Journal of Machine Learning Research 22 (2021) 1-7 Submitted 3/21; Revised 6/21; Published 6/21

mlr3pipelines – Flexible Machine Learning Pipelines in R

Martin Binder[1] MARTIN.BINDER@STAT.UNI-MUENCHEN.DE
Florian Pfisterer[1] FLORIAN.PFISTERER@STAT.UNI-MUENCHEN.DE
Michel Lang[1] MICHEL.LANG@STAT.UNI-MUENCHEN.DE
Lennart Schneider[1] LENNART.SCHNEIDER@STAT.UNI-MUENCHEN.DE
Lars Kotthoff[2] LARSKO@UWYO.EDU
Bernd Bischl[1] BERND.BISCHL@STAT.UNI-MUENCHEN.DE

[1] *Department of Statistics, LMU Munich, Germany*
[2] *Department of Computer Science, University of Wyoming, USA*

Editor: Alexandre Gramfort

Abstract

Recent years have seen a proliferation of ML frameworks. Such systems make ML accessible to non-experts, especially when combined with powerful parameter tuning and AutoML techniques. Modern, applied ML extends beyond direct learning on clean data, however, and needs an expressive language for the construction of complex ML workflows beyond simple pre- and post-processing. We present `mlr3pipelines`, an R framework which can be used to define linear and complex non-linear ML workflows as directed acyclic graphs. The framework is part of the `mlr3` ecosystem, leveraging convenient resampling, benchmarking, and tuning components.

Keywords: machine learning pipelines, preprocessing, automated machine learning

1. Introduction

As one of the most popular and widely-used software systems for statistics and ML, R (R Core Team, 2020) has several packages that provide a standardized interface for predictive modeling, such as `caret` (Kuhn, 2008), `tidymodels` (Kuhn and Wickham, 2020b), `mlr` (Bischl et al., 2016), and its successor `mlr3` (Lang et al., 2019). But real-world applications often require complex combinations of ML (pre-) processing steps, which can be expressed as a directed acyclic graph (DAG); we will call such graphs ML pipelines or ML workflows. Specifying such a workflow in an ML system without direct support requires error-prone glue code to combine the individual pieces. One particular difficulty is that (in ML) each pipeline operation is not a stateless function application, but consists of a train and predict stage, where the former not only transforms its inputs into an output, but also learns an internal parameter state, which the latter relies on.[1] `mlr3pipelines` provides a domain-specific language which allows building ML pipelines from individual processing operations (`PipeOp`s, also see Figure 1). It ships with a large collection of such operations and allows their custom extension through user-defined operations.

1. This implies that the pipeline idiom in `mlr3pipelines` is quite different compared to `magrittr`, `dplyr`, and `tidymodels`.

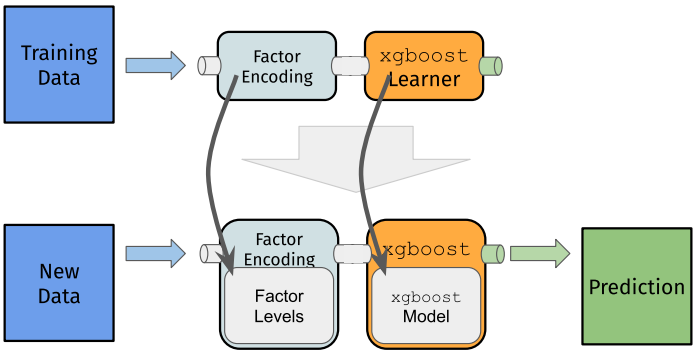


Figure 1: Train and predict steps for a short linear pipeline. Trained `PipeOp`s carry state / parameters (factors and the fitted model) that can be applied to test data.

2. Related Work

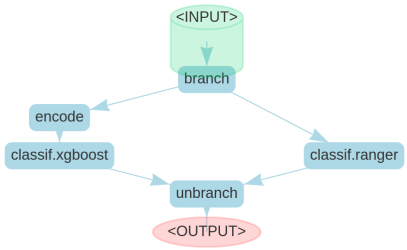
By far, the most widely-used implementation of ML pipelines is Python `scikit-learn`'s (Pedregosa et al., 2011) `pipeline` module (Buitinck et al., 2013). Unlike our software, `scikit-learn` supports only linear pipelines directly, although complex pipelines can be expressed through a wrapper mechanism. Several extensions such as `baikal` (Tineo, 2019) and `neuraxle` (Chevalier et al., 2019) extend scikit-learn's pipelining via a graph-based API similar to ours. `tidymodels` provides the `recipes` R package (Kuhn and Wickham, 2020a) for building linear preprocessing pipelines with limited flexibility and the `workflows` package (Vaughan, 2020) for combining these with models into pipelines. The `mlr` extension `mlrCPO` (Binder, 2021) also focuses on linear pipelines and has limited support for more complex structures. Industry is increasingly providing systems that support ML pipelines, e.g., Microsoft's `ml.net` (Ahmed et al., 2019) for C# and H2O (H2O.ai, 2021) with bindings for Python and R. The `d3m` software (Milutinovic et al., 2017) was developed as part of DARPA's Data Driven Discovery of Models program (Shen, 2018) and includes a pipeline system to combine ML primitives, again without the full flexibility of `mlr3pipelines`. The DAGs in `mlr3pipelines` go beyond simple combinations of preprocessing and ML models. They support ensemble models and conditional branching that can be represented explicitly as part of the graph structure. Existing pipelining frameworks are often limited to passing training or prediction data objects through the pipeline, while `mlr3pipelines` allows for passing arbitrary objects. Some operators, for example, pass on functions, which are then used to influence the behavior of operators later in the graph.

