

# Package: mllrnrs (via r-universe)

September 3, 2024

**Title** R6-Based ML Learners for 'mlexperiments'

**Version** 0.0.4

**Description** Enhances 'mlexperiments'

<<https://CRAN.R-project.org/package=mlexperiments>> with additional machine learning ('ML') learners. The package provides R6-based learners for the following algorithms: 'glmnet' <<https://CRAN.R-project.org/package=glmnet>>, 'ranger' <<https://CRAN.R-project.org/package=ranger>>, 'xgboost' <<https://CRAN.R-project.org/package=xgboost>>, and 'lightgbm' <<https://CRAN.R-project.org/package=lightgbm>>. These can be used directly with the 'mlexperiments' R package.

**License** GPL (>= 3)

**URL** <https://github.com/kapsner/mllrnrs>

**BugReports** <https://github.com/kapsner/mllrnrs/issues>

**Depends** R (>= 3.6)

**Imports** data.table, kdry, mlexperiments, R6, stats

**Suggests** glmnet, lightgbm (>= 4.0.0), lintr, mlbench, mlr3measures, ParBayesianOptimization, quarto, ranger, splitTools, testthat (>= 3.0.1), xgboost

**VignetteBuilder** quarto

**Config/testthat/edition** 3

**Config/testthat/parallel** false

**Date/Publication** 2024-07-05 09:24:53.829592 UTC

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**SystemRequirements** Quarto command line tools (<https://github.com/quarto-dev/quarto-cli>).

**RoxygenNote** 7.3.1

**Repository** <https://kapsner.r-universe.dev>

**RemoteUrl** <https://github.com/kapsner/mllrnrs>

**RemoteRef** HEAD

**RemoteSha** d5e430689e1e81ba15f5c6803028d18993247f3b

## Contents

LearnerGlmnet . . . . .	2
LearnerLightgbm . . . . .	4
LearnerRanger . . . . .	7
LearnerXgboost . . . . .	9
<b>Index</b>	<b>12</b>

---

LearnerGlmnet	<i>R6 Class to construct a Glmnet learner</i>
---------------	---

---

### Description

The LearnerGlmnet class is the interface to the glmnet R package for use with the mlexperiments package.

### Details

Optimization metric: Can be used with

- [mlexperiments::MLTuneParameters](#)
- [mlexperiments::MLCrossValidation](#)
- [mlexperiments::MLNestedCV](#)

### Super class

[mlexperiments::MLLearnerBase](#) -> LearnerGlmnet

### Methods

#### Public methods:

- [LearnerGlmnet\\$new\(\)](#)
- [LearnerGlmnet\\$clone\(\)](#)

**Method** `new()`: Create a new LearnerGlmnet object.

*Usage:*

```
LearnerGlmnet$new(metric_optimization_higher_better)
```

*Arguments:*

`metric_optimization_higher_better` A logical. Defines the direction of the optimization metric used throughout the hyperparameter optimization.

*Returns:* A new LearnerGlmnet R6 object.

*Examples:*

```
LearnerGlmnet$new(metric_optimization_higher_better = FALSE)
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
LearnerGlmnet$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

### See Also

[glmnet::glmnet\(\)](#), [glmnet::cv.glmnet\(\)](#)

### Examples

```
# binary classification

library(mlbench)
data("PimaIndiansDiabetes2")
dataset <- PimaIndiansDiabetes2 |>
  data.table::as.data.table() |>
  na.omit()

seed <- 123
feature_cols <- colnames(dataset)[1:8]

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = feature_cols]
)
train_y <- as.integer(dataset[, get("diabetes")]) - 1L

fold_list <- splitTools::create_folds(
  y = train_y,
  k = 3,
  type = "stratified",
  seed = seed
)
glmnet_cv <- mlexperiments::MLCrossValidation$new(
  learner = mllrnrs::LearnerGlmnet$new(
    metric_optimization_higher_better = FALSE
  ),
  fold_list = fold_list,
  ncores = 2,
  seed = 123
)
glmnet_cv$learner_args <- list(
  alpha = 1,
  lambda = 0.1,
  family = "binomial",
```

```

    type.measure = "class",
    standardize = TRUE
  )
  glmnet_cv$predict_args <- list(type = "response")
  glmnet_cv$performance_metric_args <- list(positive = "1")
  glmnet_cv$performance_metric <- mlexperiments::metric("auc")

# set data
glmnet_cv$set_data(
  x = train_x,
  y = train_y
)

glmnet_cv$execute()

