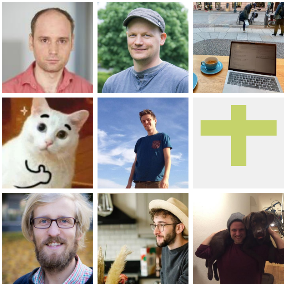
Object-oriented Machine Learning in R

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

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



Michel Lang, Bernd Bischl, Jakob Richter, Martin Binder, Marc Becker, Patrick Schratz, Raphael Sonabend, Lennart Schneider and more!

November 12, 2021

TODAY'S CONTENT

- Motivation: Why do we want to use mlr3?
- The mlr3verse
- ML Building-Blocks
- Example: Benchmark experiment

WHAT ARE YOU SUPPOSED TO TAKE AWAY

- Have an overview of most technical possibilities in mlr3
- Know where to find help
- Be able to find the best machine learning method for your problem

MOTIVATION: MAKING BENCHMARKS EASY!

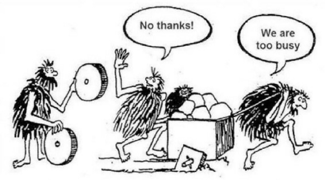
By unifying

- interface to train and predict methods,
- interface to learner's hyperparameters,
- interface to optimizers,
- resampling,
- preprocessing independently from the data,
- parallelization, and
- error handling

methods can be used interchangeably and can be easily benchmarked.

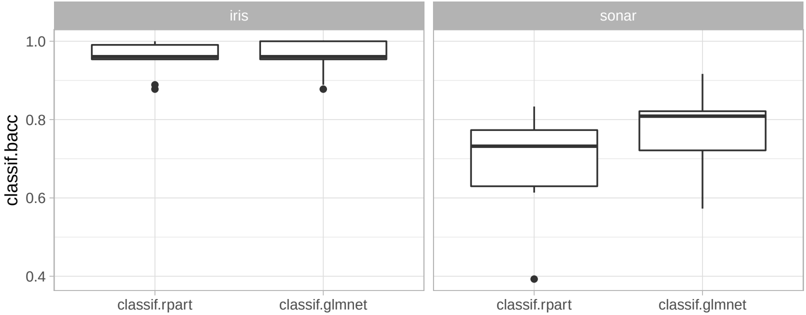
IS IT WORTH TO "LEARN" MLR3

- Avoid making mistakes by relying on tested functionality
  - predefined performance measures
  - resampling
  - ...
- Easily scale up your benchmark
  - integrated parallelization
  - benchmarking functions
  - on clusters: `batchtools` + `mlr3batchmark`
- New methods can be easily integrated in the mlr3verse



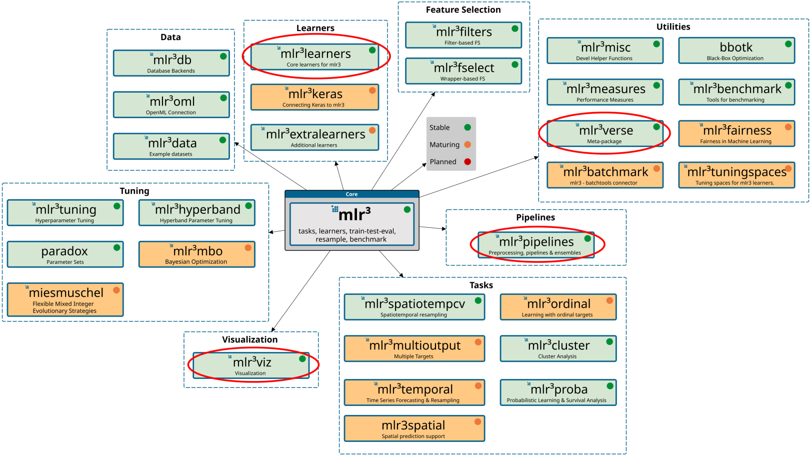
MLR3: A SHORT EXAMPLE

```r
library(mlr3verse)
tasks = list(
  as_task_classif(iris, target = "Species"), # task from df
  tsk("sonar") # example task
)
learners = lrns(c("classif.rpart", "classif.glmnet"))
bmg = benchmark_grid(tasks, learners, rsmp("cv"))
bmr = benchmark(bmg)
autoplot(bmr, measure = msr("classif.bacc")) # balanced accuracy
```