3. Design, Functionality, and Examples

`mlr3pipelines` represents ML workflows as `Graph` objects: DAGs, whose vertices are `PipeOp`s, which represent arbitrary ML processing operations. The pipeline can either be called to train or predict. Inputs and intermediate objects, most commonly data, move along the DAG's edges. When they pass through a vertex, they are processed by the cor-

```
factor_xgboost = po("encode") %>>%
  lrn("classif.xgboost")

pipe = ppl("branch", list(
  xgboost = factor_xgboost,
  ranger = lrn("classif.ranger")
))
```



Listing 1 (left) and Figure 2 (right): Example of a branching pipeline. LHS: The `%>>%` operator builds a linear partial graph. The "ppl branch" template constructs two alternative paths *xgboost* and *ranger*, where the latter does not require factor encoding. A new hyperparameter controls the path through which the data will flow. RHS: The pipeline can be plotted with `pipe$plot(html = TRUE)`.

responding `PipeOp`, and, depending on the call, are either transformed by its `train()` or `predict()` method, where the former also creates the operator's internal state.

This ensures that no information leakage from test data occurs, which is required for the evaluation of predictive systems (Bischl et al., 2012). `mlr3pipelines` and the `mlr3` ecosystem are integrated with each other, so that `mlr3`'s `Learner`s can be used as `PipeOp`s and `Graph`s adhere to the same interface as `mlr3` learners and can be, for example, resampled and tuned just like any other `Learner`. This also enables effortless parallelization of these operations for pipelines. `mlr3pipelines` provides the `%>>%` operator, which concatenates `Graph`s (or `PipeOp`s) into larger `Graph`s. Templates for more complex but frequently used graph patterns are provided through the `ppl()` lookup function. Outputs from different nodes can be combined in non-trivial ways, for example, joining features created by different preprocessing steps, to create non-linear structures. Other examples include alternative path branching (one of several flows is executed, depending on a hyperparameter), ensembling (predictions from different `PipeOp`s are averaged), and stacking (predictions from different `PipeOp`s are combined in another `PipeOp`, usually a `Learner`, to produce a final prediction). Listing 1 shows an example of branching for model and preprocessing selection. Many more examples can be found at `https://mlr3gallery.mlr-org.com/#category:mlr3pipelines`.

Figure 3 shows examples of complex pipeline components. Some of these are already used in other packages in the `mlr3` ecosystem, e.g., `mlr3proba` (Sonabend et al., 2021) uses the pipeline in Figure 3(i). Users can easily implement their own `PipeOp`s and define their exposed hyperparameters, by inheriting from the `PipeOp` class to, for example, implement custom feature extraction and processing.

4. Hyperparameter Tuning and AutoML

Each `Graph` exposes the hyperparameters of its constituent `PipeOp`s for joint tuning via any of the automated tuning methods in `mlr3`. Simple tuners such as grid and random search, as well as advanced black-box optimizers like Bayesian Optimization (Snoek et al., 2012) and Hyperband (Li et al., 2018) are available through `mlr3tuning`. Building upon the branching principle of Listing 1, this allows to build entire AutoML systems by combining

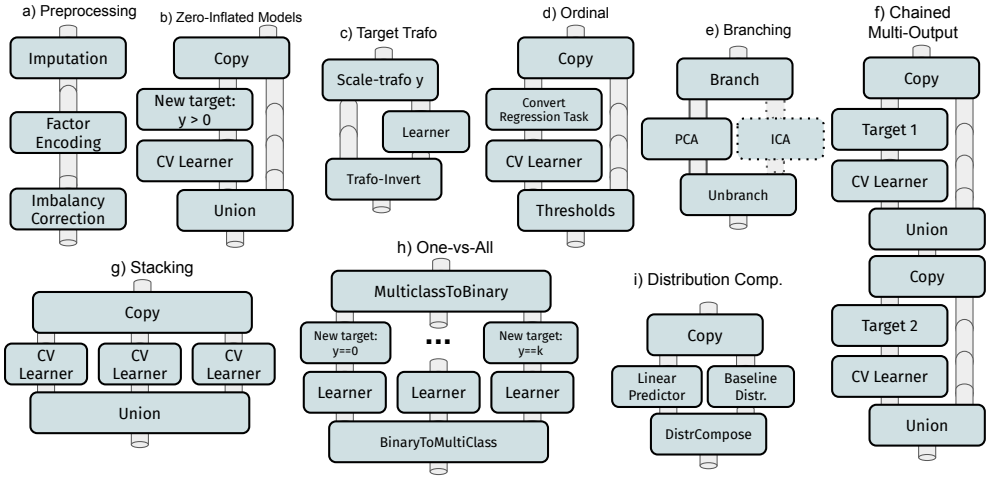


Figure 3: Example pipelines constructed from simple building blocks: (a) typical preprocessing pipeline, (b) zero-inflated data (Zuur et al., 2009), (c) target transformations (scaling to $[0, 1]$) before a model and back afterward, (d) ordinal regression through thresholding, (e) alternative path branching between different options, (f) chaining (Read et al., 2011), (g) stacking (Wolpert, 1992), (h) multiclass classification through ensembling of multiple class-vs-rest binary classifiers, (i) estimation of survival distributions and continuous risk rankings from linear predictors through composition (Sonabend et al., 2021).

