Package 'mlsurvlrnrs'

July 5, 2024

Title R6-Based ML Survival Learners for 'mlexperiments'

Version 0.0.4

Description Enhances 'mlexperiments'
<https://CRAN.R-project.org/package=mlexperiments> with additional
machine learning ('ML') learners for survival analysis. The package
provides R6-based survival 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 'rpart'
<https://CRAN.R-project.org/package=rpart>. These can be used directly
with the 'mlexperiments' R package.

License GPL (>= 3)

URL <https://github.com/kapsner/mlsurvlrnrs>

BugReports <https://github.com/kapsner/mlsurvlrnrs/issues>

Depends R (>= 3.6)

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

Suggests glmnet, lintr, mlr3measures, ParBayesianOptimization, quarto,
ranger, rpart, splitTools, survival, testthat (>= 3.0.1),
xgboost

VignetteBuilder quarto

Config/testthat/edition 3

Config/testthat/parallel false

Date/Publication 2024-07-05 14:40:01 UTC

Encoding UTF-8

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

RoxygenNote 7.3.1

NeedsCompilation no

Author Lorenz A. Kapsner [cre, aut, cph]
(<https://orcid.org/0000-0003-1866-860X>)

Maintainer Lorenz A. Kapsner <lorenz.kapsner@gmail.com>

Repository CRAN

Contents

---

c_index   *c_index*

Description

Calculate the Harrell's concordance index (C-index)

Usage

```
c_index(ground_truth, predictions)
```

Arguments

ground_truth   A survival::Surv object with the ground truth.

predictions   A vector with predictions.

Details

A wrapper function around glmnet::Cindex() for use with mlexperiments.

See Also

glmnet::Cindex()

Examples

```
set.seed(123)
gt <- survival::Surv(
  time = rnorm(100, 50, 15),
  event = sample(0:1, 100, TRUE)
)
preds <- rbeta(100, 2, 5)
```

```
c_index(gt, preds)
```

LearnerSurvCoxPHCox   *R6 Class to construct a Cox proportional hazards survival learner*

Description

The `LearnerSurvCoxPHCox` class is the interface to perform a Cox regression with the `survival` R package for use with the `mlexperiments` package.

Details

Can be used with

- [mlexperiments::MLCrossValidation](#)

Super class

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

Methods

Public methods:

- [LearnerSurvCoxPHCox$new()](#)
- [LearnerSurvCoxPHCox$clone()](#)

Method `new()`: Create a new `LearnerSurvCoxPHCox` object.

Usage:

```
LearnerSurvCoxPHCox$new()
```

Returns: A new `LearnerSurvCoxPHCox` R6 object.

Examples:

```
LearnerSurvCoxPHCox$new()
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

[survival::coxph()](#)

Examples

```
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

split_vector <- splitTools::multi_strata(
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
             as.character() |>
             as.integer()),
  time = dataset[, get("time")],
  type = "right"
)

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


surv_coxph_cox_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvCoxPHCox$new(),
  fold_list = fold_list,
  ncores = 1L,
  seed = seed
)
surv_coxph_cox_optimizer$performance_metric <- c_index

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

```
surv_coxph_cox_optimizer$execute()


## ------------------------------------------------
## Method `LearnerSurvCoxPHCox$new`
## ------------------------------------------------

LearnerSurvCoxPHCox$new()
```

LearnerSurvGlmnetCox   *R6 Class to construct a Glmnet survival learner for Cox regression*

Description

The `LearnerSurvGlmnetCox` class is the interface to perform a Cox regression with the `glmnet` R package for use with the `mlexperiments` package.

Details

Optimization metric: C-index Can be used with

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

Super class

[`mlexperiments::MLLearnerBase`](#) -> `LearnerSurvGlmnetCox`

Methods

Public methods:

- [`LearnerSurvGlmnetCox$new()`](#)
- [`LearnerSurvGlmnetCox$clone()`](#)

Method `new()`: Create a new `LearnerSurvGlmnetCox` object.

Usage:

`LearnerSurvGlmnetCox$new()`

Returns: A new `LearnerSurvGlmnetCox` R6 object.

Examples:

`LearnerSurvGlmnetCox$new()`

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

glmnet::glmnet(), glmnet::cv.glmnet()

Examples

```r
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

param_list_glmnet <- expand.grid(
  alpha = seq(0, 1, .2)
)

ncores <- 2L

split_vector <- splitTools::multi_strata(
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
             as.character() |>
             as.integer()),
  time = dataset[, get("time")],
  type = "right"
)


fold_list <- splitTools::create_folds(
  y = split_vector,
  k = 3,
  type = "stratified",
```

```
    seed = seed
)

surv_glmnet_cox_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvGlmnetCox$new(),
  fold_list = fold_list,
  ncores = ncores,
  seed = seed
)
surv_glmnet_cox_optimizer$learner_args <- list(
  alpha = 0.8,
  lambda = 0.002
)
surv_glmnet_cox_optimizer$performance_metric <- c_index

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

surv_glmnet_cox_optimizer$execute()

## ------------------------------------------------
## Method `LearnerSurvGlmnetCox$new`
## ------------------------------------------------

LearnerSurvGlmnetCox$new()
```

LearnerSurvRangerCox   *R6 Class to construct a Ranger survival learner for Cox regression*

Description

The `LearnerSurvRangerCox` class is the interface to perform a Cox regression with the `ranger` R package for use with the `mlexperiments` package.

