Science, Netherlands [3]cronos Unternehmensberatung GmbH, Germany [4]FOM University of Applied Sciences, Germany. Correspondence to: Gero Szepannek <gero.szepannek@hochschule-stralsund.de>.

Proceedings of the 3^{rd} Teaching in Machine Learning Workshop, PMLR, 2022. Copyright 2022 by the author(s).

Shiny (Chang et al., 2021) provides an opportunity to build an interactive, dynamic, user-friendly and visually appealing web applications for teaching (Doi et al., 2016). By using a Shiny application for teaching machine learning by example we can reduce possible technical overload while keeping the opportunity to inspect the resulting R code.

2. Machine Learning with mlr3 and mlr3shiny

The deployed framework `mlr3shiny` (Tetzlaff & Szepannek, 2022) for teaching is based on one of the most powerful state-of-the-art machine learning frameworks `mlr3` (Lang et al., 2019), an evolution of `mlr` (Bischl et al., 2016) which dates back to 2010 (Szepannek et al., 2010). For students it is important to understand the different process steps of the model development. Following mlr3, these steps can be summarized as follows:

- Define a task (data plus target variable).
- Define a learner.
- Parameterize the learner.
- Define one or several performance measure(s) of interest.
- Define a resampling strategy to split the data into training and validation data.
- Train and evaluate the learner based on the data as given by the resampling strategy and the performance measure(s) from the previous steps.
- Optimize the performance of the data by tuning the parameters of the learner.

One of the main learning goals for students is to gain conceptual understanding of the consequences of model complexity. Of course this is on one hand related to the hyper-parameters of the used method and on the other hand to over-fitting and under-fitting. Experienced users of course know this and use validation data to deal with these issues. For novice learners this seems to be a hard task, also noted by (Zieffler et al., 2021) for secondary teachers.

3. Teaching Example

3.1. Use Case: Credit Scoring

In this section a case study is presented that can be used to familiarize students with the machine learning model development process and introduce the relevant concepts as discussed above. The use case is driven by the business case of credit scoring. So it may be used e.g. in an introductory data science class for business related majors. It turned out that this is possible using the use case as presented below even in a self study setting.

For this sake the publicly available German credit data (Hoffmann, 1990) are referenced as provided by the UCI machine learning benchmark repository (Dua & Graff, 2017). The study is described using mlr3shiny version 0.3.0 from GitHub [1]. The data are already included in mlr3 and are thus chosen for this case study such that readers are able to mimic the different modelling steps.

The aim is to build a binary classification model to predict bad (defaulted) customers before granting a credit. While at the same time as many bad customers should be detected the percentage of rejections among the good customers should be as small as possible. This need as motivated from a business perspective not only emphasizes the need of different performance measures (here: sensitivity and specificity) but also intuitive illustrates the trade of between both.

In the example different models are compared. In addition tuning is demonstrated using the model agnostic classification threshold parameter and students can easily understand the impact of tuning on the performance measures.

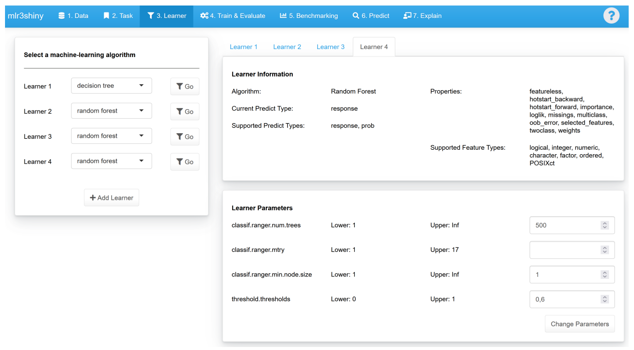
Finally, the code can be extracted which allows students to try first steps towards programming.

3.2. Defining a task

The data can be selected in the **2. Task** tab in the horizontal bar top of the app. In the *Task processing* window *bad* is selected as positive class of the target variable and the features *installment rate*, *number credits* and *present residence* are dropped.