mlr3verse

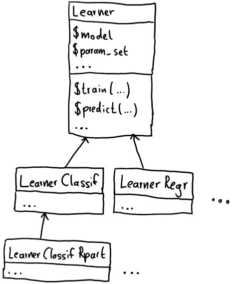
THE MLR3VERSE



Principles of mlr3

MLR3 PHILOSOPHY

- Object-orientation with **R6**
  - Make use of inheritance
  - Make slight use of reference semantics
- Embrace **data.table**:
  - Internal objects are stored in tabular structure.
  - List columns to arrange complex objects
- Be **light on dependencies**:
  - `R6`, `data.table`, `lgr`, ...
  - Special packages are loaded from mlr3 extension libraries



MLR3: OBJECTS AND FUNCTIONS

User created objects:

- Tasks: data + meta information
- Learner: ml algorithm + hyperparameter + model
- Measure: formula + meta information
- Resampling: strategy (+ indices)

Further objects:

- Prediction, ResampleResult, BenchmarkResult, ...

Functions to create objects:

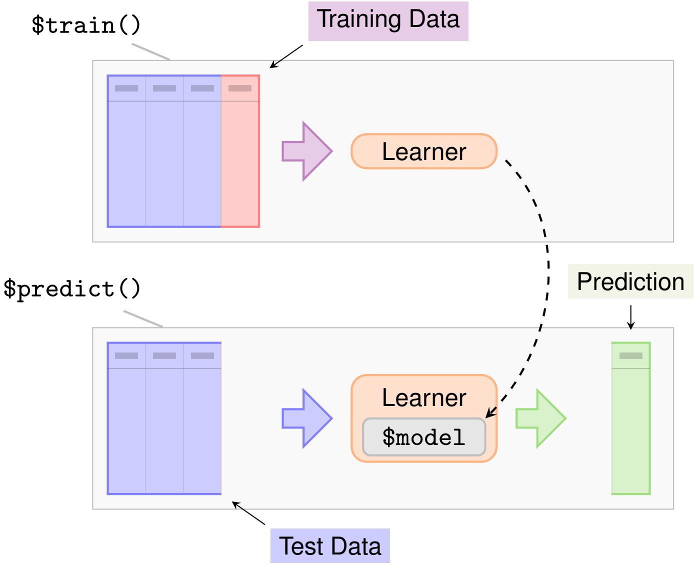
- `tsk(), as_task_*()`: Task
- `lrn(), lrns()`: Learners
- `msr(), msrs()`: Measures
- `rsmp()`: Resampling strategies

Hint: calling e.g. just `lrns()` prints all available learners in the `mlr_learners` dictionary. Dictionaries can get populated by add-on packages (e.g. `mlr3extralearners`)

Functions:

- `resample()`
- `benchmark_grid()` + `benchmark()`
- `future::plan()`: enables parallelization
- `mlr3viz::autoplot()`: visualizes mlr3 objects

LEARNING ALGORITHMS



TRAIN, PREDICT, SCORE

```
task = as_task_classif(iris, target = "Species")
```

Objects have *fields* that contain information about the object.

```
c(task$nrow, task$ncol)

#> [1] 150   5
```

Some can be overwritten:

```
learner = lrn("classif.fnn")
learner$param_set

#> <ParamSet>
#>            id    class lower upper nlevels default  value
#>        <char>   <char> <num> <num>   <num>  <list> <list>
#> 1:          k ParamInt     1   Inf     Inf       1
#> 2: algorithm ParamFct    NA    NA       3 kd_tree

learner$param_set$values$k = 4
```