Details

Optimization metric: C-index Can be used with

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

Super class

[`mlexperiments::MLLearnerBase`](#) -> `LearnerSurvRangerCox`

Methods

> **Public methods:**
>
> - `LearnerSurvRangerCox$new()`
> - `LearnerSurvRangerCox$clone()`
>
> **Method** `new()`: Create a new `LearnerSurvRangerCox` object.
>
> *Usage:*
>
> `LearnerSurvRangerCox$new()`
>
> *Returns:* A new `LearnerSurvRangerCox` R6 object.
>
> *Examples:*
>
> `LearnerSurvRangerCox$new()`
>
> **Method** `clone()`: The objects of this class are cloneable with this method.
>
> *Usage:*
>
> `LearnerSurvRangerCox$clone(deep = FALSE)`
>
> *Arguments:*
>
> deep Whether to make a deep clone.

See Also

ranger::ranger()

Examples

```
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

param_list_ranger <- expand.grid(
  sample.fraction = seq(0.6, 1, .2),
  min.node.size = seq(1, 5, 4),
  mtry = seq(2, 6, 2),
  num.trees = c(5L, 10L),
  max.depth = seq(1, 5, 4)
)

ncores <- 2L

split_vector <- splitTools::multi_strata(
```

```
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
             as.character() |>
             as.integer()),
  time = dataset[, get("time")],
  type = "right"
)

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

surv_ranger_cox_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvRangerCox$new(),
  fold_list = fold_list,
  ncores = ncores,
  seed = seed
)
surv_ranger_cox_optimizer$learner_args <- as.list(
  data.table::data.table(param_list_ranger[1, ], stringsAsFactors = FALSE)
)
surv_ranger_cox_optimizer$performance_metric <- c_index

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

surv_ranger_cox_optimizer$execute()


## ------------------------------------------------
## Method `LearnerSurvRangerCox$new`
## ------------------------------------------------

LearnerSurvRangerCox$new()
```

LearnerSurvRpartCox   *LearnerSurvRpartCox R6 class*

Description

This learner is a wrapper around `rpart::rpart()` in order to fit recursive partitioning and regression trees with survival data.

Details

Optimization metric: C-index * Can be used with

- mlexperiments::MLTuneParameters
- mlexperiments::MLCrossValidation
- mlexperiments::MLNestedCV

Implemented methods:

- `$fit` To fit the model.
- `$predict` To predict new data with the model.
- `$cross_validation` To perform a grid search (hyperparameter optimization).
- `$bayesian_scoring_function` To perform a Bayesian hyperparameter optimization.

Parameters that are specified with `parameter_grid` and / or `learner_args` are forwarded to rpart's argument `control` (see `rpart::rpart.control()` for further details).

Super class

`mlexperiments::MLLearnerBase` -> LearnerSurvRpartCox

Methods

Public methods:

- `LearnerSurvRpartCox$new()`
- `LearnerSurvRpartCox$clone()`

Method `new()`: Create a new `LearnerSurvRpartCox` object.

Usage:

`LearnerSurvRpartCox$new()`

Details: This learner is a wrapper around `rpart::rpart()` in order to fit recursive partitioning and regression trees with survival data.

Examples:

`LearnerSurvRpartCox$new()`

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

rpart::rpart(), c_index(), rpart::rpart.control()

rpart::rpart(), c_index(),

Examples

```
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

ncores <- 2L

split_vector <- splitTools::multi_strata(
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
             as.character() |>
             as.integer()),
  time = dataset[, get("time")],
  type = "right"
)

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

```
surv_rpart_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvRpartCox$new(),
  fold_list = fold_list,
  ncores = ncores,
  seed = seed
)
surv_rpart_optimizer$learner_args <- list(
  minsplit = 10L,
  maxdepth = 20L,
  cp = 0.03,
  method = "exp"
)
surv_rpart_optimizer$performance_metric <- c_index

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

surv_rpart_optimizer$execute()



## ------------------------------------------------
## Method `LearnerSurvRpartCox$new`
## ------------------------------------------------