3.3. Defining a learner

In a second step, two default learners are instantiated in the **3. Learner** tab [2]: a *decision tree* and a *random forest* without any additional modifications to their configuration (i.e. their default parameters) via the *Learner Parameters* window (cf. figure 1, bottom right).



Select a machine-learning algorithm

Learner 1	decision tree	Go
Learner 2	random forest	Go
Learner 3	random forest	Go
Learner 4	random forest	Go

Learner Information

Algorithm:	Random Forest	Properties:	featureless, hotstart_backward, hotstart_forward, importance, loglik, missings, multiclass, oob_error, selected_features, twoclass, weights
Current Predict Type:	response		
Supported Predict Types:	response, prob		
		Supported Feature Types:	logical, integer, numeric, character, factor, ordered, POSIXct

Learner Parameters

classif.ranger.num.trees	Lower: 1	Upper: Inf	500
classif.ranger.mtry	Lower: 1	Upper: 17	
classif.ranger.min.node.size	Lower: 1	Upper: Inf	1
threshold.thresholds	Lower: 0	Upper: 1	0,6

Figure 1. Learner Tab of the mlr3shiny GUI.

3.4. Performance Evaluation

The performance of both learners is compared in the **5. Benchmarking** tab using a *holdout* resampling strategy and the (default) *Resampling Parameter Setting* with a 0.667 fraction of training data.

For the performance benchmark two *Measure(s of) Aggregated Performance* are selected: *true positive rate (TPR/recall/sensitivity)* and *true negative rate (TNR/specificity)*. While the first measure reflects the percentage of bad (defaulted) credits rejected by the model the second one returns the number of positive (correctly repaid) credits that would have been accepted by the model (cf. above). The *Aggregated Benchmark Result* window shows that the random forest model outperforms the decision tree with regard to both measures and is thus preferable.

3.5. Hyper-parameter Tuning

Nonetheless, in contrast to a high specificity of more than 90%, only less than half of the bad credits (44%) will be rejected by the model. Consequently, two additional random forests learners are created in the **3. Learner** tab and the *threshold.thresholds* parameter in the *Learner Parameters* is reduced to either 0.45 or 0.4.

After a second benchmark it turns out that for a reduced classification threshold of 0.45 the TPR (recall) can be improved by 12% (from 0.47 to 0.58) while at the same time the TNR (specificity) reduces from 0.901 to 0.858. For a further decrease of the threshold value to 0.4 the TPR improves another 5% to 0.63 but at the same time the TNR decreases to a value below 0.8.

3.6. Final Model

After a final decision for one of the models (here the forest with a threshold of 0.45) a final model selected in the *Apply*

[1] `https://github.com/LamaTe/mlr3shiny`

[2] Currently supported learners are: linear and logistic regression, decision trees (Therneau & Atkinson, 1997), support vector machines (Lang et al., 2019), random forests (Wright & Ziegler, 2017) and xgboost (Chen & Guestrin, 2016).

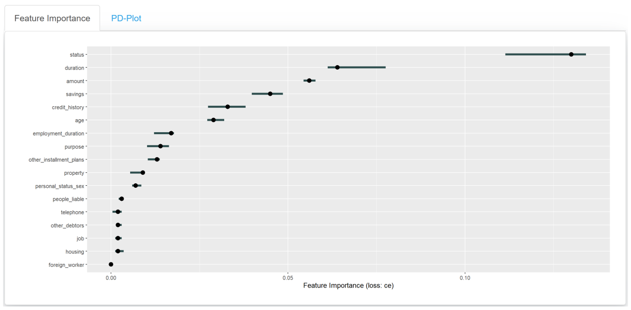


Figure 2. Feature Importance plot.