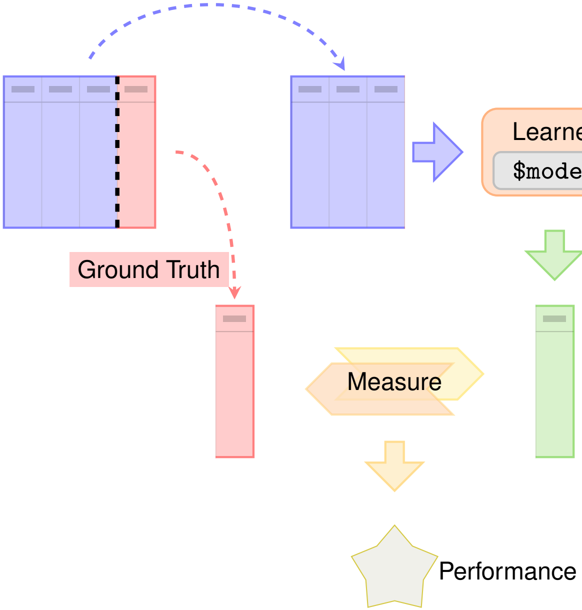
But are also checked for feasibility:

```
learner$param_set$values$k = -1

#> Error in self$assert(xs): Assertion on 'xs' failed: k: Element 1
is not >= 1.
```

TRAIN, PREDICT, SCORE

```
str(learner$model)

#>  NULL

learner$train(task, row_ids = (1:75) * 2 - 1)
str(learner$model)

#> List of 4
#>  $ formula:Class 'formula'  language Species ~ .
#>  $ data   :Classes 'data.table' and 'data.frame':
75 obs. of  5 variables:
#>   ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
#>   ..$ Petal.Length: num [1:75] 1.4 1.3 1.4 1.4 1.4 1.5 1.4 1.2 1.3 1.7 ...
#>   ..$ Petal.Width : num [1:75] 0.2 0.2 0.2 0.3 0.2 0.2 0.1 0.2 0.4 0.3 ...
#>   ..$ Sepal.Length: num [1:75] 5.1 4.7 5 4.6 4.4 5.4 4.8 5.8 5.4 5.7 ...
#>   ..$ Sepal.Width : num [1:75] 3.5 3.2 3.6 3.4 2.9 3.7 3 4 3.9 3.8 ...
#>   ..- attr(*, ".internal.selfref")=<externalptr>
#>  $ pv     :List of 1
#>   ..$ k: int 4
#>  $ kknn   : NULL

pred = learner$predict(task, row_ids = (1:75) * 2)
```

PERFORMANCE EVALUATION

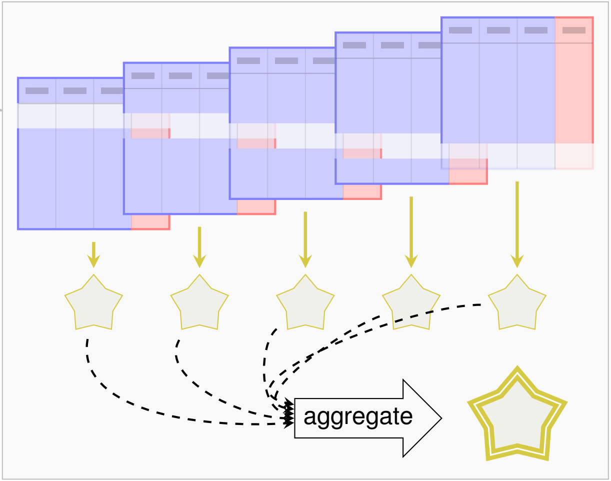


TRAIN, PREDICT, SCORE

```r
pred$confusion

#>             truth
#> response     setosa versicolor virginica
#>   setosa         25          0         0
#>   versicolor      0         24         2
#>   virginica       0          1        23

pred$score()

#> classif.ce
#>       0.04

pred$score(msr("classif.bacc"))

#> classif.bacc
#>         0.96

head(as.data.table(pred), 4)

#>    row_ids  truth response
#>      <int> <fctr>   <fctr>
#> 1:       2 setosa   setosa
#> 2:       4 setosa   setosa
#> 3:       6 setosa   setosa
#> 4:       8 setosa   setosa
```

RESAMPLING

`resample()`



RESAMPLING

- Resample description: How to split the data

```
cv5 = rsmp("cv", folds = 5)
```

- Use the `resample()` function for resampling:

```
task = tsk("iris")
learner = lrn("classif.rpart")
rr = resample(task, learner, cv5, store_models = TRUE)
```

(`store_models = TRUE` so we can access models in `rr` later.)