multiple learners and preprocessing options. Jointly tuning the selection of these steps with their (subordinate) hyperparameters yields a single pipeline, tailored for a specific task.

5. Availability, Documentation, Code Quality Control

All packages of the `mlr3` ecosystem are released under LGPL-3 on GitHub (`https://github.com/mlr-org`) and on CRAN. Package documentation is available at `https://mlr3pipelines.mlr-org.com` and in the (work-in-progress) mlr3 book (`https://mlr3book.mlr-org.com`), with examples in the mlr3 gallery (`https://mlr3gallery.mlr-org.com`). An extensive suite of unit tests is run on each change via a continuous integration system.

6. Outlook

`mlr3pipelines` is complete and ready for production use. Our focus for future improvements is better integration of automated ML and deep learning (through `mlr3keras` and `mlr3torch`), and leveraging parallel processing specifically for pipelines.

Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. LK is supported by NSF grant #1813537.

References

Zeeshan Ahmed, Saeed Amizadeh, Mikhail Bilenko, Rogan Carr, Wei-Sheng Chin, Yael Dekel, Xavier Dupre, Vadim Eksarevskiy, Senja Filipi, Tom Finley, Abhishek Goswami, Monte Hoover, Scott Inglis, Matteo Interlandi, Najeeb Kazmi, Gleb Krivosheev, Pete Luferenko, Ivan Matantsev, Sergiy Matusevych, Shahab Moradi, Gani Nazirov, Justin Ormont, Gal Oshri, Artidoro Pagnoni, Jignesh Parmar, Prabhat Roy, Mohammad Zeeshan Siddiqui, Markus Weimer, Shauheen Zahirazami, and Yiwen Zhu. Machine learning at Microsoft with ML.NET. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2448–2458, 2019.

Martin Binder. *mlrCPO: Composable Preprocessing Operators and Pipelines for Machine Learning*, 2021. URL `https://CRAN.R-project.org/package=mlrCPO`. R package version 0.3.7-2.

Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Claus Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275, 2012.

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

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: Experiences from the scikit-learn project. In *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*, 2013.

Guillaume Chevalier, Alexandre Brillant, and Éric Hamel. Neuraxle - a Python framework for neat machine learning pipelines, 09 2019. URL `https://github.com/Neuraxio/Neuraxle`.

H2O.ai. *h2o software*, 10 2021. URL `https://github.com/h2oai/h2o-3`. H2O version 3.32.1.3.

Max Kuhn. Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

Max Kuhn and Hadley Wickham. *recipes: Preprocessing tools to create design matrices*, 2020a. URL `https://CRAN.R-project.org/package=recipes`. R package version 0.1.16.

Max Kuhn and Hadley Wickham. *Tidymodels: A collection of packages for modeling and machine learning using tidyverse principles.*, 2020b. URL `https://www.tidymodels.org`.

Michel Lang, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*, 4(44):1903, 2019.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL `http://jmlr.org/papers/v18/16-558.html`.

Mitar Milutinovic, Atılım Güneş Baydin, Robert Zinkov, William Harvey, Dawn Song, Frank Wood, and Wade Shen. End-to-end training of differentiable pipelines across machine learning frameworks. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL `http://jmlr.org/papers/v12/pedregosa11a.html`.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333, 2011.

Wade Shen. Darpa's data driven discovery of models (D3M) and software defined hardware (SDH) programs. In D. Chen, H. Homayoun, and B. Taskin, editors, *Proceedings of the 2018 on Great Lakes Symposium on VLSI, GLSVLSI 2018, Chicago, IL, USA, May 23-25, 2018*, page 1. ACM, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

Raphael Sonabend, Franz J. Király, Andreas Bender, Bernd Bischl, and Michel Lang. mlr3proba: An R package for machine learning in survival analysis. *Bioinformatics*, 02 2021.

Alejandro Gonzalez Tineo. *baikal*, 2019. URL `https://github.com/alegonz/baikal`.

Davis Vaughan. *workflows: Modeling Workflows*, 2020. URL `https://CRAN.R-project.org/package=recipes`. R package version 0.2.1.

David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

Alain F Zuur, Elena N Ieno, Neil J Walker, Anatoly A Saveliev, and Graham M Smith. Zero-truncated and zero-inflated models for count data. In *Mixed Effects Models and Extensions in Ecology with R*, pages 261–293. Springer, New York, NY, USA, 2009.