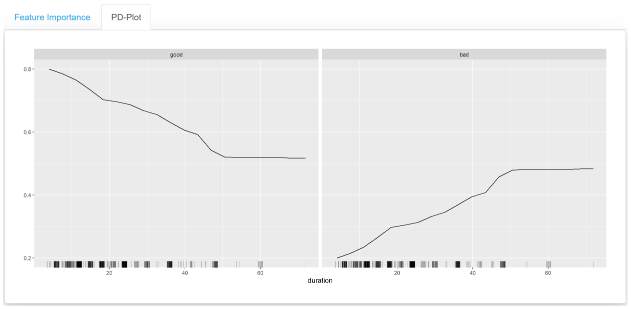


Figure 3. Partial Dependence plot for the variable duration.

best learner on new data window of the **6. Predict Tab** and then retrained on the entire data (training + holdout) . Afterwards, be data can be imported and predicted by the model. Both the model and the predictions can be exported for further use outside of mlr3shiny.

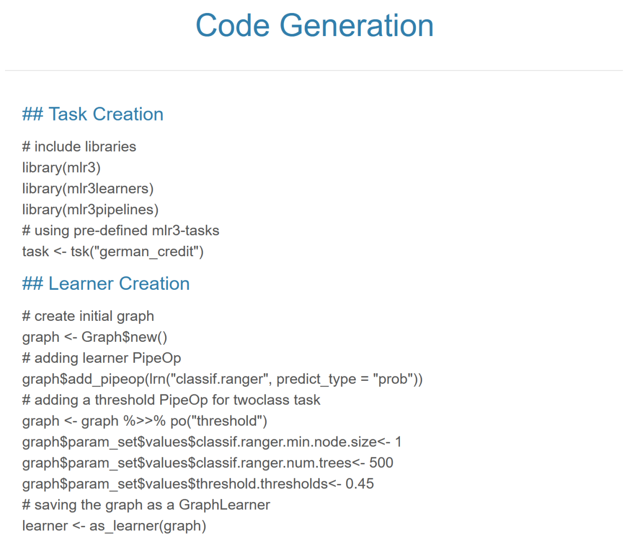
As a supplement to the standard modelling process the **7. Explain Tab** allows to apply to standard model agnostic methods from the domain of explainable AI (variable importance and partial dependence plots, for an overview cf e.g. Bücker et al., 2021). This not only allows to introduce basic ideas of XAI but further helps sensitizing students for the need to develop some understanding of the model.

In the example it can be seen that the most important variables are *status*, *duration* and *amount* (cf. fig. 2)[3] while from the partial dependence plot (fig. 3) for the variable duration it can be seen that the risk increases if a customer desires a longer time to pay the credit back.

3.7. Code Extraction

As an additional feature, the code needed for model training and evaluation can be extracted by *Show the Code* in the **6. Predict Tab** (cf. fig. 4). This running code snipped can be used from students for first steps in mlr3 and be the perfect starting point for any future self studies.

[3]For the choice of *ce* as loss function.



Code Generation

Task Creation

```
# include libraries
library(mlr3)
library(mlr3learners)
library(mlr3pipelines)
# using pre-defined mlr3-tasks
task <- tsk("german_credit")
```

Learner Creation

```
# create initial graph
graph <- Graph$new()
# adding learner PipeOp
graph$add_pipeop(lrn("classif.ranger", predict_type = "prob"))
# adding a threshold PipeOp for twoclass task
graph <- graph %>>% po("threshold")
graph$param_set$values$classif.ranger.min.node.size<- 1
graph$param_set$values$classif.ranger.num.trees<- 500
graph$param_set$values$threshold.thresholds<- 0.45
# saving the graph as a GraphLearner
learner <- as_learner(graph)
```

Figure 4. First lines from the extracted code.

3.8. Summary and Outlook

The case study illustrates how models can be selected and optimized with regard to appropriate performance measures. Whenever several measures are considered at the same time, a compromise has to be made for the trade of between the different measures. The final model choice should respect the particular requirements to model performance of the given context or business situation. Note that for this simple use case only the threshold parameter has been tuned. For further information on the algorithm specific parameters, the app refers a user to the corresponding documentations. Additionally to a local installation deployment of the app via a Shiny Server or an shinyapps.io account are possible.