- We get a `ResamplingResult` object:

```
print(rr)

#> <ResampleResult> of 5 iterations
#> * Task: iris
#> * Learner: classif.rpart
#> * Warnings: 0 in 0 iterations
#> * Errors: 0 in 0 iterations
```

RESAMPLING

- Predictions of individual folds

```
predictions = rr$predictions()
predictions[[1]]
```

```
#> <PredictionClassif> for 30 observations:
#>     row_ids     truth  response
#>           1    setosa    setosa
#>           2    setosa    setosa
#>          11    setosa    setosa
#> ---
#>         147 virginica virginica
#>         148 virginica virginica
#>         149 virginica virginica
```

RESAMPLING

- Predictions of individual folds

```
predictions = rr$predictions()
predictions[[1]]
```

```
#> <PredictionClassif> for 30 observations:
#>     row_ids     truth  response
#>           1    setosa    setosa
#>           2    setosa    setosa
#>          11    setosa    setosa
#> ---
#>         147 virginica virginica
#>         148 virginica virginica
#>         149 virginica virginica
```

- Score of individual folds

```
scores = rr$score()
scores[1:3, c("iteration", "classif.ce")]
```

```
#>    iteration classif.ce
#>        <int>      <num>
#> 1:         1      0.033
#> 2:         2      0.100
#> 3:         3      0.033
```

RESAMPLING

- Access to models of individual folds (only if `$store_models = TRUE`)

```
rr$learners[[1]]$importance()
#>  Petal.Width Petal.Length Sepal.Length  Sepal.Width
#>           70           62           39           25
```

RESAMPLING

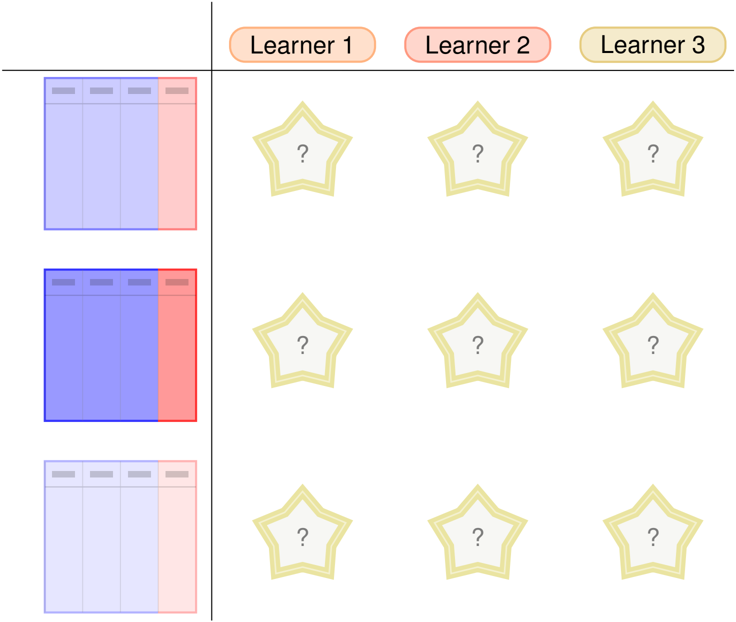
- Access to models of individual folds (only if `$store_models = TRUE`)

```
rr$learners[[1]]$importance()
#>  Petal.Width Petal.Length Sepal.Length  Sepal.Width
#>           70           62           39           25
```

- Aggregate over multiple folds:

```
sapply(rr$learners, function(x) x$importance()) %>%
  apply(1, mean)
#>  Petal.Width Petal.Length Sepal.Length  Sepal.Width
#>           71           66           44           29
```