LearnerSurvRpartCox$new()
```

LearnerSurvXgboostAft   *R6 Class to construct a Xgboost survival learner for accelerated fail-
                          ure time models*

Description

The `LearnerSurvXgboostAft` class is the interface to accelerated failure time models with the
xgboost R package for use with the mlexperiments package.

Details

Optimization metric: needs to be specified with the learner parameter `eval_metric`. Can be used
with

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

Also see the official xgboost documentation on aft models: https://xgboost.readthedocs.io/en/stable/tutorials/aft_survival_analysis.html

Super classes

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

Methods

Public methods:

- [LearnerSurvXgboostAft$new()](#)
- [LearnerSurvXgboostAft$clone()](#)

Method new(): Create a new LearnerSurvXgboostAft object.

Usage:

```
LearnerSurvXgboostAft$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 LearnerSurvXgboostAft R6 object.

Examples:

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

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

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

Examples

```
# execution time >2.5 sec
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

param_list_xgboost <- expand.grid(
  objective = "survival:aft",
```

```
  eval_metric = "aft-nloglik",
  subsample = seq(0.6, 1, .2),
  colsample_bytree = seq(0.6, 1, .2),
  min_child_weight = seq(1, 5, 4),
  learning_rate = c(0.1, 0.2),
  max_depth = seq(1, 5, 4)
)
ncores <- 2L

split_vector <- splitTools::multi_strata(
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
             as.character() |>
             as.integer()),
  time = dataset[, get("time")],
  type = "right"
)

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

surv_xgboost_aft_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvXgboostAft$new(
    metric_optimization_higher_better = FALSE
  ),
  fold_list = fold_list,
  ncores = ncores,
  seed = seed
)
surv_xgboost_aft_optimizer$learner_args <- c(as.list(
  data.table::data.table(param_list_xgboost[1, ], stringsAsFactors = FALSE)
),
nrounds = 45L
)
surv_xgboost_aft_optimizer$performance_metric <- c_index

# set data
surv_xgboost_aft_optimizer$set_data(
  x = train_x,
  y = train_y
```

```
)

surv_xgboost_aft_optimizer$execute()



## ------------------------------------------------
## Method `LearnerSurvXgboostAft$new`
## ------------------------------------------------

LearnerSurvXgboostAft$new(metric_optimization_higher_better = FALSE)
```

LearnerSurvXgboostCox   *R6 Class to construct a Xgboost survival learner for Cox regression*

Description

The LearnerSurvXgboostCox class is the interface to perform a Cox regression with the xgboost R package for use with the mlexperiments package.

Details

Optimization metric: needs to be specified with the learner parameter eval_metric. Can be used with

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

Super classes

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

Methods

Public methods:

- [LearnerSurvXgboostCox$new()](#)
- [LearnerSurvXgboostCox$clone()](#)

Method new(): Create a new LearnerSurvXgboostCox object.

Usage:

LearnerSurvXgboostCox$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 `LearnerSurvXgboostCox` R6 object.

Examples:

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

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

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

Examples

```
# execution time >2.5 sec
# survival analysis

dataset <- survival::colon |>
  data.table::as.data.table() |>
  na.omit()
dataset <- dataset[get("etype") == 2, ]

seed <- 123
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(dataset)[3:(ncol(dataset) - 1)]

param_list_xgboost <- expand.grid(
  objective = "survival:cox",
  eval_metric = "cox-nloglik",
  subsample = seq(0.6, 1, .2),
  colsample_bytree = seq(0.6, 1, .2),
  min_child_weight = seq(1, 5, 4),
  learning_rate = c(0.1, 0.2),
  max_depth = seq(1, 5, 4)
)
ncores <- 2L

split_vector <- splitTools::multi_strata(
  df = dataset[, .SD, .SDcols = surv_cols],
  strategy = "kmeans",
  k = 4
)

train_x <- model.matrix(
  ~ -1 + .,
  dataset[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])]
```

```
)
train_y <- survival::Surv(
  event = (dataset[, get("status")] |>
              as.character() |>
              as.integer()),
  time = dataset[, get("time")],
  type = "right"
)

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

surv_xgboost_cox_optimizer <- mlexperiments::MLCrossValidation$new(
  learner = LearnerSurvXgboostCox$new(
    metric_optimization_higher_better = FALSE
  ),
  fold_list = fold_list,
  ncores = ncores,
  seed = seed
)
surv_xgboost_cox_optimizer$learner_args <- c(as.list(
  data.table::data.table(param_list_xgboost[1, ], stringsAsFactors = FALSE)
),
nrounds = 45L
)
surv_xgboost_cox_optimizer$performance_metric <- c_index

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

surv_xgboost_cox_optimizer$execute()



## ------------------------------------------------
## Method `LearnerSurvXgboostCox$new`
## ------------------------------------------------

LearnerSurvXgboostCox$new(metric_optimization_higher_better = FALSE)
```

Index