4. Evaluation

`mlr3shiny` has, e.g., been used in an introductory data science class for business informatics by the first author in spring 2022. According to the feedback from the students the tool offers a ”straightforward introduction” to machine learning and a ”very structured approach to understand the different steps for building and validating a model”. In addition, the students positively mentioned the ”possibility to interactively try out different hyperparameter settings” and learn about their effects on the model’s performance as well as the included data sets which ”make it easy to take some first steps”. While the student feedback is positive a next step will be a formal evaluation of the attitudes and learning outcomes.

5. Discussion

In a world of ubiquitous data and algorithmic decision making instructors should carefully think about the desired learning outcomes of their course. For novice learners technology tools like the one presented can provide an opportunity to avoid obfuscation. By introducing machine learning by use cases we can give students an opportunity to learn concepts and at the same time discuss the real world consequences of our decisions within the workflow.

References

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., and Jones, Z. M. mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170):1–5, 2016. URL https://jmlr.org/papers/v17/15-066.html.

Bücker, M., Szepannek, G., Gosiewska, A., and Biecek, P. Transparency, auditability and explainability of machine learning models in credit scoring. *Journal of the Opetrational Research Society*, 2021. doi: 10.1080/01605682.2021.1922098.

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.7.1.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. doi: 10.1145/2939672.2939785.

Cobb, G. Mere renovation is too little too late: We need to rethink our undergraduate curriculum from the ground up. *The American Statistician*, 69(4):266–282, 2015. doi: https://doi.org/10.1080/00031305.2015.1093029.

Doi, J., Potter, G., Wong, J., Alcaraz, I., and Chi, P. Web application teaching tools for statistics using r and shiny. *Technology Innovations in Statistics Education*, 9(1), 2016. doi: http://dx.doi.org/10.5070/T591027492.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Gerbing, D. W. Enhancement of the command-line environment for use in the introductory statistics course and beyond. *Journal of Statistics and Data Science Education*, 29(3):251–266, 2021. doi: https://doi.org/10.1080/26939169.2021.1999871.

Hoffmann, H. Die anwendung des cart-verfahrens zur statistischen bonitätsanalyse von konsumentenkrediten. *Zeitschrift für Betriebswirtschaft*, 60:941–962, 1990.

Hutter, F., Kotthoff, L., and Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

Lang, M., Binder, M., Richter, J., Schratz, P., Pfisterer, F., Coors, S., Au, Q., Casalicchio, G., Kotthoff, L., and Bischl, B. mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*, 2019. doi: 10.21105/joss.01903.

Szepannek, G., Gruhne, M., Bischl, B., Krey, S., Harczos, T., Klefenz, F., and Weihs, C. Perceptually based phoneme recognition in popular music. In Locareck-Junge, H. and Weihs, C. (eds.), *Classification as a Tool for Research*, pp. 731–758, 2010. doi: 10.1007/978-3-642-10745-0_83.

Tetzlaff, L. and Szepannek, G. *mlr3shiny: Machine Learning in 'shiny' with 'mlr3'*, 2022. R package version 0.2.0.

Therneau, T. and Atkinson, E. J. *An introduction to recursive partitioning using the rpart routines*, 1997. Divsion of Biostatistics 61.

Wild, C. J., Pfannkuch, M., Regan, M., and Parsonage, R. Accessible conceptions of statistical inference: Pulling ourselves up by the bootstraps. *International Statistical Review*, 85(1):84–107, 2017. doi: https://doi.org/10.1111/insr.12117.

Wright, M. N. and Ziegler, A. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01.

Zieffler, A., Justice, N., delMas, R., and Huberty, M. D. The use of algorithmic models to develop secondary teachers' understanding of the statistical modeling process. *Journal of Statistics and Data Science Education*, 29(1):131–147, 2021. doi: 10.1080/26939169.2021.1900759.