PERFORMANCE COMPARISON: BENCHMARK



PERFORMANCE COMPARISON

- Multiple Learners, multiple Tasks:

```r
library("mlr3learners")
learners = list(lrn("classif.rpart"), lrn("classif.kknn"))
tasks = list(tsk("iris"), tsk("sonar"), tsk("wine"))
```

- Set up the *design* and execute benchmark:

```r
design = benchmark_grid(tasks, learners, cv5)
bmr = benchmark(design)
```

- We get a `BenchmarkResult` object which shows that `kknn` outperforms `rpart`:

```r
bmr_ag = bmr$aggregate()
bmr_ag[, c("task_id", "learner_id", "classif.ce")]
```

```
#>    task_id    learner_id classif.ce
#>     <char>        <char>      <num>
#> 1:    iris classif.rpart      0.060
#> 2:    iris  classif.kknn      0.047
#> 3:   sonar classif.rpart      0.265
#> 4:   sonar  classif.kknn      0.163
#> 5:    wine classif.rpart      0.118
#> 6:    wine  classif.kknn      0.039
```

BENCHMARK RESULT

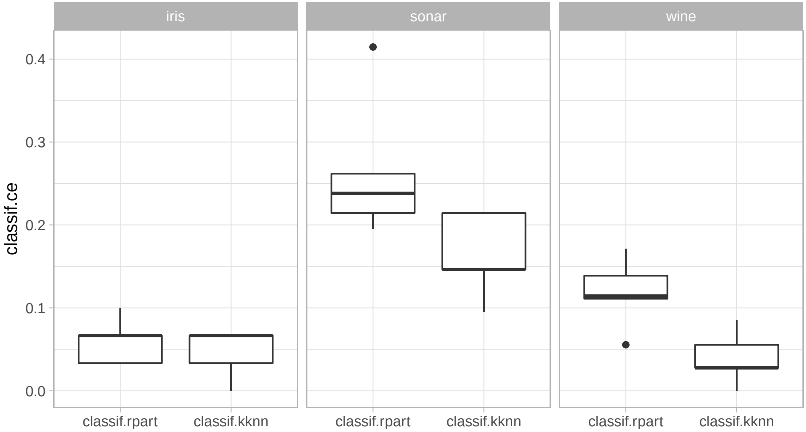
What exactly is a `BenchmarkResult` object?
Just like `Prediction` and `ResamplingResult`!

- Table representation using `as.data.table()`
- Active bindings and functions that make information easily accessible

BENCHMARK RESULT

The `mlr3viz` package contains `autoplot()` functions for many mlr3 objects

```r
library(mlr3viz)
autoplot(bmr)
```



BENCHMARK RESULT

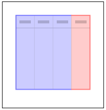
Many objects can be transformed into a `data.table()`.

```
as.data.table(bmr)[1:4, -1]

#                   task                         learner          resampling
#                 <list>                          <list>              <list>
# 1: <TaskClassif[47]> <LearnerClassifRpart[36]> <ResamplingCV[19]>
# 2: <TaskClassif[47]> <LearnerClassifRpart[36]> <ResamplingCV[19]>
# 3: <TaskClassif[47]> <LearnerClassifRpart[36]> <ResamplingCV[19]>
# 4: <TaskClassif[47]> <LearnerClassifRpart[36]> <ResamplingCV[19]>
#    iteration              prediction
#        <int>                  <list>
# 1:         1 <PredictionClassif[19]>
# 2:         2 <PredictionClassif[19]>
# 3:         3 <PredictionClassif[19]>
# 4:         4 <PredictionClassif[19]>
```

MLR3: SHORT RECAP

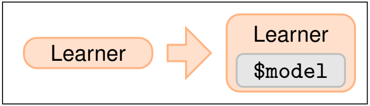
Ingredients:

Data



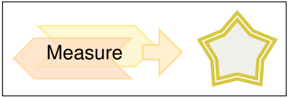
`TaskClassif`,
`TaskRegr`,
`tsk()`

Learning Algorithms



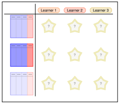
`lrn()` ⇒ `Learner`,
↪`Learner$train()`,
↪`Learner$predict()` ⇒ `Prediction`

Performance Evaluation



`rsmp()` ⇒ `Resampling`,
`msr()` ⇒ `Measure`,
`resample()` ⇒ `ResamplingResult`,
↪ `ResamplingResult$score()`,
↪ `ResamplingResult$aggregate()`

Performance Comparison



`benchmark_grid()`,
`benchmark()` ⇒ `BenchmarkResult`

mlr3pipelines

MLR3PIPELINES

Main author: Martin Binder (LMU)

Machine Learning Workflows:

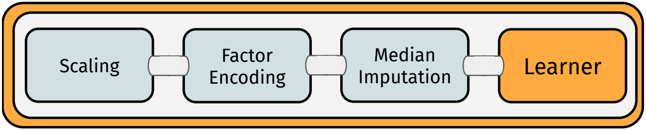
- **Preprocessing**: Feature extraction, feature selection, missing data imputation,. . .
- **Ensemble methods**: Model averaging, model stacking
- `mlr3`: modular model fitting

⇒ `mlr3pipelines`: modular ML workflows

MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

```
graph_pp = po("scale") %>>%
  po("encode") %>>%
  po("imputemedian") %>>%
  lrn("classif.rpart")
```