## -----
## Method `LearnerGlmnet$new`
## -----

LearnerGlmnet$new(metric_optimization_higher_better = FALSE)

```

---

LearnerLightgbm

*R6 Class to construct a LightGBM learner*


---

## Description

The `LearnerLightgbm` class is the interface to the `lightgbm` R package for use with the `mlexperiments` package.

## Details

Optimization metric: needs to be specified with the learner parameter `metric`. The following options can be set via `options()`:

- `"mlexperiments.optim.lgb.nrounds"` (default: 5000L)
- `"mlexperiments.optim.lgb.early_stopping_rounds"` (default: 500L)
- `"mlexperiments.lgb.print_every_n"` (default: 50L)
- `"mlexperiments.lgb.verbose"` (default: -1L)

`LearnerLightgbm` can be used with

- [mlexperiments::MLTuneParameters](#)
- [mlexperiments::MLCrossValidation](#)
- [mlexperiments::MLNestedCV](#)

## Super class

`mlexperiments::MLLearnerBase` -> `LearnerLightgbm`

## Methods

### Public methods:

- [LearnerLightgbm\\$new\(\)](#)
- [LearnerLightgbm\\$clone\(\)](#)

**Method** `new()`: Create a new `LearnerLightgbm` object.

*Usage:*

```
LearnerLightgbm$new(metric_optimization_higher_better)
```

*Arguments:*

`metric_optimization_higher_better` A logical. Defines the direction of the optimization metric used throughout the hyperparameter optimization.

*Returns:* A new `LearnerLightgbm` R6 object.

*Examples:*

```
LearnerLightgbm$new(metric_optimization_higher_better = FALSE)
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
LearnerLightgbm$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

[lightgbm::lgb.train\(\)](#), [lightgbm::lgb.cv\(\)](#)

## Examples

```
# binary classification

library(mlbench)
data("PimaIndiansDiabetes2")
dataset <- PimaIndiansDiabetes2 |>
  data.table::as.data.table() |>
  na.omit()

seed <- 123
feature_cols <- colnames(dataset)[1:8]

param_list_lightgbm <- expand.grid(
  bagging_fraction = seq(0.6, 1, .2),
  feature_fraction = seq(0.6, 1, .2),
  min_data_in_leaf = seq(10, 50, 10),
  learning_rate = seq(0.1, 0.2, 0.1),
  num_leaves = seq(10, 50, 10),
  max_depth = -1L
)
```

```

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = feature_cols]
)
train_y <- as.integer(dataset[, get("diabetes")]) - 1L

fold_list <- splitTools::create_folds(
  y = train_y,
  k = 3,
  type = "stratified",
  seed = seed
)
lightgbm_cv <- mlexperiments::MLCrossValidation$new(
  learner = mllrnrs::LearnerLightgbm$new(
    metric_optimization_higher_better = FALSE
  ),
  fold_list = fold_list,
  ncores = 2,
  seed = 123
)
lightgbm_cv$learner_args <- c(
  as.list(
    data.table::data.table(
      param_list_lightgbm[37, ],
      stringsAsFactors = FALSE
    ),
  ),
  list(
    objective = "binary",
    metric = "binary_logloss"
  ),
  nrounds = 45L
)
lightgbm_cv$performance_metric_args <- list(positive = "1")
lightgbm_cv$performance_metric <- mlexperiments::metric("auc")

# set data
lightgbm_cv$set_data(
  x = train_x,
  y = train_y
)

lightgbm_cv$execute()

## -----
## Method `LearnerLightgbm$new`
## -----

LearnerLightgbm$new(metric_optimization_higher_better = FALSE)

```

---

LearnerRanger      *R6 Class to construct a Ranger learner*

---

## Description

The LearnerRanger class is the interface to the ranger R package for use with the `mlexperiments` package.

## Details

Optimization metric:

- classification: classification error rate
- regression: mean squared error Can be used with
- [mlexperiments::MLTuneParameters](#)
- [mlexperiments::MLCrossValidation](#)
- [mlexperiments::MLNestedCV](#)

## Super class

[mlexperiments::MLLearnerBase](#) -> LearnerRanger

## Methods

### Public methods:

- [LearnerRanger\\$new\(\)](#)
- [LearnerRanger\\$clone\(\)](#)

**Method** `new()`: Create a new LearnerRanger object.

*Usage:*

```
LearnerRanger$new()
```

*Returns:* A new LearnerRanger R6 object.

*Examples:*

```
LearnerRanger$new()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
LearnerRanger$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

[ranger::ranger\(\)](#)

**Examples**

```

# binary classification

library(mlbench)
data("PimaIndiansDiabetes2")
dataset <- PimaIndiansDiabetes2 |>
  data.table::as.data.table() |>
  na.omit()

seed <- 123
feature_cols <- colnames(dataset)[1:8]

param_list_ranger <- expand.grid(
  num.trees = seq(500, 1000, 500),
  mtry = seq(2, 6, 2),
  min.node.size = seq(1, 9, 4),
  max.depth = seq(1, 9, 4),
  sample.fraction = seq(0.5, 0.8, 0.3)
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = feature_cols]
)
train_y <- as.integer(dataset[, get("diabetes")]) - 1L

fold_list <- splitTools::create_folds(
  y = train_y,
  k = 3,
  type = "stratified",
  seed = seed
)

ranger_cv <- mlexperiments::MLCrossValidation$new(
  learner = mllrnrs::LearnerRanger$new(),
  fold_list = fold_list,
  ncores = 2,
  seed = 123
)
ranger_cv$learner_args <- c(
  as.list(
    data.table::data.table(
      param_list_ranger[37, ],
      stringsAsFactors = FALSE
    ),
  ),
  list(classification = TRUE)
)
ranger_cv$performance_metric_args <- list(positive = "1")
ranger_cv$performance_metric <- mlexperiments::metric("auc")

# set data
ranger_cv$set_data(

```

```
x = train_x,  
y = train_y  
)  
  
ranger_cv$execute()  
  
## -----  
## Method `LearnerRanger$new`  
## -----  
  
LearnerRanger$new()
```

---

LearnerXgboost

*R6 Class to construct a Xgboost learner*

---

## Description

The LearnerXgboost class is the interface to the xgboost R package for use with the `mlexperiments` package.

## Details

Optimization metric: needs to be specified with the learner parameter `eval_metric`. The following options can be set via `options()`:

- `"mlexperiments.optim.xgb.nrounds"` (default: 5000L)
- `"mlexperiments.optim.xgb.early_stopping_rounds"` (default: 500L)
- `"mlexperiments.xgb.print_every_n"` (default: 50L)
- `"mlexperiments.xgb.verbose"` (default: FALSE)

LearnerXgboost can be used with

- [mlexperiments::MLTuneParameters](#)
- [mlexperiments::MLCrossValidation](#)
- [mlexperiments::MLNestedCV](#)

## Super class

[mlexperiments::MLLearnerBase](#) -> LearnerXgboost

## Methods

### Public methods:

- [LearnerXgboost\\$new\(\)](#)
- [LearnerXgboost\\$clone\(\)](#)

**Method** `new()`: Create a new `LearnerXgboost` object.

*Usage:*

```
LearnerXgboost$new(metric_optimization_higher_better)
```

*Arguments:*

`metric_optimization_higher_better` A logical. Defines the direction of the optimization metric used throughout the hyperparameter optimization.

*Returns:* A new `LearnerXgboost` R6 object.

*Examples:*

```
LearnerXgboost$new(metric_optimization_higher_better = FALSE)
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
LearnerXgboost$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

[xgboost::xgb.train\(\)](#), [xgboost::xgb.cv\(\)](#)

## Examples

```
# binary classification

library(mlbench)
data("PimaIndiansDiabetes2")
dataset <- PimaIndiansDiabetes2 |>
  data.table::as.data.table() |>
  na.omit()

seed <- 123
feature_cols <- colnames(dataset)[1:8]

param_list_xgboost <- expand.grid(
  subsample = seq(0.6, 1, .2),
  colsample_bytree = seq(0.6, 1, .2),
  min_child_weight = seq(1, 5, 4),
  learning_rate = seq(0.1, 0.2, 0.1),
  max_depth = seq(1, 5, 4)
)
```

```

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = feature_cols]
)
train_y <- as.integer(dataset[, get("diabetes")]) - 1L

fold_list <- splitTools::create_folds(
  y = train_y,
  k = 3,
  type = "stratified",
  seed = seed
)
)
xgboost_cv <- mlexperiments::MLCrossValidation$new(
  learner = mllrnrs::LearnerXgboost$new(
    metric_optimization_higher_better = FALSE
  ),
  fold_list = fold_list,
  ncores = 2,
  seed = 123
)
)
xgboost_cv$learner_args <- c(
  as.list(
    data.table::data.table(
      param_list_xgboost[37, ],
      stringsAsFactors = FALSE
    ),
  ),
  list(
    objective = "binary:logistic",
    eval_metric = "logloss"
  ),
  nrounds = 45L
)
)
xgboost_cv$performance_metric_args <- list(positive = "1")
xgboost_cv$performance_metric <- mlexperiments::metric("auc")

# set data
xgboost_cv$set_data(
  x = train_x,
  y = train_y
)
)

xgboost_cv$execute()

## -----
## Method `LearnerXgboost$new`
## -----

LearnerXgboost$new(metric_optimization_higher_better = FALSE)

```

# Index

`glmnet::cv.glmnet()`, [3](#)  
`glmnet::glmnet()`, [3](#)

`LearnerGlmnet`, [2](#)  
`LearnerLightgbm`, [4](#)  
`LearnerRanger`, [7](#)  
`LearnerXgboost`, [9](#)  
`lightgbm::lgb.cv()`, [5](#)  
`lightgbm::lgb.train()`, [5](#)

`mlexperiments::MLCrossValidation`, [2, 4, 7, 9](#)  
`mlexperiments::MLLearnerBase`, [2, 4, 7, 9](#)  
`mlexperiments::MLNestedCV`, [2, 4, 7, 9](#)  
`mlexperiments::MLTuneParameters`, [2, 4, 7, 9](#)

`ranger::ranger()`, [7](#)

`xgboost::xgb.cv()`, [10](#)  
`xgboost::xgb.train()`, [10](#)