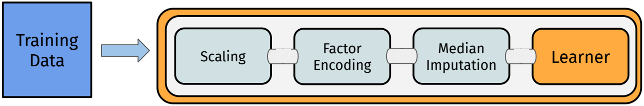


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

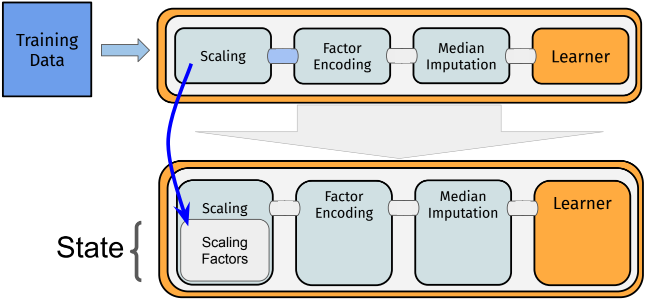


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

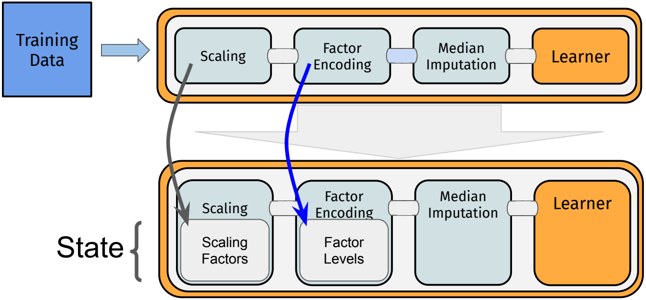


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

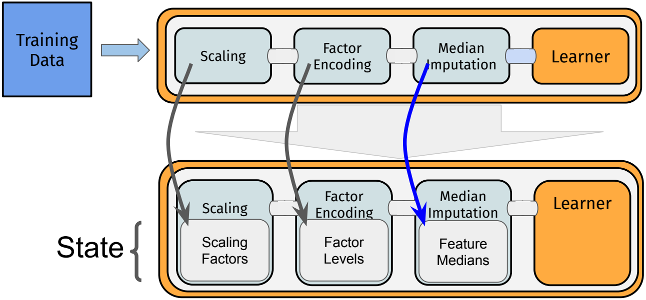


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

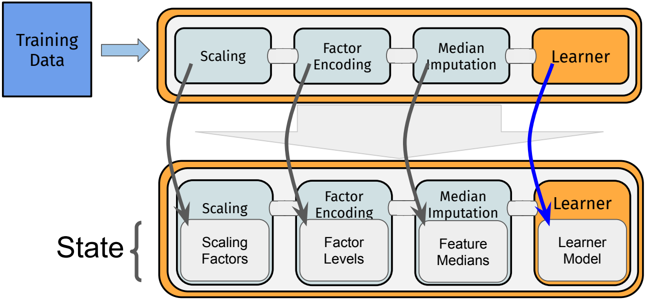


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

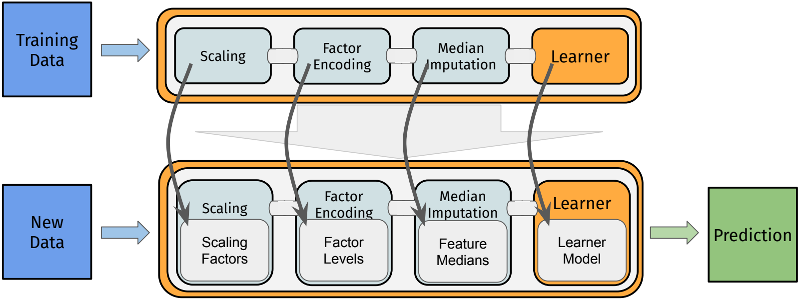


MLR3PIPELINES IN ACTION

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`
- `predict()`tion: Data propagates, uses `$states`

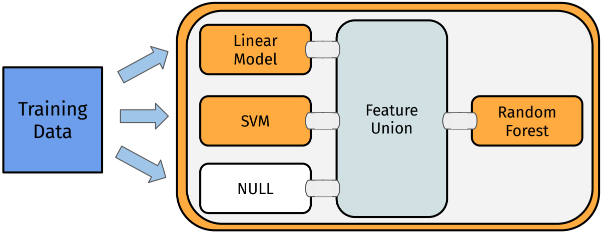
```r
glrn$predict(task)
```



MLR3PIPELINES IN ACTION

Ensemble Method: Stacking

```r
graph_stack = gunion(list(
      po("learner_cv", learner = lrn("regr.lm")),
      po("learner_cv", learner = lrn("regr.svm")),
      po("nop"))) %>>%
  po("featureunion") %>>%
  lrn("regr.ranger")
```

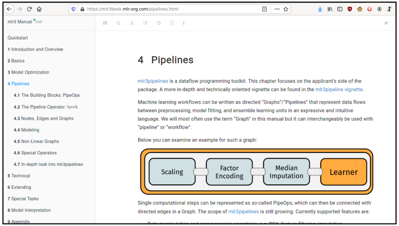


WHAT WAS NOT COVERED TODAY?

- Error handling (fallback learners), Database backends, Parallelization
- Hyperparameter tuning
- Cost-sensitive classification, survival learning, feature selection, geospatial methods
- Model interpretation (interpretable machine learning)
- ...

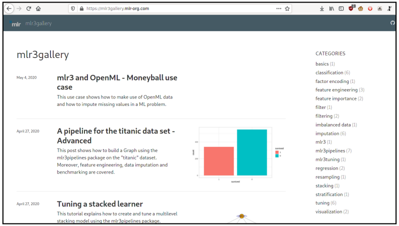
MLR3 RESOURCES

mlr3 book



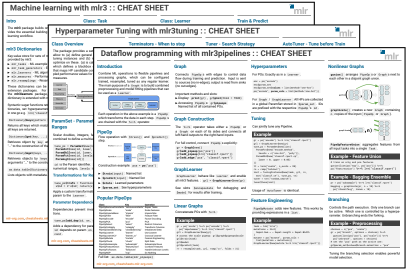
`https://mlr3book.mlr-org.com/`

mlr3 Use Case Gallery



`https://mlr3gallery.mlr-org.com/`

Cheat Sheets



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

More:

- Stackoverflow: `https://stackoverflow.com/tags/mlr3`
- Mattermost channel: `https://lmmisld-lmu-stats-slds.srv.mwn.de/mlr_invite/`
- GitHub Issue in one of the projects: `https://github.com/mlr-org/`